Fuzzy Controlling Window for Elliptic Curve Cryptography in Wireless Networks

Xu Huang  
Faculty of Information Sciences and Engineering  
University of Canberra  
Canberra, Act 2601 Australia  
Xu.Huang@canberra.edu.au

Dharmendra Sharma  
Faculty of Information Sciences and Engineering  
University of Canberra  
Canberra, Act 2601 Australia  
Dharmendra.Sharma@canberra.edu.au

Abstract—The rapidly developing wireless communications and embedded micro-electro systems has made wireless sensor networks (WSN) possible for extensive applications. However, the security of the WSN becomes one of the major concerns in these applications. Elliptic curve cryptography (ECC) prominently provides solid potential for WSN security with its small key size and its high security strength. However, in order to satisfy the full range of applications there is an urgent need to reduce key calculation time. Due to scalar multiplication operation in ECC takes about 80% of key calculation time on WSN, this paper we proposed a fuzzy controller for a dynamic window allowing the calculation processing to run under optimum conditions with the balanced for RAM and ROM at the sensor node within WSN. The whole quality of Service (QoS) is improved, in particular the power consuming is more efficiently. The simulation results showed that the average calculation time decreased by approximately 15% in comparison to traditional algorithms in an ECC WSN.

Keywords- fuzzy controller, Elliptic curve cryptography (ECC), scalar multiplication, non-adjacent form, slide window

I. INTRODUCTION

The rapidly developing of wireless communications has become popular and important in our daily life, together with rapid growth in very large scale integrated (VLSI) technology, embedded systems and micro electro mechanical systems (MEMS) has enabled production of inexpensive sensor nodes more popular, which can communicate information over shorter distances with efficient use of power [1]. In the WSN systems, the sensor node will detect the interested information, processes it with the help of an in-built microcontroller and communicates results to a sinker or base station. Generally, the base station is a more powerful and functional node, which can be linked to a central station via satellite or internet communication to form a network. There are many deployments for WSNs depending on various applications including environmental monitoring e.g. detection of radioactive sources [2], distributed control systems [3], volcano detection [4,5], and computing platform for tomorrow’s internet [6], agricultural and farm management [7].

Contrast to traditional networks, a WSN generally has many resource constraints [4] due to the limited size. As an example, the MICA2 mote consists of an 8 bit ATMega 128L microcontroller working on 7.3 MHz. As a result nodes of WSN have limited computational power. Radio transceiver of MICA motes can normally achieve maximum data rate of 250 Kbits/s, which restricts available communication resources. The flash memory that is available on the MICA mote is only 512 Kbyte. Apart from these limitations, the onboard battery is 3.3 V with 2A-Hr capacity. Therefore, those above restrictions with the current state of art protocols and algorithms become a challenge for the wireless sensor networks due to their high communication overhead. ECC bring the light for this challenge with its great potential.

Elliptic Curve Cryptography was first introduced by Neal Koblitz [9] and Victor Miller [10] independently in the early eighties. The advantage of ECC over other public key cryptography techniques such as RSA, Diffie-Hellman is that the best known algorithm for solving elliptic curve discrete logarithm problem (ECDLP) which is the underlying hard mathematical problem in ECC which will take the fully exponential time. On the other hand the best algorithm for solving RSA and Diffie-Hellman takes sub exponential time [11]. Therefore, the ECC problem can only be solved in exponential time and, to date, there is a lack of sub exponential methods to attack ECC. It is the target for this paper to propose a fuzzy controller for a dynamic window allowing the calculation processing to run under optimum conditions with the balanced for RAM and ROM at the sensor node within WSN.

Let’s briefly have a look at an ECC. An elliptic curve E over GF(p) can be defined by $y^2 = x^3 + ax + b$ where $a, b \in \text{GF}(p)$ and $4a^3 + 27b^2 \neq 0$ in the GF(p).

The point $(x, y)$ on the curve satisfies the above equation and the point at infinity denoted by $\infty$ is said to be on the curve.

If there are two points on the curve namely, $P(x_1, y_1)$, $Q(x_2, y_2)$ and their sum is given by point $R(x_3, y_3)$ the algebraic formulas for point addition and point doubling are given by following equations:

We have: $x_3 = \lambda^2 - x_1 - x_2$

$y_3 = \lambda(x_1 - x_3) - y_1$

$\lambda = \frac{y_2 - y_1}{x_2 - x_1}$, if $P \neq Q$
\[ \lambda = \frac{3x^2 + a}{2y_1}, \quad \text{if } P = Q \]

![Figure 1: Point addition and point doubling on elliptic curve.](image)

Where the addition, subtraction, multiplication and inverse are the arithmetic operations over \( \text{GF}(p) \), which can be shown in Figure 1.

II. ELLIPTIC CURVE DIFFIE-HELLMAN SCHEME FOR WSN

Before we get into our innovation method, it is necessary to have a closer look at the popular legacy scheme for WSNs. As the paper [13] showed that the original Diffie-Hellman algorithm with RSA requires a key of 1024 bits to achieve sufficient security but Diffie Hellman based on ECC can obtain the same security level with only 160 bit key size.

A classical operation of a Elliptic Curve Diffie Hellman scheme can be shown in the Figure 2.

We can initially assume that Alice and Bob agree on a particular curve with base point \( P \). They generate their public keys by multiplying \( P \) with their private keys namely \( K_A \) and \( K_B \). Then, sharing public keys, they generate a shared secret key by multiplying public keys by their private keys. The secret key can be obtained by \( R = K_A * Q_A = K_B * Q_B \). With the known values of \( Q_A, Q_B \) and \( P \), it is computationally intractable for an eavesdropper to calculate \( K_A \) and \( K_B \) which are the private keys of Alice and Bob. As a result, adversaries cannot calculate \( R \) the shared secret key.

In ECC two heavily used operations are involved namely, scalar multiplication and modular reduction. Gura et. al. [14] showed that 85% of execution time is spent on scalar multiplication. Scalar multiplication is the operation of multiplying point \( P \) on an elliptic curve \( E \) defined over a field \( \text{GF}(p) \) with positive integer \( k \) which involves point addition and point doubling. Operational efficiency of \( kP \) is always affected by the type of coordinate system chosen for the point \( P \) on the elliptic curve and the algorithm used for recoding of integer \( k \) in scalar multiplication.

In this research paper we propose an innovative algorithm is based on one’s complement for representation of integer \( k \) which can accelerate the computation of scalar multiplication in wireless sensor networks. Then, we propose a fuzzy controlling dynamic window to make computations always under optimum conditions with the balanced for RAM and ROM at the sensor node within WSN.

The number of point doubling and point addition operations in scalar multiplication depends on the recoding of integer \( k \). Expressing integer \( k \) in binary format highlight this dependency.

![Figure 2: Diffie-Hellman protocol based on ECC](image)

The number of zeros and number of ones in the binary form, their places and the total number of bits will affect the computational cost of scalar multiplications. The Hamming weight as represented by the number of non-zero elements, determines the number of point additions and bit length of integer \( K \) determines the number of point doublings operations in scalar multiplication.

A point addition when \( P \neq Q \) requires one field inversion and three field multiplications [13]. Squaring is counted as regular multiplication. This cost can be expressed by \( 1I + 3M \), where \( I \) denotes the cost of inversion and \( M \) denotes the cost of multiplication.

A point doubling when \( P = Q \) requires \( 1I + 4M \) as we can neglect the cost of field additions as well as the cost of multiplications by small constant 2 and 3 in the above formulae.

In Binary Method, a scalar multiplication is the computation of the form \( Q = kP \), where \( P \) and \( Q \) are the elliptic curve points and \( k \) is positive integer. This is implemented by repeated elliptic curve point addition and doubling operations. In binary method the integer \( k \) is represented in binary form:

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

In the binary method it scans the bits of \( K \) can be either from left-to-right or from right-to-left. The binary method for the computation of \( kP \) is given in the following algorithm 1, as shown below:

**Algorithm 1:** In Binary Method, left to right for point multiplication

| Input: A point \( P \in E(\text{F}_p) \), an \( l \) bits integer \( k = \sum_{j=0}^{l-1} K_j 2^j \), \( K_j \in \{0,1\} \) |
| Output: \( Q = kP \) |
| 1. \( Q \leftarrow 0 \) |
| 2. For \( j = l \) to 0 do: |
| 2.1 \( Q \leftarrow 2Q \) |
| 2.2 If \( K_j \) = 1 then \( Q \leftarrow Q + P \) |
| 3. Return \( Q \) |
The cost of multiplication, within binary method, depends on the number of non-zero elements and the length of the binary representation of \( k \). If the representation appears \( k_{i,j} \neq 0 \) then binary method will require \((l-1)\) point doublings and \((W-1)\) where \( l \) is the length of the binary expansion of \( k \), and \( W \) is the Hamming weight of \( k \) (i.e., the number of non-zero elements in expansion of \( k \)). As an example, if \( k = 629 = (1001111011)_{2} \), it will require \((W-1) = 6 - 1 = 5\) point additions and \((l-1) = 10 - 1 = 9\) point doublings operations.

In Signed Digit Representation Method, a subtraction has virtually the same cost as addition in the elliptic curve group. The negative of point \((x,y)\) can be expressed as \((x, -y)\) for odd characters. This leads to scalar multiplication methods based on addition–subtraction chains, which will help to reduce the number of curve operations. When integer \( k \) is represented with the following form, which is a binary signed digit representation.

\[
k = \sum_{j=0}^{l} S_j 2^j, \quad S_j \in \{1,0,-1\}
\]

When a signed-digit representation has no adjacent non-zero digits, i.e. \( S_j S_{j+1} = 0 \) for all \( j \geq 0 \) it is called a non-adjacent form (NAF).

The following algorithm 2 computes the NAF of a positive integer given in binary representation.

**Algorithm 2: Conversion from Binary to NAF**

**Input:** An integer \( k = \sum_{j=0}^{l-1} K_j 2^j, \quad K_j \in \{0,1\} \)

**Output:** NAF \( k = \sum_{j=0}^{l} S_j 2^j, \quad S_j \in \{1,0,-1\} \)

1. \( C_0 \leftarrow 0 \)
2. For \( j = 0 \) to \( l \) do:
3. \( C_{j+1} \leftarrow [(K_j + K_{j+1} + C_j)/2] \)
4. \( S_j \leftarrow K_j + C_j - 2C_{j+1} \)
5. Return \((S_0, S_1, \ldots, S_l)\)

NAF usually has fewer non-zero digits than binary representations. The average hamming weight for NAF form can be obtained as \((n-1)/3.0\). So generally NAF requires \((n-1)\) point doublings and \((n-1)/3.0\) point additions. The binary method can be revised accordingly and is given another algorithm for NAF, and this modified method is called the Addition Subtraction method.

### III. Window with Variable Sizes in ECC

Next we are going to use the algorithm based on subtraction by utilization of the 1’s complement is most common in binary arithmetic. The 1’s complement of any binary number may be found by the following equation [19-22]:

\[
C_1 = (2^a - 1) - N
\]

where \( C_1 \) = 1’s complement of the binary number, \( a \) = number of bits in \( N \) in terms of binary form, \( N \) = binary number

From a closer observation of the equation (1), it reveals the that any positive integer can be represented by using minimal non-zero bits in its 1’s complement form provided that it has a minimum of 50% Hamming weight. The minimal non-zero bits in positive integer scalar are very important to reduce the number of intermediate operations of multiplication, squaring and inverse calculations used in elliptic curve cryptography as we have seen in previous sections.

The equation (1) can therefore be modified as per below:

\[
N = (2^a - C_1 - 1)
\]

For an example, we may take \( N=1788 \), then it appears \( N= (110111111100)_{2} \) in its binary form as \( C_1 \), 1’s Complement of the number of \( N= (00100000011)_{2} \) is in binary form so we have \( a = 11 \)

After putting all the above values in the equation (2) we have:

\[
1788 = 211 - 00100000011 \rightarrow 1, this can be reduced as below:
1788 = 100000000000 - 00100000011 \rightarrow 1
\]

So we have

\[
1788 = 2048 - 256 - 2 - 1 \rightarrow 1
\]

It is evident, from equation (3), that the Hamming weight of scalar \( N \) has reduced from 8 to 5 which will save 3 elliptic curve addition operations. One addition operation requires 2 Squaring, 2 Multiplication and 1 inverse operation. In this case a total of 6 Squaring, 6 Multiplication and 3 Inverse operations will be saved.

The above recoding method based on one’s complement subtraction combined with sliding window method provides a more optimized result.

As an example, let us compute \( [763] P \) (in other words \( k = 763 \), with a sliding window algorithm with \( K \) recoded in binary form and window sizes ranging from 2 to 10. It is observed that as the window size increases the number of pre-computations also increases geometrically. At the same time number of additions and doubling operations decrease.

Now we present the details for the different window size to find out the optimal window size using the following example:

**When Window Size \( w = 2 \)**

\[
763 = (10111111011)_{2}
\]

No of precomputations = \( 2^2 - 1 = 2^2 - 1 = [3] P \)

\[
763 = 111110 10 11\]

The intermediate values of \( Q \) are

\[
\]

Computational cost = 9 doublings, 4 additions, and 1 pre-computation.

**When Window Size \( w = 3 \)**

No of pre-computations = \( 2^3 - 1 = 2^3 - 1 = [7] P \)


\[

The intermediate values of \( Q \) are

\[
\]

Computational cost = 7 doublings, 3 additions, and 3 pre-computation.
computations.

**Algorithm for sliding window scalar multiplication on elliptic curves.**

1. $Q \leftarrow P$, and $i \leftarrow l - 1$
2. while $i \geq 0$
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. $u \leftarrow (n_1, \ldots, n_s)$, $[n_i = n_s = 1$ and $i - s + 1 \leq k]$
9. $Q \leftarrow Q \oplus [u]P$ [u is odd so that [u]P is precompute d]
10. $i \leftarrow s - 1$
11. return $Q$

We continue to derive the remaining calculations for window size $w = 6$, window size $w = 7$, window size $w = 8$, window size $w = 9$, and window size $w = 10$. The results for all calculations are presented in Table 1.

The effects of “doublings” and “additions” as shown in Table 1 are further considered below.

| WINDOW SIZE VS NO OF DOUBLINGS, ADDITIONS AND PRE COMPUTATIONS |
|-----------------|-----------------|-----------------|-----------------|
| Window Size     | No of Doublings | No of Additions | No of Pre computations |
| 2               | 9               | 4               | 1               |
| 3               | 7               | 3               | 3               |
| 4               | 6               | 2               | 7               |
| 5               | 5               | 1               | 15              |
| 6               | 4               | 1               | 31              |
| 7               | 3               | 1               | 61              |
| 8               | 3               | 1               | 127             |
| 9               | 1               | 1               | 251             |
| 10              | 0               | 0               | 501             |

**IV. WINDOW WITH A FUZZY CONTROLLER IN ECC**

It is shown, from Table one, that there is a tradeoff between the computational cost and the window size. However, it should realize that this tradeoff is underpinned by the balance between computing cost (or the RAM cost) and the precomputing (or the ROM cost) of the node in the network.

It is also shown that, from above description, that the variety of wireless sensor network applications will create many difficulties for controlling the complex trade-off. Therefore, we propose a fuzzy controller system to provide dynamic control, with which to ensure the optimum window size is obtained for the tradeoff between the cost of pre-computation and computation.

The fuzzy decision problem, introduced by Bellman and Zadeh, has as a goal the maximization of the minimum value of the membership functions of the objectives to be optimized. Normally, the fuzzy optimization model can be represented as a multi-objective programming problem as follows [21]:

$$
\text{Max : } \min \{\mu_i(D)\} \& \min \{\mu_i(U)\} \quad \forall s \in S \& \forall l \in L
$$

such that $A_i \leq C_i \quad \forall l \in L,$

$$
\sum_{r \in R_s, x_r = 1} \forall p \in P \& \forall s \in S, x_{rs} = 0 \text{ or } 1 \quad \forall r \in R \& \forall s \in S
$$

In above equation, the objective is to maximize the minimum membership function of all delays, denoted by $D$, and the difference between the recommend value and the measured value, denoted by $U$.

Our fuzzy control system is extended from our previous work and shown in Figure 3. For accurate control, we designed a three inputs fuzzy controller. The first input is storage room, which has three statuses, showing storage room in one of the three, namely (a) low, (b) average, and (c) high. The second input is pre-computing working load (PreComputing) in one of three states, namely (a) low, (b) average, and (c) high. The third input is Doubling, expressing how much working load for the calculation “doubling” which has three cases, namely (a) low, (b) average, and (c) high. The output is one, called WindowSize, to express the next window size should be moved in which way, which has three states for the window sizes, namely (a) down, (b) stay, and (c) up.

![Figure 3: Three inputs fuzzy window control system](image)

There are 26 Fuzzy Rules listed as follows (wight are unit):

1. If (StorgeRoom is low) and (PreComputing is low) and (Doubling is low) then (WindowSize is Up)
2. If (StorgeRoom is low) and (PreComputing is low) and (Doubling is average) then (WindowSize is Up)
3. If (StorgeRoom is low) and (PreComputing is low) and (Doubling is high) then (WindowSize is stay)
4. If (StorgeRoom is low) and (PreComputing is average) and (Doubling is low) then (WindowSize is Up)
5. If (StorgeRoom is low) and (PreComputing is average) and (Doubling is average) then (WindowSize is Up)
6. If (StorgeRoom is low) and (PreComputing is average) and (Doubling is high) then (WindowSize is stay)
7. If (StorageRoom is low) and (PreComputing is high) and (Doubling is low) then (WindowSize is Up)
8. If (StorageRoom is low) and (PreComputing is high) and (Doubling is average) then (WindowSize is stay)
9. If (StorageRoom is low) and (PreComputing is high) and (Doubling is high) then (WindowSize is stay)
10. If (StorageRoom is low) and (PreComputing is low) and (Doubling is low) then (WindowSize is Up)
11. If (StorageRoom is average) and (PreComputing is low) and (Doubling is average) then (WindowSize is Up)
12. If (StorageRoom is average) and (PreComputing is low) and (Doubling is high) then (WindowSize is stay)
13. If (StorageRoom is average) and (PreComputing is average) and (Doubling is low) then (WindowSize is Up)
14. If (StorageRoom is average) and (PreComputing is average) and (Doubling is average) then (WindowSize is stay)
15. If (StorageRoom is average) and (PreComputing is average) and (Doubling is high) then (WindowSize is Down)
16. If (StorageRoom is average) and (PreComputing is high) and (Doubling is average) then (WindowSize is stay)
17. If (StorageRoom is average) and (PreComputing is high) and (Doubling is high) then (WindowSize is stay)
18. If (StorageRoom is high) and (PreComputing is low) and (Doubling is low) then (WindowSize is Up)
19. If (StorageRoom is high) and (PreComputing is low) and (Doubling is average) then (WindowSize is Down)
20. If (StorageRoom is high) and (PreComputing is low) and (Doubling is high) then (WindowSize is Down)
21. If (StorageRoom is high) and (PreComputing is average) and (Doubling is low) then (WindowSize is stay)
22. If (StorageRoom is high) and (PreComputing is average) and (Doubling is average) then (WindowSize is stay)
23. If (StorageRoom is high) and (PreComputing is average) and (Doubling is high) then (WindowSize is Down)
24. If (StorageRoom is high) and (PreComputing is high) and (Doubling is low) then (WindowSize is Down)
25. If (StorageRoom is high) and (PreComputing is high) and (Doubling is average) then (WindowSize is Down)
26. If (StorageRoom is high) and (PreComputing is high) and (Doubling is high) then (WindowSize is Down)

The three inputs with 26 fuzzy rules in Mamdani model running fuzzy controller part is shown in Figure 4. The three inputs are StorageRoom, PreComputing and Doubling. The output is WindowSize.

The output with StorageRoom and PreComputing is shown in Figure 5. The surface StorageRoom vs. Doubling is shown in Figure 6.

Figure 4: The block diagram of current fuzzy controller

Figure 5: The output of the surface for the StorageRoom vs. PreComputing.

The surface StorageRoom vs. PreComputing is shown in Figure 7.

From above figures, it is clearly observed that in the low window size side, if the storage room is low the dominated function of “doubling” will play role as Figure 5 shown but if the window size is at the high side, the storage room will be fairly stay at the middle either for PreComputing or Doubling, which is the doubling will sharply increased when window size a little bit larger that also can be shown from Table 1. From Figure 6 it is clearly to show when the storage room is getting big, it would be nice to have larger window size for the “doubling”.

Now if we change the weight for above fuzzy rules as such the rules 1, 5, 10, 13, 14, 15, 16, 18, 20, 21, 22, 23, 25, and 26 are set in 0.5 (the rest will keep the same) due to the major functions are controlled by the storage room, and doubling will rapidly increasing by the window size larger. The outputs will changed as the average storage room will increased 0.04% and the other two inputs are decreased by 0.02% the output become window staying a little wider side by 0.003%.

Figure 6: The output of the surface for the StorageRoom vs. Doubling.
It is clear that this fuzzy controller for the dynamic window is also involved a tradeoff between accuracy and control costs. For example the same system may go further for the second order parameters, not just check the changes about the input variables but also check the change tendencies of the variables, which will be discussed in our another paper.

If we keep the storage constant and the situation shown by Figure 7 is how those two major factors shown in Table 1 to impact on the output.

![Figure 7: The output of the surface for the StorageRoom = constant and PrecComputing vs. Doubling.](image)

The simulations of the example described in above were implemented. With equation (2), the computational cost has been reduced from 3 additions as in the binary method to only 1 addition in one’s complement subtraction form. The number of pre-computations has remained the same. This can be proved for different window sizes.

In our simulations, the proposed method together with a fuzzy window size controller makes the ECC calculation almost 15% more efficient than traditional methods in ECC wireless sensor network.

V. CONCLUSION

The positive integer in point multiplication may be recorded with one’s complement subtraction to reduce the computational cost involved in this heavy mathematical operation for wireless sensor network platforms. As the NAF method involves modular inversion operation to get the NAF of binary number, the one’s complement subtraction form can provide a very simple way of recoding the integer. There is always decision between pre-computing and computing, the former is related to the storage and the latter is associated with computing capability and capacity. The window size may be the subject of trade-off between the available RAM and ROM at a particular instance on a sensor node, which can be controlled by fuzzy controller. The final simulation in a sensor wireless network shows that about 15% more efficient than transitional method can be obtained with ECC.

REFERENCE