ELLIPTIC CURVE CRYPTOGRAPHY IN SENSOR NETWORKS WITH HIDDEN GENERATOR POINT FOR SPEEDUP SCALAR MULTIPLICATION

Brajesh Patel*

Neha Jha**

Abstract:

In Network Security, Cryptography mechanism is used to secure communication with the help of various kind of algorithms provides applicability and suitability for constraint environment such as mobile sensor applications. Here both computing resources and power availability are limiting factors. Thus, it is expecting to have cryptographic algorithms for low power consuming, less computing, and highly secure for mobile sensor networks. Elliptic curve cryptography (ECC) is an emerging favorite because requires less computational power, communication bandwidth, and memory when compared to other cryptosystems. In this paper we present our new design, a hidden generator point, which offers an improvement in protection from the man-in- middle attack which is a major vulnerability for the sensor networks. Even though there are other ways to implement hidden generator point.

Keywords- Elliptic Curve Cryptographic(ECC); public key; hidden generator point; man-in-middle attack, fast scalar multiplication

* Deptt of Computer Science, HoD (Com. Sci.), SRIT, Jabalpur(m.p) India.

** Deptt of Computer Science, M.Tech (4th SEM),CTA, SRIT, Jabalpur(m.p) India.

I INTRODUCTION:

Cryptography, then, not only protects data from theft or alteration, but can also be used for user authentication. There are, in general, three types of cryptographic schemes typically used to accomplish these goals: secret key (or symmetric) cryptography, public-key (or asymmetric) cryptography. A **wireless sensor network (WSN)** consists of spatially distributed autonomous sensors to *monitor* physical or environmental conditions, such as temperature, sound, vibration, pressure, motion or pollutants and to cooperatively pass their data through the network to a main location.. Wireless network (WN) has been experiencing an explosive growth in recent years and offered attractive flexibility to network operators and users..Elliptic curve cryptography (ECC) has been widely investigated for public-key cryptography purposes [1,2]. It was introduced by Koblitz and Miller in 1980s and has attracted increasing attention in recent years due to its linear scalability, a small software footprint, low hardware implementation cost, low bandwidth requirement, and high device performance. Normally the structure of an ECC operation involves three computational levels, namely scalar multiplication algorithm, point arithmetic and field arithmetic [3]. The main focus has been on improvements at the point arithmetic level to decrease the time of ECC scalar multiplication. For point adding, a combination of projective and affine coordinates, i.e. mixed addition [4], has offered an efficient formula.

In the case of adding points in the same coordinate system, the required formula proves more costly and is realized as general addition. Recently, some approaches have been developed to compute faster scalar multiplications, such as double-base chain [5], and ternary/binary method [6] which have introduced tripling as a new point operation. Several papers discuss particular implementations over different situations [7-14]. Some of them have raised different ways to investigate the EEC such as multi-base system in ECC [15] and key exchange protocol in elliptic curve cryptography with no public point [8] that is complex and high time costs. In this paper, a new algorithm based on ones complement for fast scalar multiplication is first introduced, with which 12.5% time is saved in comparison with the results of other complementary method based algorithms [17].In real life, the wireless mobile and sensor networks are vulnerable to the so-called man-in-middle (MinM) attacks. Section 4 proposes the new algorithm based on one's complement for fast scalar multiplication. The paper concludes with a summary of work completed and results achieved.

II. TRADITIONAL PROTOCOL IN ELLIPTIC CRYPTOGRAPHY:

A special addition operation is defined over elliptic curves and this with the inclusion of a point *O*, called point at infinity. If three points are on a line intersecting an elliptic curve, then their sum is equal to this point at infinity *O*, which acts as the identity element for this addition operation. Sometimes the general equation (1) can be referred as Weierstrass equation as shown in (2):

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

If we wanted use a elliptic curve to be used for cryptography the necessary condition is the curve is not singular, i.e. the discriminant Of polynomial: $f(x) = x^3 + ax + b$:

$$4a^3 + 27b^2 \neq 0 \quad (3)$$

Figures 1 and 2 show the two elliptic curves are

$$y^2 = x^3 + 2x + 5 \quad (4) \text{ and}$$

$$y^2 = x^3 - 2x + 1 \quad (5)$$

We can see those two equations meet

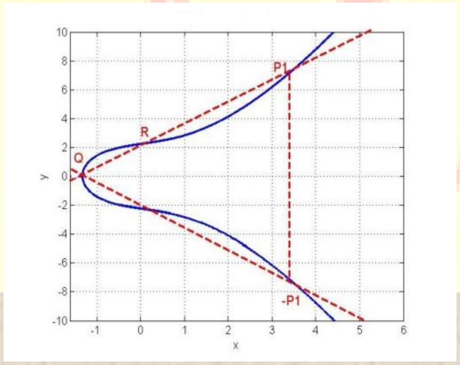

Figure 1. Elliptic curves equation (4)

An elliptic curve is the set of solutions of an equation of the form can be shown as below:

$$y^2 + axy + by = x^3 + cx^2 + dx + e \quad (1)$$

Where *a*, *b*, *c*, *d*, and *e*, are real numbers.

An elliptic group over the Galois Field $E_p(a,b)$ is obtained by computing $x^3 + ax + b \bmod p$ for $0 \le x < p$. The constants *a* and *b* are non negative integers smaller than the prime number *p* and as here we used "mod *p*", so equation (3) should be read as:

$$4a^3+27b^2 \bmod p \neq 0$$

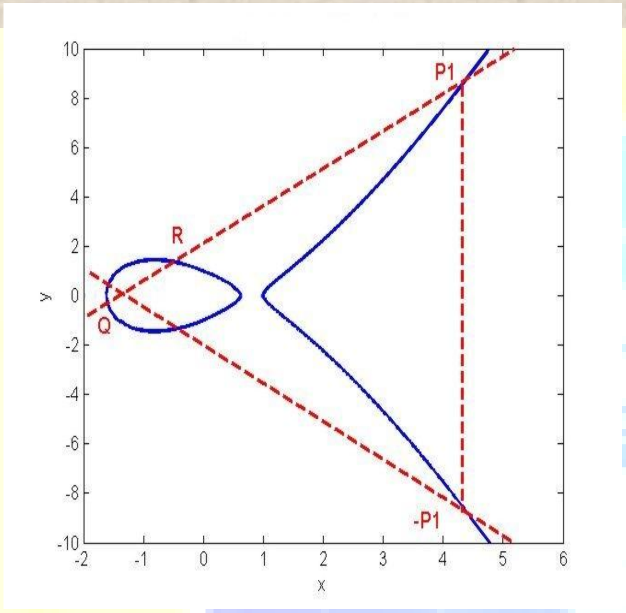

Figure 2. Elliptic curve equation (5)

For each value of *x* one needs to determine whether or not it is a quadratic residue. If it is the case, then there are two values in the elliptic group. If not, then the point is not in the elliptic $E_p(a,b)$ group. When we fixed a prime number, *p* and then we can have the Galois Field E_p (*a*, *b*) group via the fixed constants *a and b* following the above conditions.

For example, let the points $P = (x1, y1)$ and Q $(x2, y2)$ be in the elliptic group $E_p(a,b)$ group and *O* be the point at infinity. The rules for addition over the elliptic group $E_p(a,b)$ are :

(1) $P+O = O + P = P$

(2) If $x2 = x1$ and $y2 = -y1$, that is $P(x1, y1)$ and $Q = (x2, y2) = (x1 \text{-} y1) = -P$, that is the case:

$P+Q = O$.

(3) If $Q \neq -P$, then their sum $P + Q = (x_3, y_3)$ is given by;

$x_3 = \lambda \quad - x_1 - x_2 \mod p$

$y_3 = \lambda(x_1 - x_3) - y_1 \mod p$ (7)

In order to express our new protocol of the hidden generator point we, without losing generality, use an example for the above description. Let's assume $p = 23$ and $a = 1$ and $b = 1$, i.e. the equation becomes: $y^2 = x^3 + x + 1 \mod 23$. We have $4a^3 + 27b^2 \mod 23 = 8 \neq 0$.

Now we need to determine if y^2 is in the set of quadratic residues or not. The calculation results are shown below for the elliptic group $E_p(a,b) = E_{23}(1,1)$ which includes the point (4, 0) corresponding to the single value $y = 0$.

The elliptic curve cryptography can be used to encrypt plaintext messages, M, into cipher texts. The plaintext message M is encoded into a point P_M from the finite set of points in the elliptic group, $E_p(a,b)$.First step consists in choosing a generator point, $G \in E_p(a,b)$, such that the smallest value of n for which $nG = O$ is a very large prime number. Normally the traditional ECC protocol is that let the elliptic group $E_p(a,b)$ and the generator point G be in public. Each user selects a private key, say $n_A < n$ and compute the public key P_A as $P_A = n_A G$. Then, the case becomes encrypting the message point P_M for the partner, say from Alice to Bob. So Alice (A) chose a random integer k and computes the cipher text pair of points P_C using Bob's public key P_B:

$P_C = [(kG),(P_M+kP_B)]$ (9)

Bob received the cipher text pair of points, PC then multiplies the first point, (kG) with his private key, nB, and then adds the result to the second point in the cipher text pair of points as shown below:

$(P_M+kP_B)-[n_B(kG)]-P_M$ (10)

which is the plaintext point, corresponding to the plaintext message M. It is noted that only Bob can obtain retrieve the plaintext information P_M by the private key nB. The cryptographic strength of ECC lies in the difficulty for a cryptanalyst to determine the secret random number k from kP and P itself. The fast method to solve this problem is known as the elliptic curve logarithm problem (ECLP) [16]

III. HIDDEN GENERATOR POINT PROTOCOL:

It is clearly to see that ECC did not take care of the man-in-middle attacks even ECC itself has its cryptographic strength as described above. There should be many ways to carry on the hidden generator point protocol; even very few papers discussed this issue for ECC applications. The next section two ways to be discussed for implementing "hidden generator protocol," further reports about other methods will be discussed in other papers due to the paper size.

pick the prime number p = 23 (it is noted that this is only for explaining the new protocol, in real life the p is much bigger than this), we have quadratic residues group (p-1)/2 = 11 and

for this group the $E_P(a,b)$ can be shown as below:

$E_{23}(1,1)$= (0,1) (0,22) (1,7) (1,16) (3,10) (3,13) (4,0)

(5,4) (5,19) (6,4) (6,19) (7,11) (7,12) (9,7)

(19,16,)(1,13) (11,20) (12,4)(12,19) (13,7) (13,16)

(17,3) (17,20) (18,3) (18,20) (19,5) (19,18)

As we described in above that any point is sitting in equation (11) can be appointed as generator point "*G*,", or wemay expressed as

$\forall q_i \in E_{23}(1,1)$ can be point G, where q is acomponent of the $E_{23}(1,1)$. In a traditional way (as in section II) the G is fixed and let it be in public. But we shall make totally different way as described in the following contents. As the generator point, "G," that used to be appointed in public now will be hidden in our method, so that there is no way to know which point $q_i \in E_{23}(1,1)$ can be point G. Therefore, the attacker cannot make the "man-in-middle" attack, if it was the case to be happened. Next, we shall show two ways to complete the ECC processing with our *hidden generator point* algorism, namely (1) make a new protocol that has the common principle to work out the generator point ,say from the distribution of elliptic $E_p(a,b)$ group; or (2) by the new protocol to work out the P_M as shown below.

In order to make a common principle to work out the generator point for Alice and Bob, for example we attempt to use the distribution of elliptic $E_p(a,b)$ group. We first need to check what the distribution of $E_p(a,b)$ group looks like. For this example, the equation (11) is already given and can be shown in Figure 3

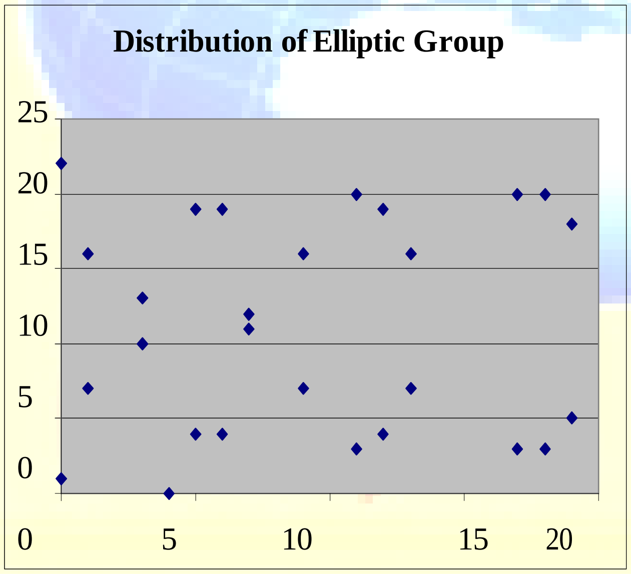

Figure 3. Distribution of Elliptic Group E23(1,1).

We may pick a generator point G by a character of the above distribution, say we pick the G when G (a, b) $\in$ Ep(a,b) with a = max {a} and b = max {b} (it is noted that other principle will apply, if it is designed as different principle). In this example, G = (19, 18). Note that we assume that put a = max {a} first, so choosing it a = 19 then choose b = max {b}. The order is important, in this example the appointed G it is not the G = (18, 20). When the generator point is fixed by our new protocol, we can have the following processing that was described in section II. It is obviously that this way is more secure as what a attacker has to do is that decrypting the message from Bob re-encrypt it with Alice's key and he can monitor the communication without detection. It involves the user of a trusted "certificate authority" (CA). When is queried the CA and returns a digitally signed "certificate" that can be compared to one that has been transmitted by another means. In an authenticated key exchange based on the difficulty of the kth root problem was described in section II. This way can be shown as Figure 4.

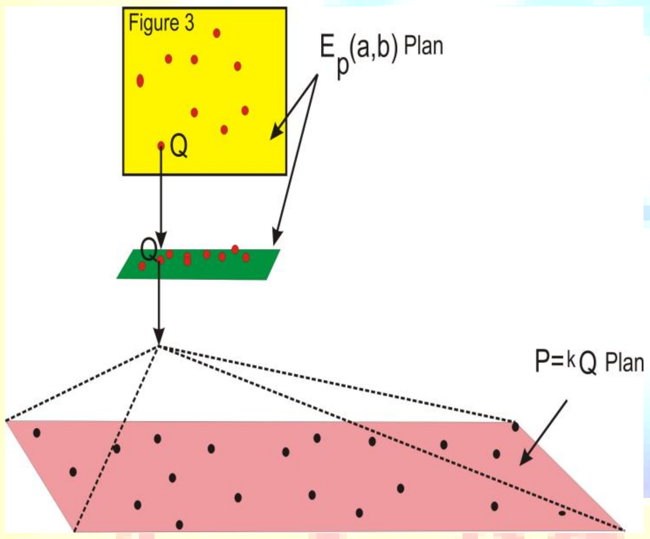

Figure 4. Block diagram for hidden generator point principle. The top plan is figure 3. When a generator is fixed by the distribution of the elliptic group then the *P = kQ* plan formed.

Now let's turn to the 2nd way, i.e., a new protocol to get the hidden generator point done.

In the 2nd approach, we need to address the case that Alice has no information about Bob's public key as traditional way does. Therefore if Alice would like to send a message to Bob, Alice cannot use public key to make cryptography to the message Alice wanted to send.

We may use, as an example, a protocol shown in Figure 5.

When Alice is going to send the message to Bob, Alice sends the pair of points P_C (as shown (1) in the figure) as below:

$$P_C = [(n_A^{-1}G),(n_A^{-1}P_M + n_A^{-1}G)] \qquad (12)$$

Here, n-1 A meets the equation: unity = n-1 .

A nA, we still called n-1 A as private key for Alice but there is no need to worry about the public key as G is hidden at current situation. So either PA or PB is not really useful in this case. When Bob received PC, he can operate as below:

$$n_A^{-1}P_M = n_A^{-1}P_M + n_A^{-1}G - n_A^{-1}G \qquad (13)$$

Then, Bob can make PD as below and sends it to Alice as shown the (2) in Figure 5.

$$P_D = n_A^{-1}n_B^{-1}P_M \qquad (14)$$

When Alice received PD, Alice can make PE and sent it to Bob as shown (3) in the Figure 5.

$$P_E = (n_A)P_D = (n_A)(n_A^{-1}n_B^{-1}P_M) = n_B^{-1}P_M \qquad (15)$$

Then when Bob received PE, Bob can obtain the message related PM that sent from Alice by

$$P_M = n_B n_B^{-1}P_M \qquad (16)$$

This can obtained only by Bob as no one has the private key that Bob has.

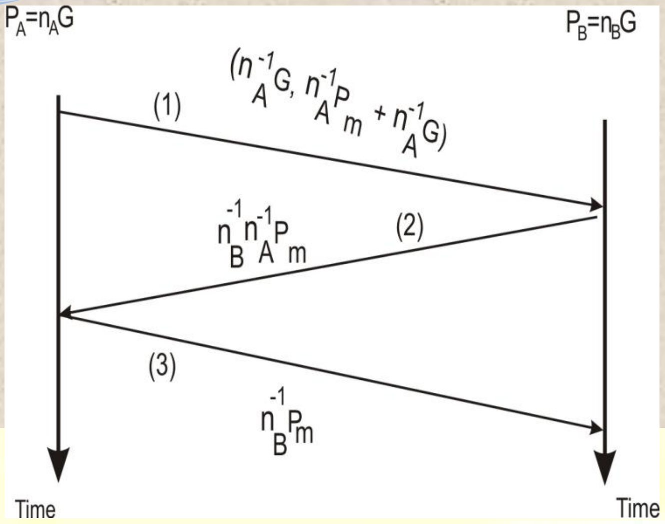

Figure 5. A protocol for the ECC with hidden generator point.

From the above discussion, it is clear that the first way described is less computing efficient in comparison with the second way due to the “mutual understanding” aspects of the protocol. However, it is need the “common principle” or “common protocol” before the communication. If this common protocol is to be sent by communication network it will have a risk to be attacked or it will have to create a “safe way” to inform first then go ahead for the rest. For the second way, it is clearly that it takes more time than that in traditional way to compute, which is the price to pay for protecting communications from the man-in-middle attacks without sending “common principle”. In fact, it is noted that if the “mutual understanding” is a function of the distribution of Ep (a,b), it could be a algorism for the hidden generator point, which will be discussed in our future paper.

IV. <u>**RELATED SPEEDING UP ALGORITHMS OVER ECC:**</u>

Experiments showed that for the popular and powerful ECCs , the most expensive operation in elliptic curve based cryptographic protocol is the scalar multiplication. Several researchers have investigated this issue, such as ECC using modified complementary [17], binary method [18], non adjacent form (NAF) [19], and mutual opposite form (MOF) [18] and complements method

[20], etc. The scalar multiplication is very expensive operation in elliptic curve based cryptographic protocol. Hence, the speed of scalar multiplication plays is a crucial factor in developing a an efficient system. Scalar multiplication is a computation of the form Q = kP, where P and Q are the elliptic curve points (as figures shown) and k is an integer. It can be obtained by repeated elliptic curve point addition and doubling operations. In the binary algorithms, the integer k is represented as

$$k = \sum_{j=0}^{l-1} k_j 2^j \text{ where } k_j \in \{0,1\}, \tag{17}$$

which scans the bits of k either from left-to-right or right-to-left. The cost of multiplication depends on the length of the binary representation of k and the number of Harming weight of scalar representation in this representation. If the representation (kn-1…k0)2 with kn-1≠0 then the number of doubling operation is (n-1). In an average, binary algorithm requires (n-1) doublings and (n-1)/2 additions. For example, k= 1778, then k= (11011111100)2 so computation of 1778 P requires 10 doublings and 5 additions. It is well known that a algorithm called non-adjacent form (NAF), based on the fact that k is represented as

$$k = \sum_{j=0}^{l-1} k_j 2^j \text{, with } k_j \in \{-1, 0, 1\}, \tag{18}$$

which using three digits {0, 1, -1}-radix 2 representation and this conversion is taken from right-to-left, which is different from equation (17). The average Hamming weight of signed binary representation is n/3 and it has the lower Hamming weight than the binary algorithm. However, it is noted the Hamming weight is one of keys to handle computation load, for example, k = 255, or (11111111)2, computation of 255P requires 7 point additions, but if it is transformed by (10000000-1)P, which is 256P-P, only one addition is required. Another algorithm is relevant, the mutual opposite form (MOF), which converts the binary string to MOF from the most significant bit efficiently. The n-bit binary string k is converted into a signed binary string. The conversion of MOF representation of an integer is highly flexible because conversion can be made either from right-to-left or left-to-right. The output of MOF is comparable efficiency with out of NAF as shown in. As described above it is clear that every mentioned algorithm targets decreasing the Hamming weight to increase efficiency of computation for ECC. The MOF and complementary algorithms have similar performance in terms of computation costs we may take complementary algorithm as part of hybrid algorithm as shown below. But we need to present the so-called "the 1's complement of binary numbers" described by Gillie [21, 22, 23]. The 1's complement of any binary number may be found

by the following equation [21]:

$$C_1 = (2^a - 1) - k \qquad (19)$$

Or $$k = (2^a - 1) - C_1 \qquad (20)$$

where

C1 = 1's complement of the number

a = number of digits to be handled by the computer

k = binary number whose 1's complement

As an example, let k =1788, or k = (11011111100)2 in its binary form. C1 = 1's Complement of the number of k and the a in this example it is in binary form is 11.

Therefore from the equation (18) we have:

$$C_1 = (2^a - 1) - k = (2^{11} - 1) - (11011111100) \\ = 00100000011 \qquad (21)$$

Therefore we

$$k = 1788 = (2^a - 1) - C_1 \\ = (2^{11} - 1) - 00100000011 \qquad (22)$$

This means $k = 1788 = (100000000000 - 00100000011 - 1)_2$, which gives :

$$1788 = 2048 - 256 - 2 - 1 - 1 \qquad (23)$$

Hence, from above equation we can see that the Hamming Weight of scalar k has reduced from original 8 to current 5, which will save 3 elliptic curve addition operations. One addition operation requires 2 squaring, 2 multiplications and 1 inverse operation. But if the original binary form of k is critical for this method as if the number of 1s in original binary form of k is > the one-half of the bit's length, i.e. 1's number ≥ a/2 then there is no need to convert the original binary format into "1's complement format" as our target is to decrease Harming weight.

Therefore, our proposed algorithm is that first check the 1's number of the binary form, if it is ≥ a/2, then go to the "complementary algorithm", if

it is not, then go to "1's complement format", i.e. go to the equation (19) then go to equation (20).

We have the algorithm as shown below:

Algorithm for faster scalar multiplication on elliptic curves:

```
1. $Q \leftarrow P_\infty$ and $i \leftarrow l - 1$
2.while $i \geq 0$ do
3.if $n_i = 0$ then $Q \leftarrow [2]Q$ and $i \leftarrow i - 1$
4.else
5. $s \leftarrow \max(i - k + 1, 0)$
6.while $n_s = 0$ do $s \leftarrow s + 1$
7.for $h = 1$ to $i - s + 1$ do $Q \leftarrow [2]Q$
8 $u \leftarrow (n_i \ldots\ldots n_s)_2$
[$n_i = n_s = 1$ and $i - s + 1 \leq k$]
```

```
9. $Q \leftarrow Q \oplus [u]P$ [$u$ is odd so that $[u]P$ is pre compute d]
10. i $\leftarrow s$ - 1
11.return $Q$
```

It is noted that there is a checking processing before we go ahead to calculate the scalar multiplication,. There is time cost but as the either way for the computation of the scale multiplication is the most efficient due to the minimizing Harming weight the saved time can pay the checking costs. In fact the final results, which are shown in the next section by the table, support this conclusion that the checking processing is relatively negligible cost in comparison with the saved time when the Harming weight is minimized.

V. **CONCLUSION:**

In recent years some cryptographic algorithms have obtained popularity due to properties that make them suitable for use in constrained environment such as mobile information appliances, sensor networks, where computing resources and power availability are limited. Elliptic curve cryptography (ECC) is one of them.

However, in the applications of ECC, in particular for the sensor networks there are always so-called man-in–middle attacks, in particular those networks are with very limited computing capacity and restricted power resources, which drew the researchers' attractions. In this paper we have presented two methods for protecting from man-in-middle attacks based on hidden generator point with ECC. The efficiency of ECC implementation is highly dependent on the performance of arithmetic operations of scalar multiplication. This paper based on discussions of the current major algorithms present a novel algorithm, hybrid of the "1's complement of binary number" and "complementary" to minimizing Harming weight to speed up the calculation over ECC. In terms of average, the proposed algorithm is about 12.5% saved time in comparison with the results of complementary based algorithm in [17].As we have seen from the check processing, there is always the case that the Hamming Weight will less than the half of the length (in terms of digit number) and the either complement of the number method or 1's complement of the number method will constantly keep the Hamming Weight minimizing, which makes this method sitting on the very power saving position due to the computing works.

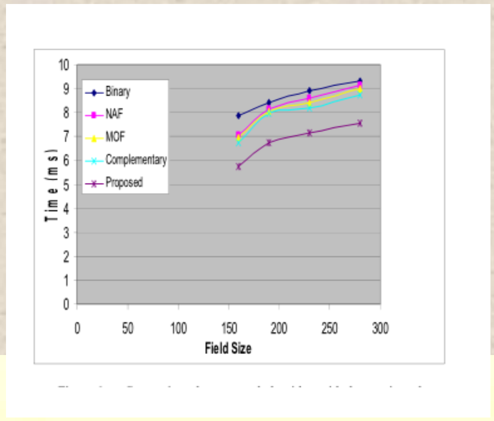

Figure 6.Comparison the proposed algorithm with the nominated algorithms by [17] (the data for the nominated algorithms were used from the same reference).

REFERENCES:

- V.S. Miller, "uses of elliptic curves in cryptography," in Advances in Cryptology, CRYPTO'85, ser . Lecture Notes in Computer Science, vol. 218, Springer, 1986. pp. 417-428.
- N. Koblitz, " Elliptic curve cryptosystems," Mathematics of Compution, vol. 48, no.177, pp.203-209, Jan 1987.
- D. Hakerson, A. Menezes, and S. Vanston , "Guide to Elliptic Curve Cryptography," Springer-Verlag, NY (2004).
- H. Cohen, A Miyaji and T. Ono, "Efficient elliptic curve exponentiation using mixed coordinates," Lectures Notes in Computer Science, 1514, 51-65 (1998).
- V. Dimitrov V., L. Imbert, and P. K. Mishra, "Efficient and secure elliptic curve point multiplication using double-base chains," Lectures Notes in Computer Science, 3788, 59-78 (2005).
- 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).
- D. Bernstein, "High-speed diffie-hellman, part 2," presented at the INDOCRYPT'06 tutorial session, Dec. 11-13, Kolkata, India (2006).

- K. Kaabneh and H. Al-Bdour, “Key exchange protocol in elliptic curve cryptography with no public point,” American Journal of Applied Sciences 2 (8): 1232-1235, 2005.
- J. Adikari, V. Dimitrov, and L. Imbert, “Hybrid binary-ternary joint sparse from and its application in ellipic curve cryptography,” Cryptology ePrint Archive, Report 2008/285, 2008.
- Bangju Wang, Huanguo Zhang and Yuhua Wang, “An efficient elliptic curves scalar multiplication for wireless network,” 2007 IFIP International Conference on Network and Parallel Computing-Workship, pp131.