Fuzzy Controlled Scalar Multiplication for ECC

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

***Abstract*—Traditionally, RSA is being used for authentication and key exchange for symmetric key cryptography (SKC). Improved network security demands forward secrecy also. Even though, RSA, a widely used key exchange approach can not provide forward secrecy, the same can be achieved by making used Elliptic Curve Diffie-Hellman Ephemeral (ECDHE) technique for SKC and RSA for the purpose of authentication. However, ECDHE_RSA based approach is more compute intensive compared to the RSA alone. The predominant operation in the ECDHE technique is Elliptic Curve (EC) based scalar multiplication. Hence, speeding up of ECDHE operation demands faster EC scalar multiplication algorithm. Binary method, Non-adjacent form (NAF) method and sliding window method are used to carry out the EC scalar point multiplication. An algorithm based on both the NAF and the sliding window techniques is considered. This technique is more efficient in terms of EC point operations. There is a trade-off between the number of EC point addition operations and the number of pre-computed values. A fuzzy based controller method is proposed to determine an optimum window width, resulting in faster scalar multiplication.**

Keywords- forward secrecy, ECC, EC-DHE, Fuzzy control.

I. INTRODUCTION

The Transport Layer Security (TLS) protocol makes use of one of the two key exchange mechanisms: RSA and Diffie-Hellman (DH). The RSA is primarily used for exchanging keys to be used for SKC based communication as the DH based key exchange is more expensive. In case of any breach in the RSA based security model, SKC keys can be extracted thereby resulting in hijack and data capture. RSA based key exchange uses the same pair of keys for many sessions, whereas DH based key exchange uses different pairs of keys for multiple sessions. Even if there occurs a single session compromise, the data capture is constrained to that session alone. This is called forward secrecy and RSA does not provide the same [1]. It means that the information, which is secure at present, will also remain secure in near future. The forward secrecy strength depends on the DH key pairs [2]. The problem of computational complexity of DH_RSA approach can be overcome by making use of Elliptic Curve (EC) based DH. Table I shows that the same standard of security with reduced number of bits is pursued by the Elliptic Curve Cryptography (ECC) compared to DH. In ECDH Ephemeral (ECDHE) technique, as shown in Fig. 1, the generated key pair is used for a single session, usually lasting for a short duration.

DH	1024	2048	3072	7689
ECC	163	233	283	409

TABLE I
COMPARABLE KEY SIZES [3]

A standard elliptic curve E, specifically for purpose of cryptography over the prime field (F_P) is given as:

$$y^2 \ mod \ p = (x^3 + ax + b) \ mod \ p, \qquad (1)$$

where a,b $\in F_P$ and $(4a^3 + 27b^2) \ mod \ p \neq 0$ [4]. The points on E, is calculate with equation (1). Addition of two points (Point Addition) and doubling of a point (Point Doubling) are the basic operation performed over EC, as given in Table II. The rest of the paper is organized as follows: Section II discusses various scalar multiplication methods, Section III presents the proposed scheme and the fuzzy controller and Section IV compares various scalar multiplication techniques.

EC Operation	Slope (S)	x_3	y_3
Point Addition $P(x_1, y_1) + Q(x_2, y_2)$	$\frac{y_2 - y_1}{x_2 - x_1}$	$S^2 - x_1 - x_2$	$S(x_1 - x_3) - y_1$
Point Doubling $2P(x_1, y_1)$	$\frac{3x_1^2 + a}{2y_1}$	$S^2 - 2x_1$	$S(x_1 - x_3) - y_1$

TABLE II
EC MATHEMATICAL OPERATION [4]

II. SCALAR MULTIPLICATION

The classical Elliptic Curve Diffie Hellman ephemeral (ECDHE) scheme is illustrated by the Fig. 1.

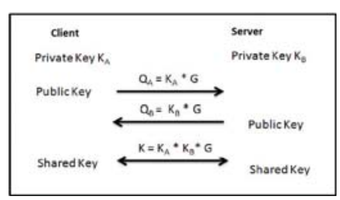

Fig. 1. Elliptic Curve Diffie-Hellman [2]

Initially, both the server and the client nodes agree on a particular Elliptic curve (EC) in the prime field F_P, with a specific base point termed the generator point, G. The G is one of the valid point on the EC curve which has highest order [4]. Both the server and client nodes set their respective private keys by selecting randomly any scalar integer in the

978-1-4673-6349-5/13/$31.00 ©2013 IEEE

prime field, F_P. The corresponding public keys, Q_A and Q_B, are computed by multiplying the generator point, G, with the corresponding private keys namely K_A and K_B. This public keys are then shared over the network between the server and the client, which again multiply them with the corresponding private key, hence generating a shared secret key given as $T = K_A * Q_B = K_B * Q_A$. Due to Elliptic Curve Discrete Logarithmic Problem (ECDLP), even though the value of Q_A and Q_B and G are spread over the network, it would be computationally infeasible to calculate the private keys K_A and K_B for an intruder [5].

The two frequently used operations in ECDHE key exchange are: scalar multiplication and modular reduction. Scalar multiplication based on ECDLP, consumes 85% of computational cost in ECC [6]. It consist of multiplying a point on the E, with a scalar integer k, such that $k \in F_P$.. Scalar Multiplication consist of point addition and point doubling operations. Table III shows the number of prime field operation required by the point addition and the point doubling over EC.

TABLE III
EQUIVALENT PRIME FIELD OPERATION

	Inversion	Multiplication
Point Addition ($P \neq Q$)	1	3
Point Doubling ($P = Q$)	1	4

Thus the speed of ECDHE key exchange method is directly proportional on the performance of the scalar multiplication on the EC. This can be achieved by adopting techniques for recoding of the scalar integer k.

Recoding of Scalar Integer k: The recoding of scalar integer k attempts to reduce the length and the number of $1'$s in the binary form of k, as the number of point addition operations depends the number of $1'$s in k and number of point doubling operations depends on the length of k, thereby speeding up the scalar multiplication operation.

A. Binary Method

Binary method is the simplest and the most computationally expensive scalar multiplication method [7]. Binary representation of the integer k helps use to conclude that, consecutive summation of the point doubling and point addition operation over the EC leads to scalar multiplication.

$$k = \sum_{j=0}^{l-1} K_j 2^j, \quad K_j \in \{0, 1\} \tag{2}$$

$$Q = kG = K_0 G + 2(K_1 G + ... + 2(K_{l-1} G)))) \tag{3}$$

In scalar multiplication, point addition (A) and point doubling (D) operation are used to determine computational cost of different algorithms. The number of $1'$s in the binary representation of k is called its Hamming weight (W) and l is the total number of bits in k. The computational cost of the Binary method is given by the equation (4).

$$Cost = (W - 1)A \; + \; (l - 1)D \tag{4}$$

B. NAF Method

Contrary to the representation of k in Binary method, if the representation of k also consist of negative bits, i.e. {-1,0,1}, then it is called as Binary Signed Digit Representation (SDR). In Non-adjacent form (NAF), both W and l are kept as small as possible. A NAF of a positive integer, k, is given by the equation (5) [7].

$$k = \sum_{j=0}^{l-1} K_j 2^j, \quad K_j \in \{-1, 0, 1\}, \tag{5}$$

such that, multiplication of any two consecutive bits is always zero i.e. $K_j * K_{j+1} = 0$. The NAF form of integer k is denoted as NAF(k), its length is at most $(l+1)$ of the binary form of k. Algorithm 1 is used to convert the integer k into its NAF(k).

Algorithm 1 Binary(k) to NAF(k) [8]

Input: Scalar k shown in equation (2)
Output:NAF(k)

$E_0 \leftarrow 0$
for $i = 0 \; to \; (l-1)$ **do**
  $E_{(i+1)} \leftarrow \lfloor (K_i + E_i + K_{(i+1)})/2 \rfloor$
  $S_i \leftarrow K_i + E_i - 2E_{(i+1)}$
end for
Return(S_l.........S_0)

Scalar multiplication for NAF(k) is obtained using the equation (3), only difference is that when -1 appears G should be subtracted from Q. The W of the positive integer k can be reduced to $(l/3)$ by using NAF(k) and the number of point doubling operations remains to be the same as in the binary method [7]. Therefore, the computational cost of the scalar multiplication using NAF(k) is given by the equation (6).

$$Cost = \frac{l}{3} A \; + \; lD \tag{6}$$

Table IV illustrates different examples of NAF (k).

TABLE IV
NAF FORM OF INTEGER

Decimal Representation	Binary Representation	NAF Representation
26	11010	$10\bar{1}010$
1122334455	1000 0101 1100 1010 1110 1101 1110 111	1000 $10\bar{1}0$ $0\bar{1}01$ $0\bar{1}0\bar{1}$ 000 $\bar{1}$ $00\bar{1}0$ $000\bar{1}$ 001

C. Sliding window Method

To reduce the computational cost of Binary and NAF methods, the digits used for representing k can be extended beyond 3 bits as in NAF, $\{-1, 0, 1\}$. This reduces the number of point additions. But this advantage comes at the cost, little amount of values that are multiple of G should be pre-computed and stored in memory, such that they are added or subtracted to the Q [7] during multiplication. The memory required to hold pre-computed values becomes a constraint.

The sliding window method processes at most consecutive w digits of the scalar integer k such that the decimal equivalent of the window-w consecutive digit should be odd. This method has no fixed window width, the same can be varied from 1 to w and 0 bit is ignored.

Algorithm 2 presents the scalar multiplication for the sliding window method with binary representation of integer k. Table V provides the details for the different window widths (w). The computational cost for the binary sliding window method is shown in Table VI, where $V(w)$ as given in the equation (7), is the average length of a run of $0's$ within the window [4].

Algorithm 2 Binary Sliding window for scalar multiplication [5]

Input : Generator point G, k, window width-w
Output:$Q = kG$

1: Calculate [x]G where $x = 1, 3, 5...., (2^{(w-1)} - 1)$
2: $j \leftarrow l - 1$, where l is length of k
3: **while** $j \geq 0$ **do**
4: if $(K_j == 0)$
5: $Q \leftarrow [2]Q$,$N \leftarrow 0$, $j \leftarrow j - 1$
6: end if
7: else
8: $i \leftarrow maximum(j - w + 1, 0)$
9: **while** $K_i == 0$ **do**
10: $i \leftarrow i + 1$
11: **end while**
12: **for** $d = 1 \quad to \quad (j - i + 1)$ **do**
13: $d = d + 1 \; and \; Q \leftarrow [2]Q$
14: **end for**
15: $N \leftarrow (K_j.......K_i)_2$
16: $j \leftarrow i - 1$
17: end else
18: $Q \leftarrow Q \oplus [N]G$
19: **end while**
20: Return Q

$$V(w) = \frac{4}{3} - \frac{(-1)^w}{3 * 2^{w-2}} \qquad (7)$$

III. Proposed Scheme

A. NAF sliding window Method

Algorithm 3 uses both sliding window method and NAF(k). The NAF(k) is computed and the same is given as input to this algorithm.

The combination of sliding window and NAF methods, reduces the number of pre-computations required compared to the combination of Binary method and sliding window methods. This improves the efficiency of the algorithm, in a system with limited memory. The computational cost for the NAF and sliding window method is given in Table VI. The computational cost of the Algorithm 2 and 3 depends upon the window width, w. An optimal window width, w, needs to be chosen before hand in order to reduce the computational cost.

Algorithm 3 NAF Sliding window for Scalar Multiplication [4]

Input: Generator Point G, integer k, window width-w
Output: $Q = kG$

1: Compute NAF(k) with Algorithm 1.
2: Calculate [x]G where $x = (1, 3, 5, ..., ((2^w - (-1)^w)/3 - 1))$
3: $j \leftarrow l - 1$ where l is the length of k
4: **while** $j \geq 0$ **do**
5: Algorithm (2) Steps 4 to 17
6: if $(N \geq 0)$
7: $Q \leftarrow Q + [N]G$, end if
8: else $Q \leftarrow Q - [N]G$, end else
9: **end while**
10: **return** (Q)

Method [4]	Number of Doublings(D)	Number of Additions(A)	Number of Pre-computations
Binary	l	$\frac{l}{w+v(w)}$	1D+$(2^{w-1} - 1)$A
NAF	l	$\frac{l}{w+v(w)}$	1D+$(2\frac{2^w - (-1)^w}{3} - 1)$A

TABLE VI
Computational cost for Sliding window Scalar Multiplication

B. Fuzzy controller

From the Table VI, it can be observed that the computational cost of sliding window method depends on the window width-w. The optimum selection of the window width-w leads to reduced number of arithmetic operations in point multiplication of ECC. This motivates the need of the controller, which can select optimum window width-w automatically. To achieve this, a controller based on fuzzy logic as shown in Fig. 2, is used. This approach proves to be a more efficient and computationally in-expensive. A fuzzy system dealing with uncertainties, is a set of fuzzy rules which converts input to output [10]. Fuzzy sets are defined by their vague and ambiguous properties, on the contrary to crisp sets. Fuzzy system helps to build inference system, which converts the human vague reasoning logic to an artificial knowledge based system.

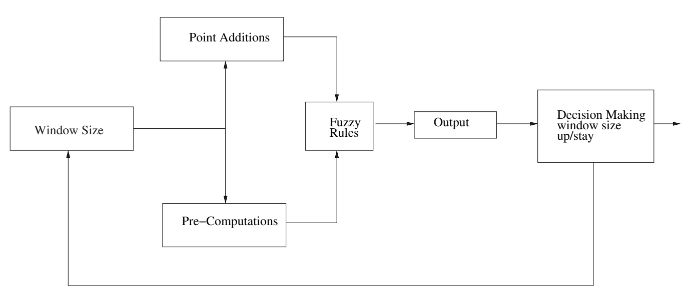

Fig. 2. Window Width Controller [9]

The block diagram of fuzzy controller for optimum win-

TABLE V
DIFFERENT WINDOW WIDTH COMPARISON IN SLIDING WINDOW METHOD

Window width-w	Number of Pre-computations	Integer k=2973	Intermediate values	Number of Additions	Number of Doublings	Pre-computations
3	3	101 11 00 111 01	5G, 10G, 20G, 23G, 46G, 92G, 184G, 368G, 736G, 743G, 1486G, 2972G, 2973G.	3	9	[3]G, [5]G, [7]G
5	15	10111 00111 01	23G, 46G, 92G, 184G, 368G, 736G, 743G, 1486G, 2972G, 2973G.	2	7	[3]G, [5]G, [7]G [9]G.......... [25]G [27]G, [29]G, [31]G

TABLE VII
RULES FOR FUZZY WINDOW CONTROLLER

Number of Point Additions	Number of Pre-computations	Window Width-w
Low	Low	Up
Low	Average	Stay
Low	High	Stay
Average	Low	Up
Average	Average	Up
Average	High	Stay
High	Low	Up
High	Average	Up
High	High	Stay

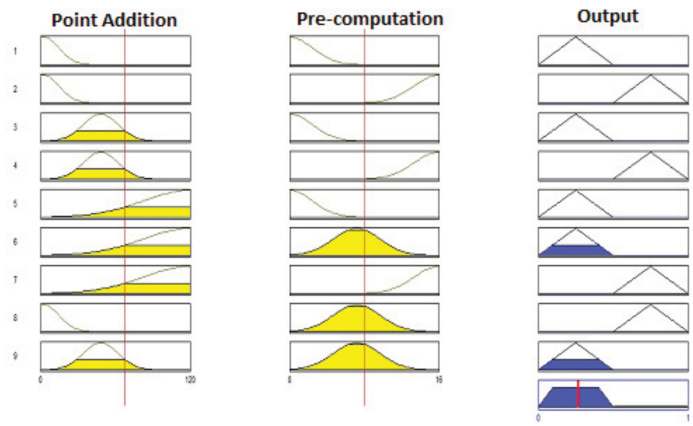

Fig. 4. Simulation of Fuzzy Rules

dow width selection as given in Fig. 2 comprises of two inputs, namely, number of point additions and number of pre-computations [9]. As only a slight change occurs in the number of point doublings for different window sizes, the same is considered constant in this fuzzy system. Here, the memory storing pre-computations is considered constant. Accordingly, the membership for the two input system is considered. Both these inputs have three statuses 1)Low 2)Average 3)High and their respective Gaussian membership functions are defined. One output, window width-w is defined for the fuzzy controller which has two statuses, namely as 1) Up and 2) Stay and the triangular membership functions are used for defining the window width-w.

Two models of fuzzy inference systems, namely Mamdani model and Takagi-Sugeno model are mostly used for building of the fuzzy system. Mamdani model deals with the fuzzy set as rules and consequent. Takagi-Sugeno deals with linear function of the input variable [10]. The fuzzy controller developed here, Fig. 3 uses the Mamdani model. The rules defined for this system [9] are given as in Table VII:

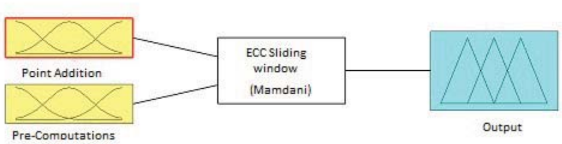

Fig. 3. The Current Fuzzy Controller

The simulation of Fuzzy rule for the fuzzy controller is shown in Fig. 4. Thus with the help of the fuzzy rules, the controller as shown in Fig 2 attempts to find out the optimum selection of the window width-w.

IV. COMPARISON

In this section, a comparison of different scalar multiplication methods is presented. The comparison as given in the Table VIII, considers number of EC mathematical operations: point addition and point doubling. The sliding window and the NAF sliding window methods are also compared along with the number of pre-computations required for different window width-w. The standard Elliptic curve, secp160r1 of 160-bit is considered, with the follwing domain parameter values:

$p = 2^{160} - 2^{31} - 1$,
$a = (D6031998D1B3BBFEBF59CC9BBFF9AEE1)_{16}$,
$b = (5EEEFCA380D02919DC2C6558BB6D8A5D)_{16}$

A 160-bit k scalar, is also considered and the same is used to carry out the comparison of all the four methods for scalar multiplication.

The following scalar integer, k of size, $160-$ bits is considered. k =(BBBB BBBB BBBB BBBB BBBB BBBB BBBB BBBB BBBB BBBB)$_{16}$.

In Table VIII Binary sliding window and NAF sliding window methods have fixed window width-w of 5, however, the window size can be varied. Table IX and X provide the number of point doublings and point additions required for diferrent window widths. The number of pre-computations, which also depends on the window size, is also used for this comparison.

Method	Number of Point Doublings	Number of Point Additions
Binary Method	159	119
NAF Method	159	42
Binary Sliding Window(w=5)	158	29
NAF Sliding Window(w=5)	158	20

TABLE VIII
DIFFERENT METHODS VS NO OF ADDITIONS AND DOUBLINGS

Window size	Number Of Point Doublings	Number Of Point Additions	Pre-computations
3	157	40	3
4	156	39	7
5	155	29	15
6	155	26	31
7	153	20	63
8	152	19	127
9	151	17	255
10	151	15	511

TABLE IX
SLIDING WINDOW METHOD

From Tables IX and X, it can be noticed that for different window width-w's, the number of pre-computations and the number of point additions change significantly. Hence, an optimum selection of the window width-w, enables to achieve reduced computational cost for the scalar multiplication methods. The same can be achieved using the Fuzzy controller as shown in Fig.2 and the surface graph for the same controller is given in Fig.5.

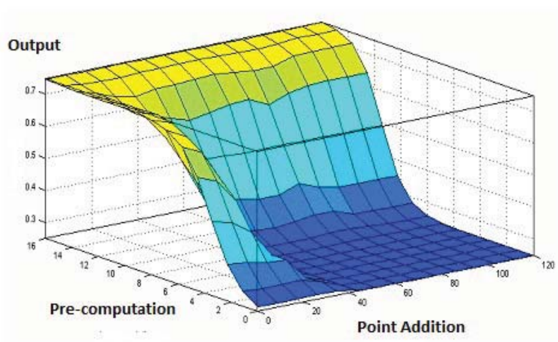

Fig. 5. Surface graph of the Fuzzy controller

V. CONCLUSIONS

In this work, different methods of EC scalar multiplication, namely Binary, NAF, Binary Sliding window and NAF sliding window, are compared. It is observed that the NAF sliding window method for an optimum window width-w, outperforms the remaining methods for scalar multiplication. This NAF sliding window method uses least number of point additions and some pre-computed values of G, which are far less than the pre-computed values required for binary sliding window method. The controller based on the fuzzy logic is used during optimum selection of the window width-w.

Window size	Number Of Point Doublings	Number Of Point Additions	Pre-computations
3	158	41	2
4	158	39	4
5	158	39	10
6	154	20	20
7	154	19	42
8	154	19	84
9	154	13	170
10	150	13	340

TABLE X
NAF SLIDING WINDOW METHOD

Thus, using fuzzy controller along with NAF sliding window method for scalar multiplication in key exchange mechanisms of ECDHE_RSA reduces the computational cost considerably.

REFERENCES

[1] B. Vincent, "Ssl/tls and perfect forward secrecy," 2011.
[2] E. Käsper, "Fast elliptic curve cryptography in openssl," *Financial Cryptography and Data Security*, pp. 27–39, 2012.
[3] S. Blake-Wilson, B. Moeller, V. Gupta, C. Hawk, and N. Bolyard, "Elliptic curve cryptography (ecc) cipher suites for transport layer security (tls)," 2006.
[4] D. Hankerson, A. Menezes, and S. Vanstone, *Guide to elliptic curve cryptography*. Springer, 2004.
[5] X. Huang, D. Sharma, and H. Cui, "Fuzzy controlling window for elliptic curve cryptography in wireless sensor networks," in *Information Networking (ICOIN), 2012 International Conference on*. IEEE, 2012, pp. 312–317.
[6] N. Gura, A. Patel, A. Wander, H. Eberle, and S. Shantz, "Comparing elliptic curve cryptography and rsa on 8-bit cpus," *Cryptographic Hardware and Embedded Systems-CHES 2004*, pp. 925–943, 2004.
[7] O. YAYLA, "Scalar multiplication on elliptic curves," Ph.D. dissertation, Master?s Thesis, Department of Cryptography, Middle East Technical University, August 2006. Available at http://www3. iam. metu. edu. tr/iam/images/3/3e/O% C4% 9Fuzyaylathesis. pdf(link tested 01-Dec-2011), 2006.
[8] X. Huang and D. Sharma, "Fuzzy controlling window for elliptic curve cryptography in wireless networks," in *Computer Sciences and Convergence Information Technology (ICCIT), 2010 5th International Conference on*. IEEE, 2010, pp. 521–526.
[9] X. Huang, D. Sharma, and P. Shah, "Efficiently fuzzy controlling with dynamic window in elliptic curve cryptography sensor networks," in *Proceeding of the International MultiConference of Engineers and Computer Scientists*, vol. 1, 2011.
[10] S. Sivanandam, S. Sumathi, and S. Deepa, *Introduction to fuzzy logic using MATLAB*. Springer, 2006.