Fast Elliptic Curve Point Multiplication for WSNs

Ravi Kishore Kodali, Srikrishna Karanam, Kashyapkumar Patel and Harpreet Singh Budwal
Department of Electronics and Communications Engineering
National Institute of Technology, Warangal, INDIA
E-mail: ravikkodali@gmail.com

***Abstract*—Wireless sensor networks facilitate real-time data processing in many applications such as intrusion detection and military surveillance. These applications inherently demand a high level of security. Public key cryptographic techniques such as Elliptic Curve Cryptography (ECC) provide a flexible interface to ensure security, requiring no pre-distribution of keys. However, ECC is very compute-intensive, owing to the computationally complex elliptic curve point multiplication operation. This work presents a technique to speed up the elliptic curve point multiplication operation, achieving a cost reduction of about 48% over the double and add algorithm and about 40% over the width-w Non Adjacent Form (NAF) algorithm. Additionally, results of its software implementation are also presented, simulating the specifications of MEMSIC's wireless sensor network development kit. Finally, we conclude that the timing results obtained from the software implementation conform to the theoretical results.**

***Index Terms*—WSN, ECC, Point Multiplication, NAF**

I. Introduction

Wireless Sensor Networks consist of numerous tiny nodes, each capable of performing real-time data processing. Nowadays, these networks are widely being used in applications such as military surveillance and intrusion detection. The sensor nodes belonging to a network interact with each other in a distributed fashion, collecting and processing information from their physical surroundings. However, these sensor nodes have several constraints when compared to a traditional desktop computer, with the most notable being limited memory and processing capabilities. These limitations pose a serious challenge to implementing security schemes on such devices. Despite these limitations, it is essential that these devices are installed with adequate security schemes to protect data integrity.

Public key cryptographic (PKC) schemes offer a flexible interface, and do not require pre-distribution of keys. The PKC standard widely used today is RSA. However, ECC [6] [9] is emerging as a viable alternative to RSA, for ECC provides same level of security as RSA at much lower sizes of the key. ECC, in recent years, has also been standardized by organizations such as the ISO [1] and has gained acceptance among various industry sectors. In spite of the computational requirements, 160-bit ECC has been implemented on Atmel's ATmega 128, demonstrating that ECC is indeed feasible on resource constrained wireless sensor networks [3]. This work focuses on speeding up the point multiplication operation, a key step in any application using Elliptic Curve Cryptography.

The paper is organized as follows: section I-A gives a brief introduction to the hardware platform which will be simulated in our implementations, section I-B gives a brief introduction to Elliptic Curve Cryptography, section II gives a brief literature survey and enumerates our contributions and sections III, IV and V describe our technique for elliptic curve point multiplication. Final theoretical and implementation results are given in section VI.

TABLE I
Features of MEMSIC's WSN development kit

Feature	Type/Value
Chip	ATmega 1281
Clock Frequency	7.37 MHz
Bit Width	8−bit
Program Memory	128 KB
Static RAM	8 KB
Power Source Type	AA, 2x
Power Source Capacity	2000 mA-Hr

A. MEMSIC's Wireless Sensor Network Development kit

The hardware platform used by the development kit is IRIS, the latest generation of motes from MEMSIC [8]. Table I gives the salient features of the development kit that will help us set up the simulation environment to verify various hypotheses and theoretical results.

B. Elliptic Curve Cryptography

ECC uses elliptic curves in which the variables and coefficients from the elements of a finite field, Z_p (p → prime). The elliptic curve equation that we work with for ECC applications is

$$y^2 = (x^3 + ax + b)$$

The set of all points satisfying this equation for given values of p, a and b is denoted by $E_p(a, b)$. Given two points P and Q in $E_p(a, b)$, it would be useful to calculate $R = P + Q$, whose coordinates are:

$$x_R = (\lambda^2 - x_P - x_Q),$$

$$y_R = (\lambda(x_P - x_R) - y_P),$$

where $\lambda = \frac{y_Q - y_P}{x_Q - x_P}$. This operation is called point addition. In case of P = Q, i.e, R = 2P (called point doubling), λ would be given as

$$\lambda = \frac{3x_P^2 + a}{2y_P}$$

A fundamental operation in ECC is point multiplication. The point multiplication operation is defined as $B = rA$, $A, B \in$

$E_p(a, b)$ and $r \in Z$, the set of integers. rA can be written as, *i.e.*, $rA = A + A + A + \cdots + A$ (r times). Therefore, point multiplication involves repeated use of point addition and point doubling equations given above. The most popular algorithm for point multiplication, called the "double and add" algorithm [5] is given in algorithm 1.

Algorithm 1 The Double and Add algorithm

```
  Input: r(in its binary form), A
1. B = $\phi$ (infinity point).
2. for $i = l - 1$ downto 0
      $B = 2B$.
      if $r_i = 1$, $B = B + A$.
3. Return $B$
  Output: $B = rA$
```

II. Literature survey and our contributions

Many improved versions of the traditional double and add algorithm have been reported previously to mitigate the cost complexity of the elliptic curve point multiplication operation. Point subtraction over a prime field is as efficient as point addition, which has motivated the development of signed representations for the scalar r [5]. A windowed approach to point multiplication was proposed by Solinas [11] and achieved reasonable efficiency. Recently, multi-radix representations of the scalars are becoming popular to achieve cost efficiency. In [2], a method of scalar representation using radices 2 and 3 was introduced. In [7], a multi-base NAF form, which efficiently represents integers using multiple bases, was introduced.

As will be seen in section III, the width-w NAF representation offers a lower computation cost when compared to the double and add algorithm. However, the width-w NAF algorithm requires odd point pre-computation, *i.e.*, $A_i = iA$ for $i = 3, \cdots, 2^w - 1$. These points are computed using the conventional double and add algorithm and hence this step is a bottleneck in the performance of the width$-w$ NAF algorithm. In this paper, we present a technique that results in a speed up in both the double and add and width$-w$ NAF algorithms. We achieve this through the use of a mixed coordinate system, employing different coordinate systems for the doubling and addition field operations. The contributions of this paper are enumerated below:

1) a *mixed* coordinate system is used to reduce the computation cost of point multiplication, and
2) implementation results of simulations performed using AVR Studio that simulates the performance of our technique on the hardware described in section I-A are also given.

III. The NAF representation

The double and add algorithm makes use of the binary representation of the scalar r. If the scalar's binary representation consists of t bits, the expected number of ones in it

Algorithm 2 Determining width-w NAF representation

```
  Input: $r$
  Output: width$-w$ NAF of $r$
1. $i = 0$.
2. while $r \geq 1$ do
     if $r$ is odd
        $r_i = u$, where $u \equiv r \bmod 2^w$
        $r = r - r_i$
     else $r_i = 0$.
3. $r = r/2$
4. $i = i + 1$
  Return $\{r_{i-1}, r_{i-2}, \cdots, r_1, r_0\}$
```

would be $t/2$. This results in the double and add algorithm costing $t/2$ point additions and t point doublings. For a 160$-$bit scalar, this cost turns out to be pretty large. This motivates the development of schemes which reduce the average density of ones in the representation, resulting in lesser point addition operations to be performed. One such scheme is the signed digit representation, according to which $r = \sum_0^{t-1} r_i 2^i$, $r_i \in \{0, \pm 1\}$. A class of signed integer representations which has proved to be useful is the Non-Adjacent Form (NAF). In this representation, the non-zero coefficient density is about 1/3 on an average, resulting in a reduction in the number of field addition operations to be performed. This computation cost can be further reduced by using a variant of the NAF, called the *width$-w$* NAF. In a width-w NAF, each coefficient r_i is odd and satisfies the following:

- $r_i \leq (2^w - 1)$.
- Out of any sequence of w consecutive coefficients, at most one is non-zero.

In this case, the average non-zero coefficient density is approximately $\frac{1}{w+1}$.

The algorithm for width$-w$ NAF point multiplication is given in algorithm 3 [11].

Algorithm 3 Width$-w$ NAF Point Multiplication

```
  Input: $w$, width$-w$ NAF of $r$, $A$
1. Determine $A_i = iA$ for $i = 3, \cdots, 2^w - 1$.
2. $B = \phi$.
3. for $i = t - 1$ downto 0 do
     $B = 2B$.
     if $r_i \neq 0$ then
        if $r_i > 0$ then $B = B + A_{r_i}$.
        else $B = B - A_{r_i}$.
  Return $B$.
  Output: $B = rA$
```

IV. Coordinate Systems and Computation Cost

The coordinate system in which the elliptic curve points are represented also impacts the cost of the point multiplication operation. This section tabulates the formulae for point addition and doubling operations and the corresponding computation costs incurred in various coordinate systems.

TABLE II
Point doubling and addition costs in various coordinate systems

Operation	Affine	Projective	Jacobian
Point Addition	$I + 2M + S$	$12M + 2S$	$12M + 4S$
Point Doubling	$I + 2M + 2S$	$7M + 5S$	$4M + 6S$

TABLE III
Coordinate System Conversions

Coordinate System	Typical Point	Substitution
Affine	(x, y)	–
Projective	(X, Y, Z)	$x = X/Z,\ y = X/Z$
Jacobian	(X, Y, Z)	$x = X/Z^2,\ y = Y/Z^2$

TABLE IV
The mixed coordinate system

Operation	Coordinate system
Point addition	Projective
Point doubling	Jacobian

As can be seen from table II, the Jacobian coordinate system offers faster doubling than the projective coordinate system. However, the projective coordinate system offers faster addition. Hence, using different coordinate systems for addition and doubling might potentially turn out to be very cost-effective. Therefore, in all our subsequent analysis, the Jacobian coordinate system is used for point addition and projective coordinate system is used for the point doubling operation. This is noted in table IV.

V. Mixed Coordinate system based width$-w$ NAF multiplication algorithm

Here, the mixed coordinate system based width$-w$ NAF multiplication algorithm is presented. It is noted in algorithm 4. A detailed explanation including cost analysis is given in the next section.

A. Cost analysis

In this section, the computational requirements of the mixed coordinate system based width-w NAF point multiplication algorithm is analysed. Assume a t bit width$-w$ NAF representation of the scalar r. As already noted, the non-zero coefficient density in the NAF representation of r is $\frac{1}{w+1}$. Therefore, the average number of non-zero coefficients in a $t-$bit NAF representation of r would be $\frac{t}{w+1}$. The computational cost of algorithm 3 would hence be equivalent to t point doublings and $\frac{t}{w+1}$ point additions. However, the use of the mixed coordinate system would result in additional overhead of converting points from one coordinate system to another. The entire cost analysis of algorithm 3 using mixed coordinates is enumerated below:

TABLE V
Formulae for point addition

Coordinate System	Point Addition ($R = P + Q$)
Projective	$X_R = vA$ $Y_R = u(v^2X_PZ_Q - A) - v^3Y_PZ_Q$ $Z_R = v^3Z_PZ_Q$ where $u = Y_QZ_P - Y_PZ_Q$, $v = X_QZ_P - X_PZ_Q$, and $A = u^2Z_PZ_Q - v^3 - 2v^2X_PZ_Q$
Jacobian	$X_R = -H^3 - 2U_1H^2 + r^2$ $Y_R = -S_1H^3 + r(U_1H^2 - X_R)$ $Z_R = Z_PZ_QH$ where $U_1 = X_PZ_Q^2$, $U_2 = X_QZ_P^2$, $S_1 = Y_PZ_Q^3$, $S_2 = Y_QZ_P^3$, $H = U_2 - U_1$ and $r = S_2 - S_1$

TABLE VI
Formulae for point doubling

Coordinate System	Point Doubling ($R = 2P$)
Projective	$X_R = 2hs$ $Y_R = w(4B - h) - 8Y_P^2s^2$ $Z_R = 8s^3$ where $w = aZ_P^2 + 3X_P^2$, $s = Y_PZ_P$, $B = X_PY_Ps$, and $h = w^2 - 8B$
Jacobian	$X_R = T$ $Y_R = -8Y_1^4 + M(S - T)$ $Z_R = 2Y_PZ_P$ where $U_1 = X_PZ_Q^2$, $U_2 = X_QZ_P^2$, $S_1 = Y_PZ_Q^3$, $S_2 = Y_QZ_P^3$, $H = U_2 - U_1$ and $r = S_2 - S_1$

1. The original point (affine system) would have to be converted into Jacobian coordinates, for the first step in algorithm 3 is doubling, and according to the mixed coordinate system described above, the Jacobian coordinate system is used for doubling and Projective coordinate system is used addition. The cost of this conversion from Affine to Jacobian would be $S + 2M$.
2. Since t point doublings are performed, the cost would be t times the cost of point doubling in Jacobian coor-

Algorithm 4 Mixed coordinate system based width$-w$ NAF Point Multiplication Algorithm

```
   Input: $w$, width$-w$ NAF of $r$, $A$ (affine coordinates)
1. Compute $A_i = iA$ for $i = 3, 5, \cdots, 2^w - 1$.
2. $B = \phi$.
3. $B_{affine} \rightarrow B_{jacobian}$
4. for $i = t - 1$ downto 0 do
   4.1 $B = 2B$.
   4.2 if $r_i \neq 0$ then
       4.2.1 $B_{jacobian} \rightarrow B_{projective}$
       4.2.2 $A_{r_i affine} \rightarrow A_{r_i projective}$
       4.2.3 if $r_i > 0$ then $B = B + A_{r_i}$.
       4.2.4 else $B = B - A_{r_i}$.
   4.3 $B_{projective} \rightarrow B_{jacobian}$
5. $B_{projective} \rightarrow B_{affine}$
6. Return $B$.
   Output: $B = rA$
```

dinates, *i.e.*, $t(6S + 4M)$.

3. The points would have to be converted from Jacobian to Projective system $\frac{t}{w+1}$ times, for the point addition operation is performed those many times, as can be observed from algorithm 3. The cost of this conversion would be $\frac{t}{w+1}(I + 2M)$.
4. Next, the point addition operation in projective coordinate system is performed $\frac{t}{w+1}$ times. In this step, it is also required that the pre-computed points A_{r_i} are converted from affine to projective system. The net cost of this step would be $\frac{t}{w+1}(12M + 2S + 2M)$.
5. Since the addition to doubling iteration happens $\frac{t}{w+1}$ times, the result of addition would have to be converted, which is in Projective system, to the Jacobian system. The cost of this step would be $\frac{t}{w+1}(2M)$.
6. Finally, the result is converted from Projective system to affine system.

Hence, the total cost C_{mnaf} would add upto

$$C_{mnaf} = (4M + S) + t(6S + 4M) + \frac{t}{w+1}(I + 2S + 18M) \quad (1)$$

Rewriting this equation,

$$C_{mnaf} = I(\frac{t}{w+1}) + M(4 + 4t + \frac{18t}{w+1}) + S(6t + 1 + \frac{2t}{w+1}) \quad (2)$$

B. Theoretical Comparisons

In this section, the cost of the mixed coordinate system based width-w NAF point multiplication method is compared with the traditional double and add and width$-w$ NAF algorithms. The average cost C_{da} of the traditional double and add algorithm, using affine coordinates, would be

$$C_{da} = t(I + 2M + 2S) + \frac{t}{2}(I + 2M + S) \quad (3)$$

Rewriting,

$$C_{da} = I(\frac{3t}{2}) + M(3t) + S(\frac{5t}{2}) \quad (4)$$

Next, in the case of the width$-w$ NAF method of multiplication, as noted in section III, the average density of non-zero coefficients is approximately $\frac{1}{w+1}$. Hence, the cost of the width$-w$ NAF multiplication algorithm will be

$$C_{naf} = t(I + 2M + 2S) + \frac{t}{w+1}(I + 2M + S) \quad (5)$$

Rewriting,

$$C_{naf} = I(t + \frac{t}{w+1}) + M(2t + \frac{2t}{w+1}) + S(2t + \frac{t}{w+1}) \quad (6)$$

For software implementations of ECC, an assumption of $8M \leq I \leq 20M$ is reasonable [4]. However, this assumption might not hold good in all cases. Specifically, it has been established by Seysen [10] that this I/M ratio can get as large as 100 in smart cards, where a cryptography specific co-processor is employed to perform computations. Hence, in this section, a generic I/M ratio analysis is performed for the mixed-coordinate based width$-w$ NAF method of point multiplication.

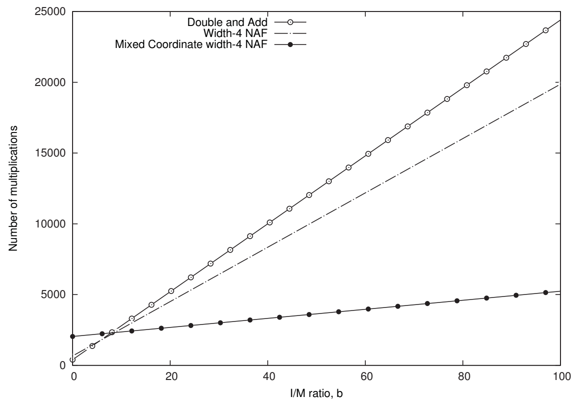

Fig. 1. A plot showing the number of multiplications versus the I/M ratio b; key size = 160 bits.

Let I/M ratio be equal to b. Setting the S/M ratio to 0.8, equations 2 and 4 become,

$$C_{mnaf} = M(4.8 + 8.8t + \frac{(19.6 + b)t}{w+1}) \quad (7)$$

$$C_{da} = M\frac{(3b + 5)t}{2} \quad (8)$$

$$C_{naf} = M((b + 3.6)t + \frac{(b + 2.8)t}{w+1}) \quad (9)$$

For a key size of 160 bits and window parameter $w = 4$, these equations reduce to

$$C_{mnaf} = M(2040 + 32b) \quad (10)$$

$$C_{da} = M(240b + 400) \quad (11)$$

$$C_{naf} = M(665.6 + 192b) \quad (12)$$

The costs given in 10, 11 and 12 are plotted against b by varying the window parameter w. The plot is shown in figure 1. As can be seen from the plot, the number of multiplications required by the mixed coordinate based width$-w$ NAF method of point multiplication is much less when compared to the traditional double and add algorithm as well as the width$-w$ NAF algorithm over a large range of I/M ratios. As can be seen from the graph, the mixed coordinate system based width-w NAF point multiplication algorithm achieves a drastic reduction in the number of multiplications for high values of b.

VI. Results

This section presents a discussion on the computational requirements of the mixed-coordinate system based width$-w$ NAF method and compares it with those of the traditional double and add and the width$-w$ NAF algorithms. As noted earlier, the width$-w$ NAF method of point multiplication offers superior performance when compared to the traditional double and add method. In addition, as can be seen from figure 1, the use of a mixed coordinate system in the width$-w$ NAF greatly enhances the efficiency of the point multiplication operation. As noted earlier, for software implementations of ECC, the I/M ratio b varies between 4 and 20. As can be seen from figure 1, for the case of $b = 20$, the number of field multiplications required by these three techniques and the reduction in the case of the mixed coordinate system based width$-w$ NAF algorithm is tabulated in table VII.

As can be noted from table VII, the mixed coordinate system based width-w NAF method of point multiplication has resulted in a reduction by about 48.4% in the number of field multiplications with respect to the double and add algorithm, whereas the reduction with respect to the traditional width$-w$ NAF technique of multiplication is about 40%.

TABLE VII
THEORETICAL RESULTS ($w = 4$, $b = 20$)

Point Multiplication Algorithm	Number of multiplications
Double and Add	5200
Width-w NAF	4505.6
Mixed Coordinate width$-w$ NAF	**2680**

Next, the results obtained from the C++ implementation of the mixed coordinate system based width$-w$ NAF point multiplication algorithm is discussed. The simulation environment is set up keeping in mind the specifications of MEMSIC's WSN development kit, described in section I-A. All simulations are performed using AVR Studio 5, for it provides good simulation environment customizability. The following settings are done in AVR Studio's project options to simulate testing on the WSN development kit:

- Chip: ATmega 1281
- Clock Frequency: 7370000 Hz

The purpose of this simulation exercise is to verify that the theoretical results discussed earlier are indeed true. Towards this end, we make use of the in-built **cycle counter** feature of AVR Studio 5, which gives the number of clock cycles elapsed since the start of program execution. Given the clock frequency, the execution time of the program can be easily determined from the number of clock cycles. The execution times of the following algorithms are compared:

- Double and Add
- Width-w NAF
- Mixed Coordinate System based Width-w NAF

Table VIII summarizes the results obtained.

As can be seen from Table VIII, the mixed coordinate system based width$-w$ NAF system resulted in an execution time reduction of about 48% in the case of 160$-$bit prime field and about 46% in the case of a 192$-$bit prime field with respect to the double and add algorithm. With respect to the traditional width-w NAF technique, the reduction is about 37% in the case of 160$-$bit prime field and about 32% in the case of 192$-$bit prime field. These results are in agreement with the theoretical results discussed earlier.

TABLE VIII
C++ IMPLEMENTATION RESULTS FOR $w = 4$ (ALL TIMES IN MILLISECONDS)

Point Multiplication Algorithm	$GF(160)$	$GF(192)$
Double and Add	19.75	30.87
Width-w NAF	16.08	24.45
Mixed Coordinate Width-w NAF	**10.23**	**16.67**

VII. CONCLUSIONS

In this paper, a mixed coordinate system based width-w NAF elliptic point multiplication algorithm is presented, achieving a computation cost reduction of about 48% with respect to the traditional double and add algorithm and about 40% with respect to the traditional width$-w$ NAF algorithm. To verify the accuracy of these theoretical values, timing results from the software implementations are also given, simulating the specifications of MEMSIC's wireless sensor network development kit.

REFERENCES

[1] ISO/IEC 14888-3. Information Technology - Security Techniques - Digital Signatures with Appendix - Part 3: Certificate Based-Mechanisms, 1998.

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

[3] N. Gura, A. Patel, A. Wander, H. Eberle, and S.C. Shantz. Comparing elliptic curve cryptography and rsa on 8-bit cpus. *Cryptographic Hardware and Embedded Systems-CHES 2004*, pages 925–943, 2004.

[4] C. Guyot, K. Kaveh, and V.M. Patankar. Explicit algorithm for the arithmetic on the hyperelliptic jacobians of genus 3. *JOURNAL-RAMANUJAN MATHEMATICAL SOCIETY*, 19(2):75–115, 2004.

[5] D. Hankerson, J.L. Hernandez, and A. Menezes. Software implementation of elliptic curve cryptography over binary fields. In *CHES, LNCS 1965*, pages 1–24, 2000.

[6] N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48:203–209, 1987.

[7] P. Longa and A. Miri. New multibase non-adjacent form scalar multiplication and its application to elliptic curve cryptosystems. In *âĂć*, number 052 in Cryptology e-print Archive. 2008.

[8] MEMSIC, Inc. *MPR-MIB Users Manual*, revision b edition, March 2012.

[9] V.S. Miller. Use of elliptic curves in cryptography. In *Advances in Cryptology*, volume 218, pages 417–426, 1986.

[10] M. Seysen. Using an rsa accelerator for modular inversion. *Cryptographic Hardware and Embedded Systems–CHES 2005*, pages 226–236, 2005.

[11] J. Solinas. Efficient arithmetic on koblitz curves. *Designs, Codes and Cryptography*, 19:195–249, 2000.