Available online at www.elixirpublishers.com (Elixir International Journal)

Electrical Engineering

Elixir Elec. Engg. 59 (2013) 15913-15918

ISSN: 2229-712X

Analyzing the effectiveness of a new technique to speed up scalar point multiplication over GF(p) by random numbers and statistics

Anil Kumar M. N* and V. Sridhar

Department of E&C, PET Research Foundation, PESCE, Mandya.

ARTICLE INFO

Article history:

Received: 13 May 2013;

Received in revised form:

12 June 2013;

Accepted: 22 June 2013;

Keywords

Random number generators,

Elliptic curve cryptography,

Binary Inversion Algorithm,

GF(p) arithmetic operators.

ABSTRACT

In this paper we present a slightly modified Binary Inversion Algorithm (BIA) to speed up scalar point multiplication of National Institute of Standards and Technology (NIST) recommended elliptic curve with moduli 2^{521} -1. The effectiveness of the above method is mathematically analyzed by using statistical analysis and by computing series of scalar point multiplications Q= k.P for randomly generated k. Our method uses 2 random generators for generating 'k' namely, Random number generator based on Elliptic Curve Operations and Blum-Blum-Shub Generator. The mathematical analysis shows that above technique speeds up the inversion operation and consequently the scalar point multiplication of the above NIST recommended curve. New architecture based on the modified BIA is proposed. The results show that the modified BIA can be implemented in the hardware to speed up the scalar point multiplication.

Introduction

The data security, authentication and integrity has become an important and urgent need for health care information, confidential communication, storage and financial services etc. The public key cryptosystem is the most efficient way to secure data transaction and messaging. The challenge to implement the most popular public key cryptosystem, RSA is the rapidly growing key size. Elliptic Curve cryptography has been considered an alternative to RSA. A lot of implementations have been reported in [1-5]. The effectiveness of using elliptic curve is that it provides same security level with shorter keys than in RSA. Therefore ECC can be used in smart cards, credit cards and mobile phones where area is a constraint. It is estimated that security level of 160 and 224 bits ECC cryptosystem is equivalent to the 1024 and 2048 bits RSA respectively. The research on different algorithms and hardware accelerations have focussed on efficient implementation of elliptic curve scalar point multiplication Q=k.P. This is the fundamental operation of all elliptic curve cryptosystems.

Two types of Finite Fields are generally used in ECC. Those are Finite Field over a large prime called as *Galois Field* GF(p) and Extended Binary Field that is known as *Galois Field* $GF(2^k)$. A very few hardware implementations of ECC on GF(p) have been reported in the literatures compared to implementations on $GF(2^k)$ [6-10]. A low power flexible GF(p) ECC processor has been reported in [11] which is suitable for RFID tags ,wireless sensors and smart cards. A flexible ECC processor over GF(p) has been reported in [12] which supports all five NIST primes with size ranges from 192 to 521. They have used NAF scalar multiplication algorithm and BIA to compute the inversion. [13] has reported Dual field processors and the design framework for ECC by using mixed projective-affine coordinates which replaces the field inversion and optimization with different area/throughput requirements. Parallelization of high speed ECC accelerators have been studied in [14]. A hardware architecture for ECC over GF(p) has been reported in [15] with a new unified modular inversion algorithm instead of Fermat's Little Theorem.

The hardware complexity to implement ECC in GF(p) is little bit higher than that of in $GF(2^k)$ but the advantage is that the k-bit arithmetic unit is capable to process any i-bit data where $1 \le i \le k$. The arithmetic operations of GF(p) can be performed faster than $GF(2^k)$ with the instructions of general purpose microprocessors. Confining designs to binary fields limit the flexibility and may not be used for Elliptic Curve Digital Signature Algorithm. This algorithm in addition to EC point operation is based on normal integer modulo operations. For binary field designs these modulo operations must be done separately by using a processor or in a separate hardware. Inversion is the costliest operation among all the modular operations. Inversion operation can be eliminated with projective coordinate systems with the cost of using parallel multipliers [13-14]. But in small devices like smart cards where area is a constraint, adding more multiplier units needs more memory and thus increases the cost. Speeding up inversion operation in both fields has been gaining attention because inversion is the most time consuming operation when affine coordinates are selected.

In this paper we have modified the BIA over GF(p) to speed up the inverse computation and consequently scalar point multiplication of NIST recommended elliptic curve with moduli 2^{521} -1. The paper proposes a new architecture to meet the above objective. The rest of the paper is organized as follows. Section II provides a brief mathematical back ground of Elliptic Curve based cryptography and Random Generators. Sections III brief about the methodology of the proposed method and in section IV results are discussed and finally conclusion is given in section V.

II. ECC BACKGROUND

2.1 ECC Algorithms structure analysis

ECC algorithms are layered on four levels as shown in figure 1.

Tele:

E-mail addresses: anil_mn1@rediffmail.com

ECC Pprotocols
EC Point multiplication
EC Point addition and doubling
Finite field arithmetic: addition, subtraction, multiplication,squaring, inversion, reduction

Fig.1 Hierarchical Structure of the ECC algorithms

ECC Protocols: Top application includes key establishment, data encryption and decryption, signature verification schemes. Public key cryptography protocols employ point multiplication as a fundamental operation and security is based on difficulty of solving Elliptic Curve Discrete Logarithm Problem which is finding scalar k, given a point P, and k.P

EC point multiplication: This includes the computation of point multiplication Q=kP. The focus of research is the representation of 'k' which decides the number of point addition and point double operations performed during point multiplication. This contains different algorithms like Binary scalar multiplication algorithm, Montgomery method, Non Adjacent Form (NAF) scalar multiplication algorithm etc.

EC point addition and doubling. This layer has different point representation in affine coordinates, projective coordinates such as standard projective coordinates, Jacobian coordinates and Chundnovsky projective coordinates and mixed coordinates. There are lots of implementation using affine and projective coordinates and effect of parallelism with different coordinate systems with different arithmetic operations. These arithmetic operations include addition, subtraction, multiplication, reduction, squaring and inversion.

Draw back of Inversion in GF(p) using Extended Euclidean Algorithm is the requirement for computationally expensive division operations. In Binary Inversion Algorithm, the division operation is replaced with cheaper shifts, subtractions and additions making this algorithm suitable for implementation in hardware [16].Montgomery inversion algorithm is applicable if Montgomery arithmetic is used with affine coordinates .

NIST primes: The FIPS 186-2 standard recommends elliptic curves over the five prime fields with moduli:

$p192 = 2^{192} - 2^{64} - 1$

$p224 = 2^{224} - 2^{96} + 1$

$p256 = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$

$p384 = 2^{384} - 2^{128} - 2^{96} + 2^{32} - 1$

$p521 = 2^{521} - 1$

Except for p521, the powers appearing in these expressions are all multiple of 32. This property yield reduction algorithms especially fast on machines with word size 32.

2.2 Elliptic Curves Over GF (P). The elliptic curve arithmetic is defined over Galois field GF(p) where p is a prime number greater than 3. All arithmetic operations are modulo p. The elliptic curve equation E over GF(p) is given by :$y^2 = x^3 + ax + b$; where $p > 3$, $4a^3 + 27b^2 \neq 0$, and x, y, a,b$\in$ GF(p). There is also a single element named the point at infinity or the zero point denoted O, which serves as the additive identity. For any point $P(x, y) \in E$, we have: $P + O = P$.

2.2.1Point addition and Point Doubling

Additions in GF(p) are controlled by the following rules:

$O = -O$

$P(x, y) + O = P(x, y)$

$P(x, y) + P(x, -y) = O$

The addition of two different points on the elliptic curve is computed as shown below.

$$P(x_1 , y_1) + P(x_2 , y_2) = P(x_3 , y_3) \text{ ; where } x_1 \neq x_2$$

$\lambda = (y_2 - y_1)/(x_2 - x_1)$

$x_3 = \lambda^2 - x_1 - x_2$

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

The addition of a point to itself (point doubling) on the elliptic curve is computed as shown below

$P(x_1 , y_1) + P(x_1 , y_1) = P(x_3 , y_3)$;

$$\lambda = (3(x_1)^2 + a) /(2y_1)$$

$x_3 = \lambda^2 - 2x_1$

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

2.2.3 Point Multiplication

Scalar multiplication Q=k.P is the result of adding point P to itself (k-1) times

$$Q = k.P = P + P + \ldots\ldots + P.$$

(k-1 Times)

The binary method is the simplest and oldest efficient method for point multiplication. It is based on the binary expansion of k. The corresponding algorithm is shown in Fig.1.

INPUT: A point P and an integer k

OUTPUT: Q = k.P

1. Q←P

2. **For** j = L− 2… 1, 0

2.1 Q ← 2 Q

2.2 **IF** $k_j = 1$ **THEN** Q←Q + P

3. **RETURN** Q

Fig.2. Binary scalar multiplication algorithm

2.3 Random Number Generators

Random numbers play an important role in providing security for various applications. The ability to generate pseudorandom numbers is very important for the key generation in cryptographic applications. A C library for Empirical Testing of Random Number generators is available in [17]. A random number generator based on the addition of points on an elliptic curve over finite field is proposed in [18].This method uses the encryption block to perform random number generation thus saves the hardware cost, memory space and design time. The theoretical analysis show that periods of this generator is sufficiently long and up to 29% of gate counts can be saved compared to implementation of a separate random number generator. The generated sequences also have passed the FIPS 140-2 statistical tests. The Blum-Blum-Shub generator[19] also referred to as cryptographically secure pseudorandom bit generator which passes the next-bit test. It is a quadratic congruential method for generation of pseudorandom bits for cryptographic purposes. The most widely used technique proposed by Lehmer is linear congruential method. The selection of parameters are very important in the generation of pseudo random sequences.

III. Methodology

The multiplicative inverse property between the number 'a' and 'NOT a' over Mersenne's prime is stated below. When 'x' is the multiplicative inverse of 'a' over GF(p) where p is a Mersenne's prime, then the multiplicative inverse of 'NOTa' is the complemented multiplicative inverse of x (NOT x). Henceforth pair refers to input 'a' which belongs to set of numbers from 1 to (p-1)/2 only and 'NOTa' is its corresponding pair which belongs to set of numbers from p-1 to (p-1)/2 +1 only. Hence there are (p-1)/2 such pairs. Let 'A' denote the occurrence of input b='NOTa' (refer fig.4) which has less number of subtraction and addition operations in the modified BIA compared to normal BIA. Let 'B' denote the occurrence of input b='NOTa' which has more number of subtraction and addition operations in the modified BIA compared to normal BIA. The probability of event A is far greater than the probability of event B.

The above two properties are used to speed up the inverse computation of some of the numbers within the range (p-1)/2+1 to p-1. Instead of computing the inverse of those numbers in the above said range directly, we complemented the number, then computed the inverse and finally complemented the output. This is the technique used in the modified Binary Inversion Algorithm. This modified BIA can be easily implemented in the hardware (explained in the following section). We have assumed negligible delays of the two stage complement operation. The above technique is used in computation of the scalar point multiplication Q=k.P. The results showed better performance with modified BIA, with reduced number of additions and subtractions than with normal Binary Inversion Algorithm.

3.1 Modified Binary Inversion Algorithm.

Figure 3 and Figure 4 show the BIA and modified BIA respectively. The extended Euclidean algorithm uses the division operations to compute the inversion. The binary inversion algorithm replaces the divisions with cheaper shifts (divisions by 2) and subtractions. The modular multiplicative inverse b^{-1} mod p of an integer b exists if and only if b and p are relatively prime, that is gcd (b,p) =1.

INPUT: Prime p and b ∈ [1, p-1]
OUTPUT: b^{-1} mod p

1. u=b, v=p, x1=1, x2=0
2. while (u !=1 and v!=1) do
   2.1 while u is even do

2.1.1 u = u/2
2.1.2 if x1 is even then x1= x1/2
else x1= (x1+p)/2
2.1.3 end while
2.2 while v is even do
2.2.1 v= v/2
2.2.2 if x2 is even then x2= x2/2
else x2 = (x2+p)/2
2.2.3 end while
2.3 if u≥ v then u=u-v, x1=x1-x2
else v=v-u, x2=x2-x1
2.4 end while

Fig 3. Binary Inversion Algorithm

INPUT: Prime p and b∈ [1, p-1]
OUTPUT: b^{-1} mod p

If (MSB of INPUT b==1), b=NOT b

1 u=b, v=p, x1=1, x2=0
2 while (u !=1 and v!=1) do
2.1while u is even do
2.1.1 u = u/2
2.1.2 if x1 is even then x1= x1/2
else x1= (x1+p)/2
2.1.3 end while
2.2 while v is even do
2.2.1 v= v/2
2.2.2 if x2 is even then x2= x2/2
else x2 = (x2+p)/2
2.2.3 end while

2.3 if u≥ v then u=u-v, x1=x1-x2
else v=v-u, x2=x2-x1
2.4 end while

If (MSB of INPUT b==1),
x1=NOT x1,x2=NOT x2
end

Fig 4. Modified Binary Inversion Algorithm

In the modified algorithm, the most significant bit of INPUT b is checked. If it is 0 then the above algorithm works as normal BIA with u variable is assigned the value of b. The output is available in any one of the variables x1 or x2. If Most Significant Bit (MSB) is 1(input value with in the range p-1 to (p-1)/2 +1), then the variable u is assigned the complemented value of INPUT b. The output is available in any one of the complemented values of x1 or x2 variables.

Figure 5 shows the new architecture of modified Binary Inversion Algorithm. The detailed architecture of Binary Inversion Algorithm has been reported in [16].

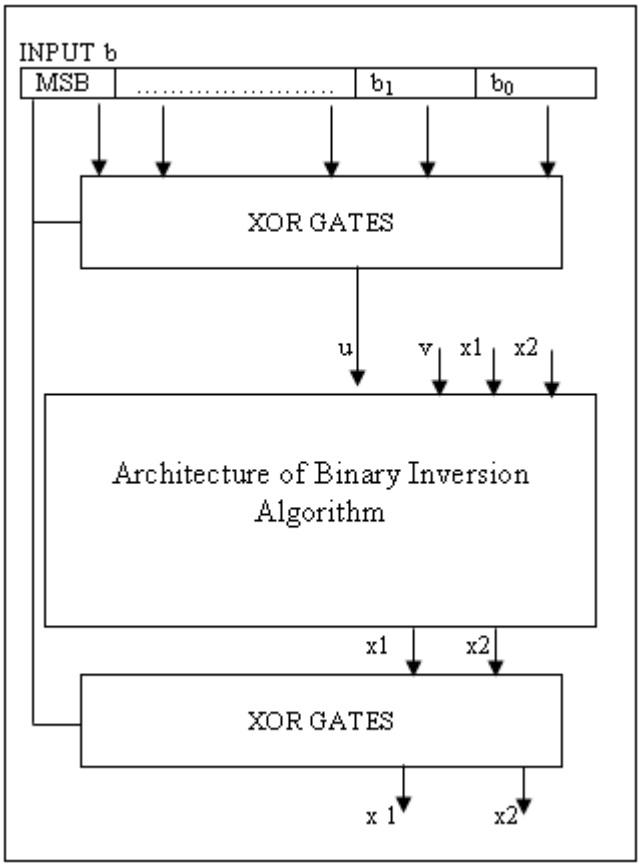

Fig. 5. New architecture of modified Binary Inversion Algorithm

In the above architecture the MSB of input b is common to one input of all the XOR gates shown in the boxes. If this MSB is 0 the above architecture is a regular inverter circuit. If the MSB is 1, then the input b is complemented and the outputs x1 and x2 are also complemented. If 'n' is the number of bits in prime 'p' then our new architecture requires only an additional 3xn number of XOR gates compared to architecture of Binary Inversion Algorithm.

IV. Results And Discussion

The speed up calculation of point multiplication is based on the difference in the addition and subtraction operations obtained with modified BIA and with normal BIA. The speed up of scalar point multiplication with modified BIA is depended on the occurrence of those inputs b='NOTa' which has less number of subtraction and addition operations in the modified BIA compared to normal BIA. Both statistical analysis and point multiplications are carried out to find the probability of occurrence of those values of b and to evaluate the performance of the modified BIA.

4.1 Statistical Analysis: The probability of occurrence of those values of 'b' = 'NOTa' which has more and lesser number of addition and subtraction operations in BIA than modified BIA are separately calculated. Then the average difference in the number of additions and subtractions for the above 2 cases are separately calculated. These obtained values are used to find out the average difference in the addition and subtraction operations when a scalar point multiplication operation is analytically performed with binary scalar multiplication algorithm (refer Fig.1).

If we assume that, on average 'n' is the number of ones in 'k' which is equal to n = L / 2, the binary method requires (L −1) point doublings and n point-additions where L denotes the number of bits of the scalar k. Usually the number of bits in 'k' is equal to number of bits in 'p'. The point doubling and point addition require inversion operation.

The average number of inversion is

(L-1) + n. ……..(1)

This equation is used to calculate the average difference in the number of subtraction and addition operations between modified BIA and BIA. Table 1 shows a detailed analysis with Mersenne's prime 2^5-1(31).

Let 'A' denote the occurrence of input b='NOTa' (refer fig.4) which has less number of subtraction and addition operations in the modified BIA compared to normal BIA.

Let 'B' denote the occurrence of input b='NOTa' which has more number of subtraction and addition operations in the modified BIA compared to normal BIA.

Let a1 denote the average difference in the number of subtraction and addition operations due to events A.

Let b1 denote the average difference in the number of subtraction and addition operations due to events B.

The average number of subtraction and addition operations reduced during point multiplication is equal to (L-1+n)(p(A).a1 – P(B).b1).

Table 1 shows the detailed analysis with Mersenne's prime 2^5 - 1.

Table 1. Analysis with Mersenne's prime 2^5 -1

Input b	Number of iterations with modified BIA	Number of iterations with BIA	|difference|
16	6	5	1
17	9	7	2
18	6	6	0
19	9	9	0
20	8	9	1
21	8	8	0
22	5	9	4
23	4	5	1
24	8	10	2
25	8	9	1
26	7	7	0
27	3	4	1
28	7	10	3
29	2	3	1
30	0	7	7

From the above table P(A)=9/30, P(B)=2/30, a1=18/9, b1=3/9.

Assuming number of bits of scalar 'k' is 5, the average number of subtraction and addition operations reduced during this point multiplication operation (denoted by 'avg') = 3.9

Table 2 shows the result with other Mersenne's primes

Mersenne's Prime	P(A)	P(B)	a1	b1	avg
2^5-1	0.3	0.066	2.33	1.5	3.899
2^7-1	0.238	0.047	2.8	1.833	5.50
2^{13}-1	0.236	0.041	2.902	2.618	10.68
2^{17}-1	0.239	0.0426	2.914	2.648	14.3
2^{19}-1	0.2386	0.0426	2.9309	2.7	16.06
2^{31}-1					

The above analysis shows that the average reduced number of subtraction and addition operations during a point multiplication operation is approximately 0.845 times of the bit length of scalar 'k'.

Because of the unfeasibility to process the whole number space with bigger Mersenne's primes and our objective is targeted to NIST recommended curve 2^{521}-1, we have restricted our analysis only on some parts of the number space of Mersenne's prime 2^{521}-1 . The table 2 shows the number space selected and the average result.

Table 2: Number space where analysis is performed

Number space
$(p^x-1)/2+1$ to $(p-1)/2+1+2^{17}$
$(p-1)/2+2^{127}$ to $(p-1)/2+2^{127}+2^{17}$
$(p-1)/2+2^{250}$ to $(p-1)/2+2^{250}+2^{17}$
$(p-1)/2+2^{350}$ to $(p-1)/2+2^{350}+2^{17}$
$(p-1)/2+2^{500}$ to $(p-1)/2+2^{500}+2^{17}$
Avg=438.6

$p=2^{521}-1$

4.2 Analysis by computing the point operations Q=k.P with randomly generated 'k'

In this method the performance evaluation of Modified Binary Inversion A is performed by computing point multiplications Q=k.P by using binary scalar multiplication algorithm on the NIST recommended elliptic curve with moduli 2^{521} -1. The curve has following parameters which are shown in Table 3.

The elliptic curve equation E over GF(p) is given by :$y^2 = x^3 + ax + b$; where p > 3, a= -3 ,$4a^3 + 27b^2 \neq 0$, and x, y, a,b∈ GF(p).

Table 3. Parameters of NIST recommended curve p521.

p = 6864797660130609714981900799081393217269435 3\ 0014330540939446345918554318339765605212255 9\ 6406614545549772963113914808580371219879997 1\ 6643812574028291115057151
r =6864797660130609714981900799081393217269435 3\ 0014330540939446345918554318339765539424505 7\ 7463332171975329639963713633211138647686124 4\ 0380340372808892707005449
b = 051 953eb961 8e1c9a1f 929a21a0 b68540ee a2da725b 99b315f3 b8b48991 8ef109e1 56193951 ec7e937b 1652c0bd 3bb1bf07 3573df88 3d2c34f1 ef451fd4 6b503f00
G x = c6 858e06b7 0404e9cd 9e3ecb66 2395b442 9c648139 053fb521 f828af60 6b4d3dba a14b5e77 efe75928 fe1dc127 a2ffa8de 3348b3c1 856a429b f97e7e31 c2e5bd66
G y = 118 39296a78 9a3bc004 5c8a5fb4 2c7d1bd9 98f54449 579b4468 17afbd17 273e662c 97ee7299 5ef42640 c550b901

In the above table p is the prime modulus, r is the order, b is the coefficient, G_x is the x coordinate of base point and G_y denote the y coordinate of base point.

In section 4.2.1, the steps involved in the generation of random numbers are discussed. In section 4.2.2, the performance evaluation of modified BIA is analyzed.

4.2.1 Generation of Random Numbers

Our method used two random number generators, namely Random Number Generator based on elliptic curve operations and Blum-Blum-Shub Generator

Random Number based on EC point operation: In the EC point multiplication method we generated random numbers for the above specified curve by using the initial seed value k which is less than the order of the curve. The steps involved are discussed below. Further details can be found in [18].

Step1: Computed $Q = k_n.P$ (n=1 initially) by using the initial seed k_n which is randomly selected and the x coordinate of the curve generated G_x is the random number.

Step2. If the random number generated is greater than the order of the curve then it is neglected and n= n+1.If the following condition is satisfied then number is stored in array and the array index is incremented.

Step3. $k_{n=} G_{x +} n$, the above steps are repeated until 100 random numbers are generated.

Random Number generation by using Blum-Blum-Shub Generator

In this method we generated another set of 100 random numbers with blum primes p= 12676506002282294014967039815 19,q=126765060022822940 1496704318359.[Further details can be found in [19]]. The generated bit sequences are grouped into 521 bits. This sequence is then compared with the order of the curve. If it is greater than the order of the curve, the most significant bits of these 521 bits are inserted with 0's until this value is lesser than the order of the curve. This random number is then stored in the array and array index is incremented and the steps are repeated until 100 random numbers are generated. But the generated sequence has to be subjected for random number testing.

4.2.2 Performance evaluation of modified BIA

We computed the point multiplication Q =kP with the 200 random numbers generated in previous section by using EC point operation and Blum-Blum-Shub Generator. The steps involved are:

Step1: The point multiplication Q=k.P is performed by substituting the first random number stored in the array.

Step2. During this point operation all the values for which inversion has to be computed are stored in an array.

Step3. Then statistical analysis explained in section 4.1 is performed on those values stored in the step2. The average reduced number of subtraction and addition operations is computed.

Step 4: The above steps are repeated for all the random numbers generated and the average of result obtained in step 3 is computed.

Table 4. Summary of the results

Average number of subtraction and addition operations reduced with modified BIA according to statistical analysis	438.6
Average number of subtraction and addition operations reduced with modified BIA when point operations are computed with randomly generated 'k' by using EC point operation	442.85
Average number of subtraction and addition operations reduced with modified BIA when point operations are computed with randomly generated 'k' by using Blum-Blum-Shub Generator	446.79
Average	442.74

The average number of subtraction and addition operations reduced during this point multiplication operation = 442.85

The above procedure is repeated with another set of 100 random numbers generated by using Blum-Blum_Shub Generator.

The average number of subtraction and addition operations reduced during this point multiplication operation = 446.79.

Table 4 shows the summary of the results obtained when statistical analysis and point multiplications are performed on NIST recommended p521 elliptic curve.

V. Conclusion And Future Scope

We have presented a new technique to speed up the computation of scalar point multiplication by slightly modifying the Binary Inversion Algorithm. The effectiveness of the technique is analysed by statistical analysis and computing the point operations Q=k.P with randomly generated 'k'. The statistical analysis and point operations on NIST recommend curve with moduli 2^{521}-1 show that modified BIA has reduced on average 442.74 number of subtraction and addition operations. The results obtained from the statistical analysis on other Mersenne's primes show that modified BIA has reduced the number of addition and subtraction operations on average by 0.845 times of the bit length of scalar 'k' which justified the result obtained when statistical analysis and point multiplication on NIST recommend curve with moduli 2^{521}-1 are performed. The above results show that proposed architecture of the modified BIA can be used to speed up the scalar point multiplication.

Our future effort will target speeding up computation of individual computational blocks of scalar point multiplication, integration of the proposed architecture with the other modules to compute scalar point multiplication in hardware.

References:

[1].C. Mclvor, M.McLoone and J.V.McCanny, "Modified Montgomery modular multiplication and RSA exponentiatuin techniques", IEE Proc. Comput.Digit.Tech., Voi.151,N9.6, November 2004

[2]QIANG Liu, Fangzhen Ma, Dong Tong, Xu Cheng, "A regular Parallel RSA Processor", The 47th IEEE International Midwest Symposium on Circuits and Systems.

[3]Jin Hua Hong, Cheng-Wen Wu, "Cellular-Array Modular Multiplier for fast RSA Public-Key Cryptosystem based on modified Booth's Algorithm", IEEE Transactions on VLSI systems, Vol.11, No.3, June 2003.

[4]Andre Vandemeulebroecke, Etienne Vanzieleghem, Tony Denayer, Paul G, " A new carry free division algorithm and its application to a single chip 1024-b RSA processor", IEEE Journal of Solid State Circuits, Vol.25, No.3, June 1990.

[5]Ming-Der Shieh, Jun-Hong Chen, Hao-Hsuan Wu,Wen-Ching Lin, "A new modular exponentiation architecture for efficient design of RSA cryptosystem", IEEE Transactions on VLSI systems, Vol.16, No.9, September 2008.

[6].William N Chelton, Mohammed Benaissa, "Fast Elliptic Curve cryptography on FPGA", IEEE Transactions on VLSI systems, Vol.16, No2. February 2008.

[7]. William N Chelton, Mohammed Benaissa, "Design of Flexible GF(2^m Elliptic Curve Cryptography Processors", IEEE transactions of VLSI systems, Vol.14, No.6, June 2006.

[8].Ray C.C. Cheung, Nicolas Jean-baptiste Telle, Wayne Luk, Peter Y.K. Cheung, "Customizable Elliptic Curve Cryptosystems", IEEE Transactions on VLSI systems, Vol.13, No.9, September 2005.

[9].Alireza Hodjat, David D. Hwang, Ingrid Verbauwhede, "A scalable and high performance elliptic curve processor with resistance to timing attacks", ITCC'05.

[10]Philip H. W. Leong, Ivan K.H.Leung, "A Microcoded Elliptic Curve Processsor using FPGA Technology",. IEEE Transactions on VLSI systems, Vol.10, No.5, October 2002.
[11]. Hamid Reza Ahmadi, Ali Afzali-Kusha, " Low-power flexible GF(p) Elliptic curve cryptography processor",
[12]. Kendall Ananyi, Hamad Alrimeigh, Daler Rakhmatov, " Flexible hardware processor for Elliptic curve cryptography", IEEE transactions on VLSI systems, Vol.17, No.8, August 2009.
[13]. Jyu-Yuan Lai, Chih-Tsun Huang, "Elixir: High throughput cost effective dual field processors and the design framework for ECC". IEEE Transactions on VLSI systems, Vol.16, No.11, October 2008
[14]. Kimmo Jarvinen, Jorma Skytta, "On parallelization of High-speed processors for Elliptic curve cryptograpgy", IEEE Transactions on VLSI systems, Vol.16, No.9, September 2008
[15]. Ciaran J McIvor, Maire McLoone, John V.McCanny, " Hardware Elliptic Curve Cryptographic Processor Over GF(p)", IEEE Transactions on Circuits and Systems , Vol.53, No.9, , September 2006
[16].Santhosh Ghosh, Monjur Alam, Indranil Sen Gupta, Dipanwita Roy Chowdhury, " A robust GF(p) parallel arithmetic unit for public key cryptography", 10th Euromicro Conference on Digital System Design Architectures, methods and tools(DSD 2007).
[17].Pierre Lecuyer and Richard Simard. " A C Library for Empirical testing of random number generators", ACM Transactions on Mathematical Software, Vol.33, No.4, Article 22, August 2007.
[18].Lap-Piu Lee, Kwok-Wo Wong, "A random number generator based on elliptic curve operations", An International Journal of computer and mathematics with applications. www.ElsevierMathematics.com.
[19]. Junod P, " Cryptographic secure PRBG: The Blum-Blum-Shub Generator." August 1999. http://crypto.junod.info/bbs.pdf