Journal of Theoretical and Applied Information Technology
10th April 2013. Vol. 50 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

IMPROVED THE EXECUTION SPEED OF ECDSA OVER GF(2^N) ALGORITHM FOR CONCURRENT COMPUTATION

[1]A. SAKTHIVEL, [2]R. NEDUNCHEZHIAN

[1]Asstt Prof., Department of Information Technology, Adithya Institute of Technology, Coimbatore, India

[2]Prof., Department of Information Technology, Sri Ramakrishna Engineering College, Coimbatore, India

E-mail: [1]asakthivel75@yahoo.com, [2]rachezhian@yahoo.co.in

ABSTRACT

Recently Elliptic Curve Digital Signature Standard (ECDSA) is used for smart card applications and financial transaction applications, because it needs less memory space, low bandwidth, limited power, reuse and portability. So it is more suitable for wireless network applications. Normally it is implemented in two ways called as ECDSA over binary field GF (2^n) for hardware applications and ECDSA over prime field GF (p) for software applications. It has two main computations. One is point addition and another point multiplication. In these operations, the point multiplication takes more execution time than point addition. Because the latency time of point multiplication is higher than point addition. So it is necessary to find out optimized implementation for point multiplication. This article suggests a technique for point multiplication over GF(2^n) to increase the speed of the execution using concurrent computation.

Keywords: *ECDSA, Binary Field, Time complexity, latency time, parallel computation.*

1. INTRODUCTION

The NIST (National Institute of Standards and Technology) has published a standard FIPS 186-v3 known as Federal Information Processing Standard 186-3 version for Digital Signature Standard (DSS) applications in 2009. This document standardized two approaches for DSS which are RSA (Rivast-Shamir-Adleman) and ECC (Elliptic Curve Cryptography) based DSS. The RSA based DSS is not suitable for wireless and smart card applications, because it needs more power for large prime number manipulation and ECC offer a much shorter key length than RSA. For example, the RSA uses 1024,2048,3072,7680 or 15360 bits of key length for encryption to provide security level, and then ECC uses only 161,224,256,384 or 512 key sizes accordingly to provide same security level. In wireless and smart card environments, the 1024-bit RSA cannot be implemented, while 163-bit ECC can be implemented [6].

The ECC based DSS is also called as Elliptic Curve Cryptography Digital Signature algorithm (ECDSA). It defines two important procedures for digital signature generation and verification based on domain parameters. This provides Authentication, Integrity and non reputation security services to avoid passive and active attacks on network [3]. This ECDSA is implemented in two different ways by using Finite Field (FF) or Galois Field (GF). One is called as ECDSA over prime field GF(p) and another is called as ECDSA over binary field GF (2^m). The ECDSA over prime field is used for software application such as online financial transactions and commercial applications and another type for hardware applications such as designing circuits for credit cards, smart cards and commercial authentication's cards. This article mainly focuses on ECDSA over binary field.

ECDSA algorithm has some limitations such as lack of implementations and low latency time of point operations. It is mainly dependent on two important point operations called as point addition and point multiplication. The point addition is a basic operation which needs only limited number of clock cycles and its latency time is low. But the point multiplication needs more number of clock cycles for its latency time [4]. So this paper suggests a software scheduling technique to reduce the dependent operation into independent operations for parallel processing. When the ECC is used in software scheduling for hardware execution, it reduces number of clock pulses and latency time of point multiplication operations in order to increase the speed of algorithm [1].

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

The section-II gives an introduction to finite field and overview of ECDSA over 2^n and subsequently the section-III concludes the literature survey of previous methods. The Section-IV describes a suggested implementation for ECDSA. This innovative technique is analysed and compared with existing ECDSA as given in section-V. Then the section-VI concludes with some of applications which may be implemented by using the proposed methodology. Finally the section VII acknowledges to previous contributors on ECDSA.

2. BASIC CONCEPTS FOR ECDSA

2.1 Finite Field over GF(2^n)

Theory of Finite Field is a main part of mathematical theory in cryptology which is used to implement ECC over 2^n. This is defined with the help of Abelain group, commutative ring and fields [11].

Theorem-1: An abelain group (G,*) is a set of elements over operation * (assumed as +) with the following group laws and commutative law.

1. a+b ∈ G
2. (a+b)+c=a+(b+c)∈G
3. a+(-a)=(-a)+a=0∈G
4. a+0=0+a=a∈G
5. a+b=b+a∈G where a,b,c,0,-a,-b,-c∈G

Theorem-2: A commutative ring (R,+,×) is an algebraic structure of G with two operations called addition and multiplication with the following rules.

1.a×b∈G ∈G
2.(a×b)×c=a×(b×c)∈G
3. (a+b)×c=(a×c)+(b×c)∈G
4. a×b=b×a. ∈G

Theorem-3: A Field is defined by using a commutative ring under the following three rules.

1. a×1=1×a=a ∈G
2. a×b=0∈G than either a=0 or b=0 ∈G
3. $a\times a^{-1}=a^{-1}\times a=1$ a and $a^{-1}\in G$

Theorem 4: Finally a finite field GF(2^n) consists of 2^n elements together with addition and multiplication. They are determined based on reducible and irreducible polynomials [7]. It means that

• GF (2^n) → contains 2^n elements and they are irreducible polynomials (generator polynomial) which are identified from reducible polynomial.

Elliptic Curve Cryptography is an algebraic structure of elliptic curves over finite fields [11]. It is defined by using polynomial basis and finite field. It mainly has a set of points over point addition and multiplication.

2.2 ECDSA over GF (2^n)

For example, the general Weierstrass Equation is considered for defining ECC over 2^n:

$$y^2+a_1xy+a_3y = x^3+a_2x^2+a_4x+a_5 \quad (1)$$

where a_1, a_3, a_2, a_4 *and* a_5 *are* co-efficient and x & y are variables [7]. And this Equation is simplified as the following to compute points on elliptic curve rapidly.

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

In this Equation, the value of x and y are calculated based on a and b , to define points on Elliptic Curve. The point has positive and negative co-ordinate values. The negative values are redefined by using the following constraint (3).

$$\Delta=4a^3+27b^2 \text{ modulo } p \neq 0 \bmod p$$
where p is a prime
and it is denoted by $E_2^n(a,b)$ (3)

These sets of points are manipulated by using point addition and point multiplication [11]. The point addition is a basic operation which needs a limited clock cycles. But the importance of ECC is to compute the point multiplication $Q=kP$, where k is a scalar value and P is a point on the elliptic curve. It needs more clock cycles to compute its result and is implemented by using point addition and point doubling [7].

If $P=(x_1, y_1)$ and $Q=(0, 0)$ then
$R=P+Q=P$ where O is origin point. (4)

If $P=(x_1, y_1)$ and $Q=(x_1,-y_1)$ then
$R=P+Q=O$ where -P is inverse of P. (5)

If $P=(x_1, y_1)$ and $Q=(x_2, y_2)$ then $R=P+Q=(x_3, y_3)$ where $x_3=\lambda^2+\lambda+x_1+x_2+a$ and $y_3=\lambda(x_1+x_3)+x_3+y_1$.

$\lambda = (y_2+y_1)/(x_2+x_1)$ if$(P \neq Q)$ (6)

Journal of Theoretical and Applied Information Technology
10th April 2013. Vol. 50 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

$\lambda = (x_{1+}y_1/ x_1)$ if (P = Q) (7)

In this case point addition is an atomic operation and a point multiplication is a complex operation of ECC to compute Q=kP [10]. The k denotes a scalar value and P,Q are points on the EC. ECDSS is an algebraic structure of elliptic curves over finite fields [3]. It has three main concepts called as digital signing, digital verifying and key generation and the procedure for each concept is explained as follows [6].

Key generation:

1. Define $E_2^n(a,b)$ where a,b→ variable where p is irreducible polynomial in the form of 2^n
2. Select another irreducible polynomial q where q is smaller than p.
3. Choose a binary integer d as private key.
4. Choose $e_1(x,y)$) form EC point set.
5. Find $e_2(x, y)=d\times e_1(x, y)$.
6. User private key is d and public key is (e_1,e_2,p,q,a,b) where e_1 and e_2 polynomial generators.

Signing on Sender side:

1. choose random binary number r where 1<r<q-1
2. Find $P(x,y)=r\times e_1(x,y)$
3. Find S_1=x (modulo) q from P
4. Find $S_2=(h(M)+d\times S_1)r^{-1}$(modulo)q
5. Send M,S_1 and S_2 to receiver side

Verifying on Receiver Side:

1. Calculate $A=h(M)S_2^{-1}$(modulo)q
2. Calculate $B=S_2^{-1}S_1$(modulo)q
3. Find $T(x,y)=A\times e_1(x,y)+B\times e_2(x,y)$
4. Is $x=S_1$ (modulo)q than it is verified otherwise it is rejected.

All of these procedures are based on point multiplication and point addition. This point multiplication is implemented by using point addition and point doubling based on Equation (6) and Equation (7) respectively. One of the common way of implementing the point multiplication is linear scalar point multiplication as shown in Equation (8) and the procedure is as follows [5].

$kP=P+2P+3P+\ldots+(k-1)P+kP$ (8)

(k times of Point addition)

The diagram representation of point multiplication by using Equation (1) is shown in Figure 1.

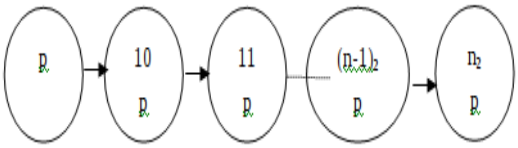

Figure 1. Linear point multiplication of kP

procedure linearmultiplication
(Point P,Integer k)

(1) Integer I, Q_0 =(0,0)
(2) I=1
(3) if(I≠k)
  (3.1) compare P with Q_0.
  (3.2) compute and update P and Q_0 by using Equations (4),(5),(6) or (7)
  (3.3) I=I+1 goto step 3.
(4) return P→ kP.

3. LITERATURE SURVEY

There are four types of dependences existing in the linear point multiplications called as register value (data), register name, control and loop carried dependences [12]. In this case, Name dependence means that two or more points refer the same register. And register value (data) decencies means a point in a register is dependent on another point in a register. A Control dependence determines the way of predicting points based on four constraints which are mentioned in Equations (4), (5), (6) and (7). A loop-carried dependence is used to check current point computation whether it depends on later or earlier iteration's computation or not. In linear multiplication, the same point is updated in each iteration called as name dependence and a point is computed by using point addition or point doubling known data dependence. Then a point computation is always depends on point addition or point doubling called as control dependence and a point multiplication always depends on previous iterations known as loop carried dependence.

The next left to right or right to left binary methodologies process a loop scanning of scalar bits and performing a point doubling followed by a point addition based on scalar bit value (equals 1) [10]. These binary algorithms involve n point doubling and n/2 point additions. The main weakness of this methodology is side channel attack. The scalar bits are traceable based on power analysis. The above mentioned control dependences are also there in this methodology. But these algorithms are only suitable for special type of elliptic curve such as Montgomery curve [8].

Journal of Theoretical and Applied Information Technology
10th April 2013. Vol. 50 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Procedure MogRL(Point P, Integer k $(k_{n-1}, \ldots, k_0)_2$)

1. $R_0 \leftarrow P$, $R_1 \leftarrow P$ and $i \leftarrow n-2$.
2. $R_0 \leftarrow 2R_0$.
3. if $k_i = 1$ then
   (a) $R_0 \leftarrow R_0 + R_1$.
   (b) I←one time shift right of i go to step 3.
4. return R_0 .

Procedure MogLR(Point P, Integer k $(k_{n-1}, \ldots, k_0)_2$)

1. $R_0 \leftarrow O$, $R_1 \leftarrow P$ and $i \leftarrow 1$.
2. if $k_i = 1$ then $R_0 \leftarrow R_0 + R_1$
3. $R_1 \leftarrow 2R_1$.
4. I← one time of shift left of I go to step 2.
5. return R_0.

The point multiplication called as regular binary algorithm which is defined by Jacobian requires more field registers such as 7 or 6 [9]. These field register is used to store additional bits value for kP computation. In this case, the numbers of inversion operations are reduced. But the number of data dependences, register dependences, control dependences are increased. Hence there is no change loop-carried dependence.

The literature survey shows that there is no optimal implementation of scalar multiplication for parallel processing. So this paper suggests a technique to solve the problem of optimizing point multiplication operations over parallel processing. The suggested innovative technique uses a divide and conquer algorithm [10]. And it calls by iteratively to break down multiplication into two or more sub-problems until these become simple to compute directly by using point addition or doubling. Finally, these solutions are combined to find out the required computation of point multiplication [2].

4. SUGGESTED METHODOLOGY

In this proposed point multiplication, the Equation (7) is used to create binary tree and skew tree based on k value. In this case each node assumed is as a point value. Finally the summing of skew tree node value and binary tree node value computes kP value. The k value is assumed as 1111. When the k value is divided by 2 in every time, quotient values 111, 11, 1 and remainder values 1,1,1 are obtained. The binary tree is created by using quotient values and points are used to compute point doubling operation based on the formula (6). And subsequently the skew tree is also formed by using reminder value and points are used to compute point addition and point doubling operations based on the formula (7). Finally these two trees are summed by using formula (4),(5),(6) or (7) to compute point doubling for kP and diagrammatically shown in Figure 2(a) and Figure 2(b). The corresponding algorithms of kP computation 3(a), 3(b), 3(c) and 3(d) are as follows.

Algorithm. 3(a) PointCompute
Input: Point P, Integer k.
Output: kP
Point P_1=(0,0), P_2=(0,0);
1. If k=1 then
   1.1 One time of P
2. If k>1 then
   2.1 Q←one time of shift left k
   2.2 if (Q>0) then
      2.2.1 P_1=call PointMulBinary(Point P)
      2.2.2 P=P_1
   2.3 R←one time of shift left k
   2.4 if (R=1) then
      2.4.1 P_2=call PointMulSkew(Point P)
      2.4.2 P=P_2
3. k=one time of shift left k and goto step 2.
4. call PointSummazation(Point P_1,Point P_2)

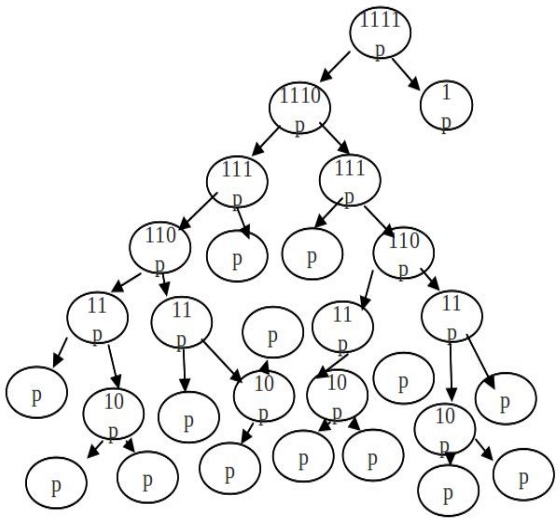

Figure. 2(a). The proposed Point multiplication of kP by using divide processes.

Algorithm. 3(b) PointMulBinary
Input: Point P, Integer Q.
Output: kP
Point Sum=(0,0).

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

1.SUM=P+SUM based on Equation (6).
2. kP=SUM

Algorithm. 3(c) PointMulskew
Input: Point P, Integer R.
Output: kP
Point Sum=(0,0).
1.SUM=P+SUM based on Equation (7).
2. kP=SUM

Algorithm. 3(d) PointSummazation
Input: Point P_1,P_2.
Output: kP
Point Sum=(0,0).
1. SUM=P_1+P_2 based on Equations (4), (5), (6) or (7).
2. kP=SUM.

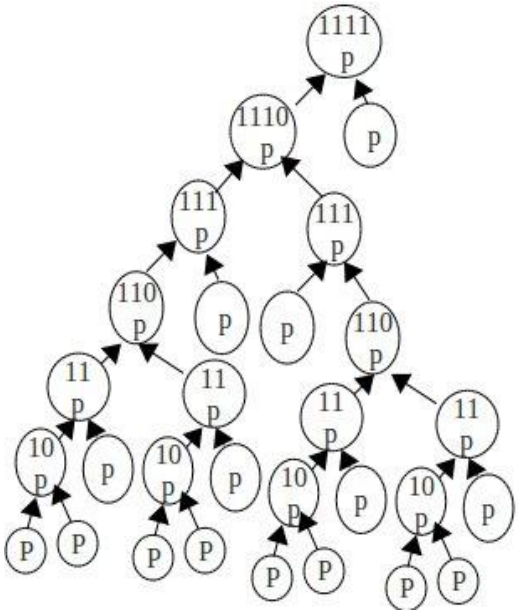

Figure. 2(b). The Proposed Point multiplications of kP by using conquer processes.

This methodology minimizes number of loop carried dependences, register value (data) dependences and predictor (control) dependence which help to improve software scheduling for hardware. And also it reduces the number of hazards and stalls at the time of execution. But the linear and other binary scalar multiplications are always dependent on the early iteration known as loop carried dependence. The Data dependence creates N-1 times of hazards and stalls to affect loop level parallelism. If it is not optimized, it will affect the performance of the computing

5. EXPERIMENTAL RESULT ANALYSIS

The $y^2=x^3+ax+b$ Equation is assumed to support a set of points on Elliptic Curve for experimentation. Then $E_2^n(a,b)$ is defined by using $E_2^4(g^4,g^2)$. The point $(x=g^5,y=g^3)$ is taken from the set to compute point multiplication for both linear scalar and the proposed binary tree multiplication as shown in the Table 1.

The mathematical study of the difficulty of a mathematical problem which describes the resources required by a computing machine to solve the problem is called computational complexity theory and is important in many branches of theoretical computer science, especially cryptography. In this case, computational complexity of point multiplication is analyzed by using the time complexity of execution in terms of clock pulses. In proposed case, there are three cases to analyze point doubling by using k value.

Table 1: Shows that different parameter value needed for simulating Point multiplication of kP

Parameter	Type	Value
E_2^n	Input	E_2^4
a	Input	g^4
b	Input	g^2
x	Input	g^5
y	Input	g^3
P	Input	(x,y)
K	Input	Number of times(N)
k_i	Input	$0<N<2^i$
kP	Output	Point multiplication Value
kP execution time	Output	Number of clock pulses

First Case is called as best case. The best case time complexity is defined by the way of an algorithm behaves under optimal conditions. There is no remainder of k value for all iterations. It means that k = 2N where N>0. Because it takes only $\log_2 N$ times to compute kP based on Quotient point computation and no need to compute skew tree computation. It is denoted by $O(\log_2 N)$.

Second case is called as worst case and it is defined the way of an algorithm behaves under all possible conditions. There is a remainder and quotient values of k for all iteration. So the k value is in the form of 2^N-1, where N>0. The k value takes only $\log_2 N$ times to compute kP based on quotient point computation, as well as remainder

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

point computation. So the time complexity of this case is defined by $O(\log_2 N)+ O(\log_2 N)$ times.

Third case is called as average case which the way algorithm of an algorithm acts under the probability of execution. There is a remainder of k values for some iterations and no remainder for some other iterations. The k value is in 2^N or 2^N-1. The computation time of this case is defined by $\log_2 N+\text{Prob}\{\log_2 N\}$ times and its denoted by $O(\log_2 N)+O(\text{Prob}\{\log_2 N\})$.

All time complexities are redefined by $\log_2 N+1$, $2\log_2 N+1$ and $\log_2 N+\text{Prob}\{\log_2 N\}+1$. The value 1 denotes the final computation of combining quotient point computation and remainder point computation to compute kP.

The best case and worst case of proposed methodologies are compared with linear scalar point multiplication. Both are simulated computation values are measured in terms of clock pulses as shown Table-2.

Table 2. Shows that the number of clock cycles needed to compute kP using linear multiplication and proposed point multiplication for both (best case and worst case) based on various k value and P in the unit of seconds.

calculates kP		Clock pulses		
i	2^i	Best	Worst	Linear
2	100	1	1	3
3	1000	2	3	4
4	10000	3	5	6
5	100000	4	6	13
6	1000000	5	7	24
7	10000000	6	9	46
8	100000000	6	10	91
9	1000000000	7	11	191
10	10000000000	7	11	380
11	100000000000	8	12	748
12	1000000000000	9	13	1503
13	10000000000000	9	14	3054

Then the proposed method is analyzed with graph shown in Figure 3(a) and Figure 3(b). In this graph, the y-axis denotes k values in the form of 2^i and x axis total number of execution time for kP.

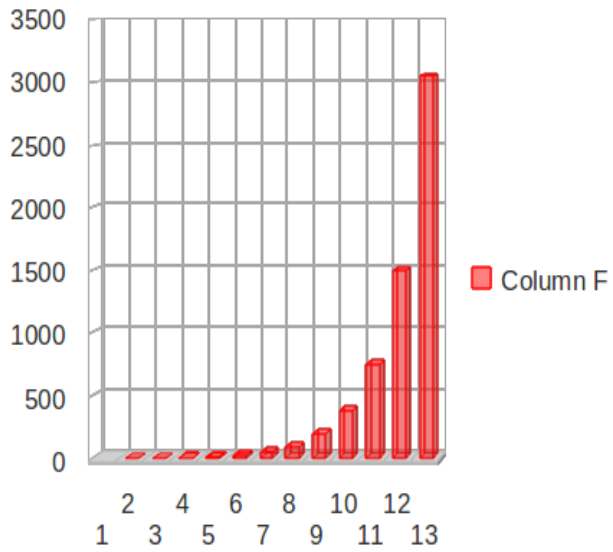

Figure. 3(a). This Graph shows that the total amount of execution time needed to compute kP for Liner point multiplication.

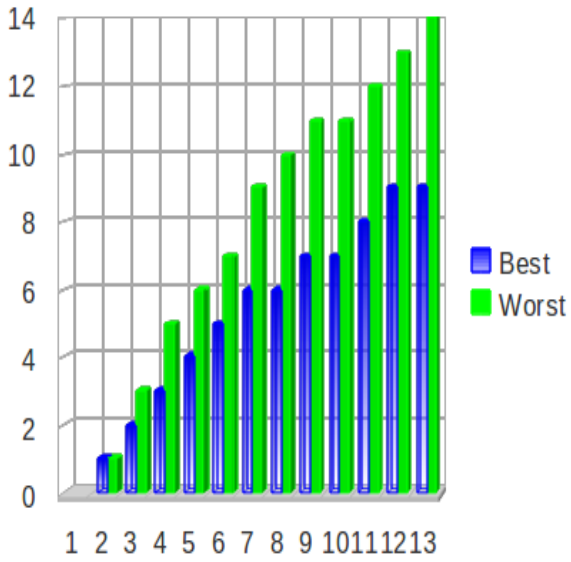

Figure. 3(b). This graph compares that the total amount time needed to compute kp based on best & worst cases of point multiplication.

The proposed strategy will reduce number of dependence into $\log_2 N+1$ for best case and $2\log_2 N+1$ for worst case dependent operations. Beside the proposed technique is useful to minimize the number of data dependences, control dependences and loop carried dependences.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

6. CONCLUSION AND APPLICATION

The ECDSA uses linear scalar point multiplication to perform digital signature signing, verification and key generation. The consequence of using this approach takes more time for execution, because these algorithms are implemented in linear fashion. But the proposed methodology which is implemented by using polynomial basis decreases number of clock pulses and power consumption for computing point multiplication and also increases the performance of ECDSA through for creating possibility to support parallel computation. It is also to utilize hardware units for minimum number of times to find out point multiplication. So it increases the life time of hardware units. Hence it is more suitable for smart card based application and wireless based authentication's operation to perform financial transactions and commercial applications using embedded devices.

ACKNOWLEDGMENT

The authors would like to thank Adithya Institute of Technology and its Research Centre for Information Security and Cryptography to do this work.

REFRENCES:

[1] Lo'ai Tawalbeh, Yaser Jararweh, and Abidalrahman Mohammad, "An Integrated Radix-4 Modular Divider/Multiplier Hardware Architecture for Cryptographic Applications", *The International Arab Journal of Information Technology*, Vol. 9, No. 3, May 2012.

[2] Adnan Abdul-Aziz Gutu, "Preference of Efficient Architectures for GF(p) Elliptic Curve Crypto Operations using Multiple Parallel Multipliers", *International Journal of Security (IJS)*, Volume (4) : Issue (4) , pp.46-63. Feb. 2010.

[3] Arash Reyhani-Masoleh,"Efficient Algorithms and Architectures for Field Multiplication Using Gaussian Normal Bases", *IEEE Transactions on computers*, vol. 55,NO. 1,pp34-48, Jan 2006.

[4] Patrick Longa and Ali Miri, "Fast and Flexible Elliptic Curve Cryptography point arithmetic over Prime fields", *IEEE Transactions on computers*, vol.57,No.3, pp.289-302, May 2008.

[5] Pradeep Kumar Mishra, "Pipelined Computation of Scalar Multiplication in Elliptic Curve Cryptosystems", *IEEE Transactions on computers*, vol 55,No.8, pp1000,Aug 2006.

[6] Gary Locke and Patrick Gallagher, "Federal Information Processing Standards Publication Digital Signature Standard (FIPS PUB 186-3)", *Information Technology Laboratory, National Institute of Standards and Technology*, June, 2009.

[7] Sangook Moon, Jaemin Park and Yongsurk Lee, "Fast VLSI Arithmetic Algorithms for High-Security Elliptic Curve Cryptographic Applications", *IEEE Transactions on Consumer Electronics*,Vol.47,No.3,pp. 700-708, Aug 2001.

[8] E. Savas and C.K. Koc, "The Montgomery Modular Inverse- Revisited", *IEEE Transactions on Consumer Electronics*, vol. 49, no. 7, pp 763-767, July 2000.

[9] Kenny Fong, Darrel Hankerson, Julio Lopez, and Alfred Menezes, "Field Inversion and Point Halving Revisited", *IEEE Transactions on Consumer Electronics*, vol. 53, no. 8,pp 1047 Aug 2004.

[10] Xiaoyu Ruan, and Rajendra S. Katti, "Left-to-Right Optimal Signed-Binary Representation of a Pair of Integers", *IEEE Transactions on computers*, vol 54,no.2, pp 124,Feb 2005.

[11] Erkay Savas and Çetin Kaya Koç "Finite Field Arithmetic for Cryptography," *IEEE Circuuts and Systems Magazine*, Digital Object Identifier 10.1109/MCAS.2010.936785, pp 40-57, Second Quarter 2010.

[12] John L.Hennessy & David A. Pattersom, "Computer Architecture a Quantitative Approach", *Elsevier*, 4th Edition, 2007.