Improved Algorithm for Elliptic Curve Scalar Multiplication Using *w*MOF and Shamir Method

E.Karthikeyan[1], P.Balasubramaniam[2]

ABSTRACT

Since the inception of elliptic curve based cryptosystems, numerous amount of research has been done to increase the efficiency. ECC is accepted widely as a next generation cryptosystem and is more suitable for limited environments like cellular phones, PDA etc. Scalar multiplication is the most time consuming operation in elliptic curve based protocols. In this paper, we discuss the new canonical form called MOF (mutual opposite form) and propose a new method, which is an extended version of Shamir's method. It is found that our proposed method significantly improves the performance of the elliptic curve exponentiation.

KEYWORDS : scalar multiplication, elliptic curve, non-adjacent form, mutual opposite form

1. INTRODUCTION

Elliptic Curve Cryptography (ECC) was independently proposed by Miller [6] and Koblitz [2] in the year 1985 and it is gaining a wide acceptance as an alternative to the conventional cryptosystems [4] like RSA and DSA. The primary reason for the attractiveness of ECC over the conventional systems is that it offers equivalent security using far smaller key sizes. For example, RSA needs 1024-bits but ECC needs just 160-bits to offer similar level of security. These advantages are particularly beneficial in applications where bandwidth, processing capacity, power availability, or storage is constraint. Such applications include: smart cards, electronic commerce, web servers, cellular telephones, PDA and pagers (See more details in [3][5]).

Elliptic curve based protocols such as ECDH (Elliptic Curve Diffie-Hellman),

ECDSA (Elliptic Curve Digital Signature Algorithm) and ECIES (Elliptic Curve Integrated Encryption Schemes) involves scalar multiplications. The speed of scalar multiplication plays an important role in deciding the efficiency of the whole system. In particular, fast multiplication is more crucial in some environments such as central servers, where the large number of key agreements (ex. e-commerce server) or signature generations occur, and in handheld devices with low computational power.

The rest of this paper is organized as follows. In section 2, the standard scalar multiplication algorithm is explained including NAF (Non-adjacent Form) conversion. New canonical representation of binary string called MOF (Mutual Opposite Form) is discussed in section 3 and in section 4 an efficient method for exponentiation called window method is discussed. Our proposed method is illustrated with example in the section 5.

2. SCALAR MULTIPLICATION

Scalar multiplication is the calculation of the form Q= k P where P and Q are the points on the curve and k is an integer. This is simply calculated by adding P repeatedly

[1]Research Scholar, Department of Computer Science, Gandhigram Rural Institute (Deemed University)
[2]Reader, Department of Mathematics,
Gandhigram Rural Institute (Deemed University)
Gandhigram, Dindigul, Tamil Nadu, India

[1] Author is currently working as a Senior Lecturer in Computer Science at Karpagam Arts and Science College, Coimbatore, Tamil Nadu, India.

k times. i.e. P + P + P... P. The binary representation of k, i.e., $k = \sum_{j=0}^{l-1} k_j 2^j$, where $k_j \in \{1,0\}$ can be used for the computation of kP by repeated "point additions" and "point doublings" (i.e. ECADD and ECDBL) and the following is an algorithm, Algorithm-1 for the computation of kP.

```
Input : k and P
        (k is an integer and P is point)

Output : Q = kP

Begin
   Q ← O
   For j = l – 1 DownTo 0
      Q ← 2Q (ECDBL)
      If (uj = 1)
            Q ← Q + P (ECADD)
Return Q
End
```

Algorithm-1 : Binary Method

The expected number of 1's in the binary representation of k is *l/2* and the expected running time is approximately l ponit doublings and *l/2* point additions for *l*-bit integer k. The non-zero digit of the binary representation requires an extra operation than the zero entries. For example, if the bit is 1, then two operations are to be carried out (ECDBL and ECADD) and if the bit is 0, only one operation i.e. ECDBL is carried out. So the computation with many zero entries of the binary representation becomes faster exponentiation. Signed representation is an alternative representation for binary strings, which is discussed in the next section.

3. Non Adjacent Form (NAF)

Subtraction of points on elliptic curve is just as an addition operation and this leads to the signed digit representation called Non-adjacent Form (NAF). In case of NAF, integer k is converted to the signed binary representation using three digits, namely '0','1' and '-1'. Here, integer k is represented as $k = \sum_{j=0}^{l-1} k_j 2^j$, where each kj ∈ {–1, 0, 1}. So the signed binary representation of an integer minimizes the number of addition / subtraction operation and it becomes an efficient method for scalar mutiplication. NAF of an integer k can be converted by the Algorithm-2.

```
Input : k where k is an Integer
Output : NAF representation of k
c ← k ; l ← 0
While (c > 0)
  If (c is odd)
        u[l] ← 2 – (c mod 4)
        c ← c – u[l]
  Else
        u[l] ← 0
  EndIf
      c ← c/2 ; l ← l + 1
End While
Return u
```

Algorithm-2: Computation of the NAF

The revised *binary method* is called '*addition-subtraction*' method, which is described in draft standard IEEE P1363 [7]. The algorithm-3 performs addition / subtraction operation depending upon the sign of each digit, scanned from left to right.

```
Input : NAF(k) and point P
Output : Q = kP
Begin
 u[ ] ← NAF(k)        Q ← O
 For j = l – 1 DownTo 0
     Q ← 2Q
     If (uj = 1)
        Q ← Q + P
     If (uj = –1)
        Q ← Q – P
  EndFor
Return Q
```

Algorithm -3: Addition-Subtraction method

The above algorithm performs *l* doublings and *l/3* additions on an average for the *l-bit* integer k. For k = 7 $(111)_2$, the binary method would require 3 doublings and

3 additions. In case of addition-subtraction method (NAF (7) is 1 0 0 –1), it would require 4 doublings and only 2 additions. The variation can be realized for large k.

Conversion of an integer k to NAF (k) starts with the least significant bit; each bit must be shifted by single digit from right-to-left. This conversion process, however, has the drawback that involves duplication and as a result, the conversion time becomes longer for large k. This is the disadvantage of the scalar multiplication algorithm, which uses NAF of an integer.

5. Mutual Opposite Form (MOF)

Mutual Opposite Form (MOF) is a new canonical representation of signed binary strings proposed in CRYPT 2004 [1]. Recoding can be done from left-to-right or right-to-left. Since the left-to-right conversion scheme saves the time and memory, MOF is preferable on constraint devices. A property of MOF is that signs of adjacent non-zero bits (without considering 0 bits) are opposite, and most non-zero bit and the least non-zero bit are 1 and -1, respectively.

5.1 MOF Conversion

Let us see the conversion from n-bit binary string to (n+1)-bit MOF. The n-bit binary string d can be converted to a signed binary string by computing md=2d-d, where '-' stands for a bitwise subtraction and the algorithm is given below. Finally, MOF is converted according to the property i.e. (1,-1)=> (0,1) and (-1,1)=> (0,-1).

$2d = d_{n-1}$	d_{n-2}	...	d_{i-1}	...	d_1	d_0		
$-d =$	d_{n-1}	...	d_i	...	d_2	d_1	d_0	
$md =$	d_{n-1}	$d_{n-2}-d_{n-1}$	...	$d_{i-1}-d_i$	...	d_1-d_2	d_0-d_1	$-d_0$

Input : n-bit binary string $d=d_{n-1}|d_{n-2}|..|d_1|d_0$
Output : MOF $md_n|...|md_1|md_0$ of d

$md_n = d_{n-1}$
for i = n - 1 down to 1 do
    $md_i = d_{i-1} - d_i$
$md_0 = -d_0$
return $(md_n, md_{n-1}, ..., md_1, md_0)$

Algorithm-4 : Left-to-Right Generation from Binary to MOF

The output of the algorithm-4 is the MOF(d) and it can be used in '*addition-subtraction*' method. For example, binary equivalent of an integer 2393 is '1 0 1 1 0 1 1 0 1 1 1 1' and the MOF of 2393 is '0 1 1 0 0 –1 0 0 –1 0 0 0 –1'. So it is found that the MOF of any integer is having less hamming weight than the binary string of the same.

5.2. Window Method

The number of non-zero entries can be further minimized by window method, which is the most common method for computing exponentiation of random elements in Abelian groups. This approach enhances the efficiency of the binary method at the expense of some pre-computations. It significantly reduces the number of point additions required, but the number of point doublings remains essentially the same. The following is an algorithm for conversion of an integer to wMOF of the same.

Input : width *w*, *n*-bit binary string
$d=d_{n-1}|d_{n-2}|...|d_1|d_0$
Output : wMOF $sd_n|sd_{n-1}|...|sd_0$ of *d*

$d_{-1} <- 0$; $d_n <- 0$
i <- 0
while i >= w - 1 do
  if $d_i = d_{i-1}$ then
      $sd_i <- 0$; i <- i - 1
  else
    $(sd_i, sd_{i-1}, ..., sd_{i-w+1}) <- Table_{wSW}(d_{i-1} - d_i, d_{i-2} - d_{i-1}, ..., d_{i-w} - d_{i-w+1})$
  i <- i - w
  if i >= 0 then
    $(sd_i, sd_{i-1}, ..., sd_0) <- Table_{i+1SW}(d_{i-1} - d_i, d_{i-2} - d_{i-1}, ..., d_0 - d_i, -d_0)$

return $(sd_n, sd_{n-1}, ..., sd_0)$

Algorithm-4: Left-to-Right Generation from Binary to MOF

The average density of non-zero bits is asymptotically 1/(w+1) for n->∞, and the digit set equals T={±1, ±3,..., ±

$(2^{w-1}-1)\}$ which seems to be minimal. The conversion of an integer to MOF(d) illustrated with the following example. Integer k=345; width w = 3. Some predefined replacements are: 1-11, 10-1 => 003, -11-1, -101 =>00-3, 1-10 => 010, -110 => 0-10.

1 0 1 0 1 1 0 0 1 (Binary)
1 -1 1 -1 1 0 -1 0 1 -1 (MOF)
0 0 3 -1 1 0 -1 0 1 -1
0 0 3 0 -1 0 -1 0 1 -1
0 0 3 0 -1 0 0 0 -3 -1 (wMOF).

The number of zero entries and non-zero entries can be compared from the above example and found that wMOF has less number of non-zero entries than other representation. So that window method is the best method for the computation of kP with some pre-computation.

6. PROPOSED METHOD

There are vast amount of researches being carried out for speeding up the scalar multiplication. Even a small improvement is also welcome, because ECC is most suitable for the constraint devices. In some elliptic curve based protocols we need to compute kP and, some cases like signature verification requires the computation of aP+bQ. Shamir proposed a simple method [8] to compute aP+bQ simultaneously and he considers NAF of binary string for the calculation ie NAF(a) and NAF(b). Solinas [8] also proposed a new method of computation of aP + bQ (More details of various representation can be found in [10]). In this paper we extend Shamir's method using wMOF of binary representation for doing parallel computation and it is illustrated in the Table-1. Let us consider that a=687 and b=729, binary equivalent is 1 0 1 0 1 0 1 1 1 1 and 1 0 1 1 0 1 1 0 0 1 and wMOF equivalent of a and b is 3 0 0 –3 1 0 0 0 -1 and 3 0 0 –1 0 –1 0 0 1.

a=687	3	0	0	-3	1	0	0	0	-1
b=729	3	0	0	-1	0	-1	0	0	1
Double	0	6P 6Q	12P 12Q	24P 24Q	42P 46Q	86P 92Q	172P 182Q	344P 364Q	688P 728Q
+3P	3P								
-3P				21 P					
+3Q	3Q								
-3Q									
+P					43P				
-P						91Q			687P
+Q									729Q
-Q				23 Q					

Table-1: Example for the Proposed method

From the Table-1, it is found that the number of additions required depends on the joint weight of a and b and the number of doubling required is one less than the number of bits in wMOF(a) or wMOF(b). Thus minimizing the joint weight could speed up the computation. If we use the conventional method for exponentiation of 687P + 729Q independently, it requires 18 ECDBL and 11 ECADD operation and one more addition in the final. But if we use our proposed method to perform the same, it requires 8 ECDBL and 6 ECADD operations only. Computation of the row 'Double' is done simultaneously by "Multi Threading" concept of JAVA. The proposed method is tested on Intel Pentium IV machine using J2EE, JCE (Java Cryptographic Extensions) and the result is given in the Table –2.

Coordinate	Algorithm	Time (ms)
Affine coordinate	Binary Method [2]	0.63
	Signed Binary Method [7]	0.46
	Proposed Method	0.31
Projective Coordinate	Binary Method	0.52
	Signed Binary Method	0.39
	Proposed Method	0.30

Table–2: Experimental Result of the Proposed Method

From the result obtained by our proposed method, we found that our method significantly improves the speed

of the scalar multiplication. There are many other parameters that could change the efficiency of the algorithm. The coordinate representation will change the efficiency of the algorithm little more. Affine coordinate requires a division in every addition and doubling operation but require fewer multiplications than the projective coordinate. On the other hand, projective coordinate does not require any division in either addition or doubling but does require a division only once in the final stage of the computation. Another coordinate system called Jacobian coordinate system offer a slower addition but a faster doubling [9]. So changing various elliptic curve parameters may improve the performance of the algorithm.

7. Conclusions & Future Work

Since the inception of ECC, vast amounts of researches are being carried out to increase the efficiency, because it is accepted widely as next generation cryptosystem very much suitable for the constraint devices. Scalar multiplication is the most time consuming operation in all elliptic curve based protocols. In this paper we discussed the new canonical representation called MOF and proposed a new method, which is an extended version of Shamir's method. It is found that our proposed method significantly improves the speed of scalar multiplication.

In the near future almost all the areas of ECC can further be enhanced, because of its attractiveness and even a small amount of development is also most welcome.

Acknowledgement

I acknowledge the Management and Faculty Members of Department of Computer Science, Karpagam Arts and Science College for their invaluable support to carryout my research work.

Reference

[1]. Katsuyuki Okeya, "Signed Binary Representations Revisited", CRYPTO 2004, pp.123-139, 2004

[2]. N Koblitz, "Elliptic Curve Cryptosystem", Mathematics of Computation, No.48, pp.203-209, 1987

[3]. J Lopez and R Dahab, "An overview of elliptic curve cryptography", Technical Report, Institute of Computing, State University of Compinas, Brazil, 2000

[4]. A J Menezes, P C van Oorschot, and S A Vanstone, "Handbook of Applied Cryptography", CRC Press, 1997.

[5]. A J Menezes and S A Vanstone,"Elliptic curve cryptosystems and their implementations", Journal of Cryptology, Vol.6, No. 4, pp.209-224, 1993.

[6]. V S Miller, "Use of Elliptic Curves in Cryptography" Advances in Cryptology–Proceedings of CRYPTO'85, LNCS-218, pp. 417-426, 1986

[7]. IEEE P1363, Standard Specifications for Public-Key Cryptography, 2000

[8]. J A Solinas, "Low-Weight Binary Representations for Pairs of Integers", Technical Report CORR 2001-41, Center for Applied Cryptographic Research, University of Waterloo, Canada, 2001.

[9]. H Cohen, A Miyaji, and T Ono, "Efficient elliptic curve exponentiation using mixed coordinates" ASIACRYPT: Advances in Cryptology, LNCS, vol. 1514, pp. 51-65, 1998.

[10].J A Muir, "Efficient Integer Representations for Cryptographic Operations", Ph.D. Theses, Center for Applied Cryptographic Research, University of Waterloo, 2004

Authors' Biography :

E.Karthikeyan completed his M.Sc (CS), M.Phil (CS) and doing Ph.D. at Gandhigram Rural Institute-Deemed University, Dindigul. He is also working as a senior lecturer in Computer Science at Karpagam Arts and Science College, Coimbatore. He has published more than 10 papers in the conferences. His area of interest is Network Security, Elliptic curve Cryptography

P.Balasubramaniam obtained his M.Sc, M.Phil and Ph.D. in mathematics. He is currently working as a Reader, Department of Mathematics, Gandhigram Rurual Institute-Deemed University, Dindigul. He has completed several research projects from various funding agencies. He has published more than 40 papers in the National / International Journals. His area of research includes Controls, Fuzzy logic, Cryptography.