Efficient Implementation of Arithmetic Operations in ECC over Binary Fields

G.N.Purohit
Department of Mathematics
AIM & ACT, Banasthali University,
Rajasthan, Banasthali-04022,
(INDIA)

Asmita Singh Rawat
Department of Computer Science
AIM & ACT, Banasthali University,
Rajasthan, Banasthali-304022,
(INDIA)

ABSTRACT
In this paper the implementation of arithmetic operations in ECC is described.Elliptic curve cryptography is very useful in the field of the network security because of its small key size and its high strength of security.In this paper briefly describing general arithmetic operations we focus on scalar multiplication . We present two techniques: (i)reducing Hamming weight of scalars in binary representation and (ii) using sliding window , for obtatining scalar multiplication in a faster manner. Use of both the techniques is explained by suitable examples.

General Terms
Elliptic curve cryptography, scalar multiplication, wireless sensor, binary fields

Keywords
Elliptic Curve Discrete Logarithm,Scalar multiplication, Diffie-Hellman Algorithm, Sliding window, hamming weight..

1. INTRODUCTION
Wireless sensor networks consist of small nodes that sense their environment, process data and communicate through wireless links. They are expected to support a wide variety of applications, many of which have at least some requirement for security. In wired data networks node rely on pre deployed trusted server help to establish trust relationship but in WSN,their trusted authorities do not exist because sensor nodes have limited memory , CPU power , and energy , hence cryptographic algorithms must be selected carefully. A survey of security issues in ad hoc and sensor networks can be found in [1].Elliptic curve cryptography (ECC) offers a popular solution to the problem of implementing public key cryptography on mobile computing device. The security of RSA, the most popular algorithm in other domains such as e-commerce is based on the hardness of integer factorization; ECC is based on the Elliptic Curve Discrete Logarithm Problem (ECDLP). ECC keys are shorter than their RSA analogues, while achieveing the same security level: A 160-bit ECC key is roughly equivqlent to a 1024-bit RSA key.Thus an ECC based system is typically more efficient and utilizes less resources than one based on RSA[2] and hence ECC has emerged as a promising alternative to traditional public key methods on WSNs [3],because of its lower processing and storage requirements. These features motivate the search for increasingly efficient algorithms and implementation of ECC for such devices.

In this paper we propose optimization for implementing ECC over binary fields, improving its performance and viability. The main operations in any

ECC based primates such as key exchange or encryption is the scalar multiplication which can be viewed as the top level operation. The point Scalar multiplication is achieved by repeated point addition and doubling. All algorithms for modular exponential can also be applied for point multiplication. In this paper we propose an algorithm for scalar multiplication which remarkably improves the computational efficiency of scalar multiplication.

The remaining of this paper is organized as follows. Related work is presented in Section.2 and

Elementary elliptic curve concepts are introduced in Section.3. Optimization in ECC implementation is considered in Section.4. Including optimization for scalar point multiplication using sliding window. Concluding remarks is included in section.5.

2. RELATED WORK
Since the introduction of elliptic curves to cryptography by Victor Miller [4] and Neal Koblitz [5] independently in 1985 a vast amount of research has been done. While implementing ECC in binary fields, elliptic curve point representation and point operation algorithms are of special significance. A comprehensive survey of binary fields and elliptic curve arithmetic for the NIST recommended elliptic curves was done in [5]. A specialized implementation for the field GF (2^{155}) was done in [6] Lopez and Dahab presented a Montgomery field multiplication for binary fields in [7]. They also did research on binary fields multiplication [8] and ECC over binary fields [9]. More point multiplication algorithms can be found in [10] and [11]. Solinas developed efficient algorithms for Koblitz curves over binary fields using complex multiplication [12]. Shantz [13] presented an efficient technique to calculate modular division, which is an important arithmetic operation in ECC and other cryptographic system . Cohen et al [14] analysed the impact of coordinate system in ECC implementation. They measured the performance of point addition (PADD) and Point Doubling(PDBL) of different coordinate system,which achieves the fastest doubling operations for binary

curves. Malan et.al[15] implemented an ECC system using polynomial basis over binary field GF(2^m).

3. ELLIPIC CURVE CRYPTOGRAPHY

3.1. Basic Concepts.

In this section, we briefly give a background about ECC and corresponding elliptic curve Diffie-Hellman Algorithm. In this section , we briefly give a background about ECC and corresponding elliptic curve Diffie-Hellman.In recent years, ECC has attracted much attention as security solutions for wireless networks due to small key size and low computational overhead. An elliptic curve over a finite field GF(Galois field) is composed of a finite group of points (x_i, y_i) where integer coordinates x_i, y_i satisfy the long Weierstrass Equation. $y^3 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$

And the coefficients a_1, a_2, a_3, a_4, a_6 are elements of GF and are the parameters of the curve. The curve discriminent is $\Delta \neq 0$ and there is also a point at infinity denoted by Θ. If GF is a field of characteristics 2, then the curve is called a binary elliptic curve.Since the field is GF (q), *q* is prime, is generally used is cryptographic applications, the equation of elliptic curve is simplified to

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

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

where $a, b \in$ *GF(q)*

The elliptic curve group operation is closed so that the addition of any two elements (points) is again an element (point0 of EC. The point at infinity O, is the identity element of the group.

Given two points *P* and *Q*, with coordinates (x_1, y_1) and (x_2, y_2) respectively, their addition results in a point R on the curve with coordinates (x_3, y_3) where x_3 and y_3 satisfy $(x_1, y_1) + (x_2, y_2) = (x_3, y_3)$

Such that

$$x_3 = \lambda^2 + \lambda + x_1 + x_2 + a$$

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

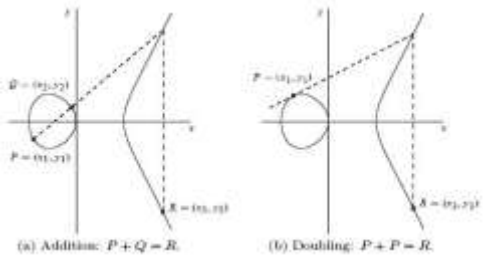

Fig.1.

where $\lambda = \dfrac{y_1 + y_2}{x_1 + x_2}$

If *P=Q*, then *R=P+P=2P* and in this case the coordinates of (x_3, y_3) of R are given as

$$x_3 = \lambda^2 + \lambda + a$$

$$y_3 = x_1^2 + (\lambda + 1)x_3$$

where $\lambda = \dfrac{x_1 + y_1}{x_1}$

As stated earlier O, is the identity element of group and stratifies

P+O=O+P=P

For P(x, y) we define –P=$(-x, y)$ as the unique inverse of P in the group satisfying.

P+ (-P) = (-P) +P=O

A multiplication

$$Q = \underbrace{P + \ldots\ldots + P}_{k-times; k \in N} = k.P$$

of a point P with an integer *k* can be seen as the multiplication addition of one and the same point *k*-times. The resulting product *Q* is again another point on the curve. Given an elliptic curve *C* over finite field F_{2^p} , a point P and a product *Q* the problem is to find a $k \in N$ that holds *Q=k.P*. This problem is known as Elliptic Curve Discrete Logarithmic Problem(ECDLP)and hard to solve[ ]. For example with a finite fields with F_{2^p} 2^P elements you need about $O(2^{P/2})$operations to find *k*.Thus the ECC relies on the difficulty of the ECDLP, that is given points *P* and *Q* in the group , to find the number *k* such that *Q=k.P*.

3.2 ECDH and ECDSA

A typical Elliptic Curve Diffie-Hellman (ECDH) scheme is illustrated in Fig.1. Initially *A* and *B* agree a system base point *P* and generate their own public keys Q_A and Q_B . Then to share a secret A and B exchange their public keys and then use their own private key to multiply the others public key . The result point *R* will be a secret . An eavesdropper *E* , may learn the public keys from *A* and *B*.

However ,

$A \xrightarrow{Q_A = k_A \cdot P} B$

(Private Key k_A) (Private Key k_B)

(Compute secret) $\xleftarrow{Q_B = k_B \cdot P}$ (Compute secret)

$R = k_A \cdot Q_B$ $R = k_B \cdot Q_A$

With the knowledge of *P* , Q_A and Q_B it is computationally intractable for *E* to get *A's* and *B's* private keys and as such *E* cannot Figure out *R*. ECC can also be used for Digital Signature Algorithms (EDSA), similarly. Suppose *A* wants to send a signed message to *B* and agree on a system base point *P* and let order of *P* in finite field F(p) be q. While A sending a message to B attaches a Digital Signature (r, s) generated by following algorithm. Let us assume that A has private key x and a public key $Q_A = xP$.

1.Select a random key *k* in $[1, q-1]$

2. Let *kP* be point (x_1, y_1) on EC. Let $r = x_1$ .If $r(\text{mod} q) = 0$ then select another k.

3. Computec $k^{-1}(\text{mod } q)$

4.Compute $s = k^{-1}(Hash(m) + xr)$,where Hash is one way hash function SHA-I. If $\Delta=0$ then select another *k*.

5. (r, s) is the digital singnature

To verify the message and the signature , B has to compute the following

1.Compute $w = s^{-1}(\text{mod } q)$ and $H(m)$.

2. Compute $u_1 = H(m)w \bmod q$ and $u_2 = rw \bmod q$

3. Compute $u_1 P + u_2 Q$, which results in point (x_2, y_2).

4. The signature is verified if $x_2 = r$.

4.OPTIMIZATIONINECC IMPLEMENTATION

In ECC implementation we perform large integer arithmetic operations including additions , subtraction, shift , multiplication, division and modular reduction. The efficiency of large integer multiplication dominates the overall performance of ECC operation. Gura et al.[ ] show that as much as 85% of execution time is spent on multiplication for a typical point multiplication in ECC. Thus the optimization on multiplication is critical for overall performance

of ECC implementation. We now present optimization for ECC operations . Starting with PADD and PDBL in ECC we consider different optimization techniques for point multiplication.

4.1. ECC Addition and Doubling

Since point multiplication can be decomposed to a series of addition and doubling operations , therefore PADD and PDBL are the fundamental ECC operations .Performing PADD and PDBL in affinie coordinates needs integer inversion ,which is considered much slower then integer multiplication. Cohen et al [15] considered these operations in Projective coordinates and Jacobian coordinates and ascertained that their use yields better performance . Further they came out with a mixed coordinates ,a combination of Affine coordinate and Modified Jacobian coordinates , which led to best performance [ ]. Cconsider a point *P* in ECC in Modified Jacobian coordinate $P(X_1, Y_1, Z_1, aZ_1^4)$ and a point Q in Affine coordinates $Q(x_2, y_2)$, then their addition is a point R with $R(X_3, Y_3, Z_3, aZ_3^4)$ in Modified Jacobian coordinates , where

$$X_3 = -W^3 - 2X_1W^2 + Y^2$$

$$Y_3 = -Y_1W^3 + V(X_1W^2 - X_3)$$

$$Z_3 = aZ_3^4$$

where $W = x_2Z_1^2 - X_1$ and $V = y_2Z_1^3 - Y$ .The result of PDBL for $P_3 = 2P_1$ is given by the following formula

$$X_3 = T$$

$$Y = M(S - T) - V$$

$$Z_3 = 2Y_1Z_1$$

$$aZ_3 = 2U(aZ_1^4)$$

As an estimation of computational complexity PADD require 9 large integer and 5 squaring where as PDBL requires .

In order to reduce computational complexity of scalar multiplication we employ two new techniques: i) Reducing the Hamming Weight of scalars and (ii) using sliding window. We discuss both the methods in next two subsections.

4.2. Reducing Hamming Weight of Scalars

The Hamming weight of a string is the number of symbols that are different from the zero-symbol of the alphabet used. Hamming weight is the number of "1" bits in the binary sequence. Looking to computational complexity, we observe that PADD is more expensive than PDBL. Since point multiplication can be decomposed to series of PADD and PDBL, it is obvious to use more PDBL than PADD to compute the point multiplication.

NAFs is an effective way to achieve the least Hamming weight for scalar *k* in point multiplication *kP*, Morain et al [ ], which accounts for the least number of point additions to calculate *kP*.As an example consider *k*= 511= $(111111111)_2$. Now for any point *P*, 511*P* requires 8 point additions. However, if we write $(111111111)_2 = (100000000)_2 - 1$

Then $511P = 512P - P = (100000000)_2 P - P$, requires only one addition. Note that point subtraction can be replaced by PADD because the inverse of an affine

point $P = (x, y)$ is $-P = (x, -y).$ Thus we can implement it in point multiplication to achieve faster results.In another approach we consider recoding of scalar k as compliment of 1 in binary arithmetic. Let N be a number in binary form ,b be number of bits in the binary form of N. Then compliment of N in 1 is given by following rule

$$C = (2^b - 1) - N \text{ or equivalently}$$

$N = 2^b - 1 - C$ where C is 1's

complement of the binary number .

As an example consider the number 2148

$N = 1948 = (1111001110\ \ 0)_2$, then

C=1's complement of the number N is

equal to $(00001100011)_2$.

It can easily be verified that

$$\begin{aligned} 1948 &= 2^{12} - (0000110001\ \ 1)_2 - 1 \\ &= 2048 - (64 + 32 + 2 + 1) - 1 \\ &= 2048 - 99 - 1 \\ &= 1948 \end{aligned}$$

The Hamming weight of 1948 has reduced rom 7 to 4 which will save 3 EC additions.Since one addition operation requires 2 squaring, 2 multiplication and 1 inverse operation , a total of 2 squaring , 6 multiplication and 3 inverse operations will be saved.

4.3 Sliding Window

Sliding windows, a technique also known as *windowing*, is used by the Internet's Transmission Control Protocol (TCP) as a method of controlling the flow of packets between two computers or network hosts. Here we consider sliding window for scalar multiplication in ECC.

With the help of Sliding Window Algorithm we can compute kP for any $k \in Z^+$ (set of positive integers). Let *P* be a point on EC and let *k*=3277 then 3277*P* is equivalent to $(110011001101)_2 P$. We consider window size from 2 to 12, however it can be of any size. For a window of size *w*, there are $2^{w-1} - 1$ pre computations . For example for w=3 there are, $(2^{3-1} - 1) = 3$ precomputations, namely 3P, 5P and 7P. One can observe that as the window size increases the number of pre computation increases but on the other hand it has been observed that number of additions and doubling operations decreases.

The number of PADD and PDBL for 3277P for different window size are given below and the same is summarized in Table-1.

1.Window size=2

k=3277

The intermediate values for calculating 3277P are

P,2P,4P,5P,10P,11P,12P,24P,48P,51P,102P,204P,408P,816P,819P,1638P3276P,3277P.

2. Window Size=3

The intermediate values for calculating 3277P are

5P,10P,20P,40P,45P,50P,100P,200P,400P,407P,814P,817P,1634P,3268P,3275P,3276P, 3277P.

3. Window Size=4

The intermediate values for calculating 3277P are

9P,18P,36P,72P,87P,100P,200P,400P,800P,815P,1630P,3260P,3273P,3274P,3277P

4. Window Size=5

The intermediate values for calculating 3277P are

5P,50P,100P,200P,400P,800P,819P,1638P,3276P,3277P.

5.Window Size=6

The intermediate values for calculating 3277P are

51P,100P,200P,400P,800P, 819P,1638P,3276P,3277P

6.Window Size=7

The intermediate values for calculating 3277P are

101P,202P,404P,808P,815P,1630P,3260P, 3277P.

7. Window Size=8

The intermediate values for calculating 3277P are

203P, 406P, 812P, 1624P, 3248P, 3277P.

8. Window Size=9

The intermediate values for calculating 3277P are

409P, 818P, 1636P, 3272P, 3277P.

9. Window Size=10

The intermediate values for calculating 3277P are

819P, 1638P, 3276P, 3277P.

10. Window Size=11

The intermediate values for calculating 3277P are

1637P, 3274P, 3277P.

11. Window Size=12

The intermediate values for calculating 3277P are

3277P.

Window Size Vs No. of Computational costs.

Table .1.

Window Size	No. of Doublings	No. of Additions	No. of Pre computation
2	11	6	1
3	9	7	3
4	8	6	7
5	7	2	15
6	5	3	31
7	5	2	63
8	4	1	127
9	3	1	255
10	2	1	511
11	1	1	1023
12	0	0	2047

The above data is represented more elegantly in the following Fig.2.

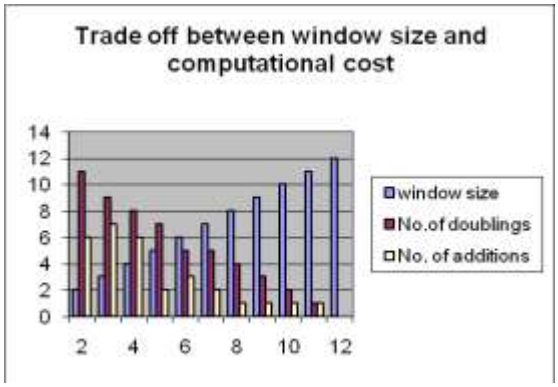

Fig.2.

It is obvious from the above results, that one has to select a suitable window size for optimal implementation of scalar multiplication, which varies with k- also. In our example the calculate the computational cost , the cost of precomputations is to be included also and in this example a window size w =5 provides optimum results.

5. CONCLUSION

In implementing arithmetic operations in ECC, the most important one is Scalar Multiplication. In this paper we have suggested two methods (i) reducing the Hamming weight of scalars of binary representation (ii) using Sliding window. The implementation of both of these methods reduces number of PADD's in scalar multiplication.

6. REFERENCES

[1].D. Djenouri, L. Khelladi,2005.A Survey of Security Issues in Mobile Ad Hoc and Sensor Networks," IEEE Communication Surveys and Tutorials, vol. 7, no. 4, pp. 2–28.

[2]. N. Gura, A. Patel, A. Wander, H. Eberle and S. C. Shantz,2004.Comparing elliptic curve cryptography and RSA on 8-bit CPUs, In "Proceedings of Workshop on Cryptographic Hardware and Embedded Systems (CHES'04)," Springer, 119–132.

[3].H. Wang, B. Sheng and Q. Li, Elliptic curve cryptography-based access control in sensor networks," International Journal of Security and Networks, Vol. 1, Nos.

[4]. V. Miller.1985 Uses of Elliptic Curve cryptography. In advances of Cryptography ,LNCS 218,pp-417-426.

[5]. N. Koblitz. 1987.Elliptic Curve Cryptosystems. Mathematics of Computation, Vol. 48,No. 177, pp. 203-209.

[6]. D. Hankerson, J.L.Hernandez and A. Menezes,2000. Software Implementation Of Elliptic Curve Cryptography Over Binary fields. Cryptographic hardware and Embedded Systems, CHES 2000, LNCS 1965, Springer-Verlag, 1-24.

[7]. R. Schroeppel, C. Beaver, R. Gonzales, R.Miller, and O. Spatscheck.1995.Fast Key Exchange with Elliptic Curve Systems. Advances in Cryptology – Crypto '95, LNCS 963, Springer-Verlang, 43-56.

[8]. J. Lopez, R. Dahab.1999. Fast Multiplication on elliptic curves over GF(2m) without Precomputation. In Cryptographic Hardware and Embedded Systems – CHES'99, volume 1717 of Lecture Notes of Computer Science, pages 316 – 327.Springer Verlag .

[9]. J. L´opez and R. Dahab,2000. High-speed Software Multiplication in F2m. Indocrypt 2000, LNCS 1977, Springer-Verlag, 203-212.

[10]. J. L´opez and R. Dahab.1999Improved Algorithms for Elliptic Curve Arithmetic in GF(2n*)*. Selected Areas in Crytography - *SAC '98*, LNCS 1556, Springer-Verlag,201-212.

[11]. K. Koyama and Y. Tsuruoka.1993 Speeding up elliptic curve cryptosystems by using a signed binary window method. Advances in Cryptology – Crypto '92, LNCS, 740, Springer-Verlag, 345-357.

[12]. C. Lim and P. Lee.1994. More flexible exponentiation with precomputation. Advances in Cryptology - Crypto '94, LNCS 839, Springer-Verlag, 95-107.

[13]. J. A. Solinas.2000 Efficient Arithmetic on Koblitz Curves. Designs, Codes and Cryptography, 19(2/3), 195-249.

[14].S. Shantz,2000. From Euclid's GCD to Montgomery Multiplication to the Great Divide, preprint.

[15]. COHEN, H.,MIYAJI, A., ANDONO, T. 1998. Efficient elliptic curve exponentiation using mixed coordinates. In Proceedings of the International Conference on the Theory and Applications of Cryptology and Information Security (ASIACRYPT'98). Springer-Verlag, London, UK, 51–65.

[16]. MALAN, D. J.,WELSH, M., AND SMITH,M. D. 2004b. A public-key infrastructure for key distribution in TinyOS based on elliptic curve cryptography. In Proceedings of the 1st IEEE International Conference on Sensor and Ad Hoc Communications and Networks. Santa Clara, CA. October.

[17]. DIFFIE, W. AND HELLMAN, M. E. 1976. New directions in cryptography. IEEE Trans. Inf. Theory. *IT-22*, 6, 644–654.