International Conference on Emerging Trends in Engineering, Science and Management
Sphoorthy Engineering College, Hyderabad, India (ESM-17)
17th and 18th March 2017 , www.conferenceworld.in ISBN: 978-93-86171-32-0

LOW LATENCY SCALABLE HIGH PERFORMANCE ELLIPTIC CURVE INVERSE BLOCK IN GF (2^m)

Ayesha Siddiqua[1], Dr. B. Sarala[2]

[1]*Department of Electronics and Communication, MVSR Engineering College, Hyderabad, (India)*

[2] *Head of Department of Electronics and Communication, MVSR Engineering College, Hyderabad, (India)*

ABSTRACT

Scalable Elliptic Curve Cryptography (ECC) inverse block for pseudo random curves is presented. Inverse block is implemented with all 5 pseudo random curves recommended by National Institute of Standards and Technology (NIST) without re-configuring the hardware. The divide and conquer method of Karatsuba Ofman algorithm is used to implement the modified multiplier for obtaining multiplication in finite field. Itoh Tsuji algorithm is implemented by using Brauer addition chain to obtain inversion in finite field. The proposed multiplier reduces the latency of inverse with negligible impact on area and power. The simulation results are analyzed using QuestaSim and synthesis is done using Cadence EDA tools. A comparison with previous techniques is also discussed in this paper.

Keywords: Bruaer addition chain, Elliptic Curve Cryptography (ECC), Itoh Tsuji algorithm, Karatsuba Ofman algorithm, National Institute of Standards and Technology (NIST).

I. INTRODUCTION

Techniques involving cryptography are widely being implemented using finite fields. Elliptic Curves were developed by Koblitz and Miller independently [1],[2]. Elliptic Curve Cryptography (ECC) gained importance in 2004 after usage of finite fields in Advanced Encryption Standard (AES) private key Cryptography. ECC makes use of finite field operations. It is more efficient because of shorter key sizes. Inverse is one of the most trivial operations performed in finite field. The inverse is obtained for all 5 National Institute of Standards and Technology (NIST) recommended key sizes by repeated multiplication and squaring.

Extended Euclidean algorithm was initially used to obtain inverse of an element. Fermat's Little theorem and Euler's theorem were practiced later to reduce complexity and increase speed. In this paper Itoh Tsuji algorithm[3] is used which is the fastest algorithm for the calculation of inverse in finite field. Fermat's Little theorem is the basis of Itoh Tsuji algorithm. It states that if 'a' is a non zero element of GF(2^m) then $a^{-1} = a^{2^{\wedge}m-2}$. Multiplication in finite field is implemented by using Karatsuba Ofman algorithm [4]. It uses divide and conquer technique thus latency and complexity is largely reduced by this algorithm [5]. Modified multiplier is used which performs finite field multiplication in 7 clock cycles irrespective of the size of the curve. Squaring in finite field is obtained by interleaving zeroes. The inverses of all 5 NIST recommended curves with key lengths of 571-bits, 409-bits, 283-bits, 233-bits and 163-bits [6] are obtained using Bruaer addition chain. ECC is

widely being used in many security standards and server side applications because of its complex engineering and shorter key requirements [7]. The proposed architecture reduces latency providing high speed with utmost security.

The rest paper is organized as follows: Section II provides literature review about finite fields and ECC; Section III provides the implementation of modified multiplier, square and reduction (SR) block, inverse block and its architecture; Section IV provides the results and comparison; and Section V concludes the paper.

II. ELLIPTIC CURVE CRYPTOGRAPHY

2.1 Finite Field Operations

ECC is based on one of the hardest arithmetic problems, the elliptic curve discrete logarithm problem, therefore making ECC a reliable cryptographic technique. The basis of ECC operations are finite field operations. The operations are finite field addition, multiplication, squaring and inversion. The finite field addition can be implemented by using bit wise X-OR operation. Multiplication operation is done by Karatsuba Ofman algorithm. Divide and conquer approach is used here to reduce the complexity of multiplication of large numbers from $O(n^2)$ to $O(n^{\log_2 3})$ [8]. The algorithm has two configurations; both the configurations are used in this paper: First configuration can be presented as:

$$X.Y = (x_1 2^l + x_0).(y_1 2^l + y_0) \quad -(1)$$
$$= x_1.y_1 2^{2\wedge 1} + [(x_0+x_1).(y_0+y_1)+x_1y_1+x_0y_0]2^l + x_0y_0$$

In equation 1 X.Y can be obtained in three multiplications of 'l' bit integers along with 4 additions, when compared to one multiplication of '2' bit integers. Second configuration can be presented as:

$$X.Y = (x_2 2^{2\wedge 1}+x_1 2^l +x_0)\cdot(y_2 2^{2\wedge 1} + y_1 2^l + y_0) \quad -(2)$$
$$= x_2\, y_2 2^{4\wedge 1}+(x_2\, y_1 +x_1\, y_2)2^{3\wedge 1} +(x_2\, y_0 +x_0\, y_2 +x_1\, y_1)2^{2\wedge 1}+(x_1\, y_0 +x_0\, y_1)2^l +x_0\, y_0$$
$$X.Y = x_2\cdot y_2 2^{4\wedge 1} + [(x_2+x_1)\cdot(y_2+y_1) + x_2\, y_2 + x_1\, y_1]2^{3\wedge 1} + [(x_2+x_0)\cdot(y_2+y_0) + x_2\, y_2 + x_0\, y_0 + x_1\, y_1]2^{2\wedge 1} +$$
$$[(x_1+x_0)\cdot(y_1+y_0) + x_1\, y_1 + x_0\, y_0]2^l + x_0\, y_0$$
$$= u_2 2^{4\wedge 1} + [v_2+u_2+u_1]2^{3\wedge 1} + [v_1+u_2+u_0+u_1]2^{2\wedge 1} + [v_0+u_1+u_0]2^l + u_0$$

In equation 2 X.Y can be obtained in six multiplications of 'l' bit integers along with 11 additions, when compared to one multiplication of '3' bit integers. 571 is the largest bit size that is to be implemented by the multiplier. The first and second configurations are used two times to implement multiplication of 571 bit. The 16 bit multiplication is done using finite field multiplier. Implementation of modified multiplier is shown in Section III.

Squaring operation is done by interleaving zeroes. Repeated squaring is done to reduce latency; addition is also implemented in the SR block. The reduction is done in the same block as shown in Section III, which reduces complexity to a very large extend. The reduction operation used in the proposed design is based on the reduction algorithms presented in [9]. Finite field inverse is obtained by repeated squaring and multiplication.

2.2 Itoh Tsuji Algorithm

Inversion is the most time consuming operation in finite field operations [10]. Itoh Tsuji algorithm is used to find inversion as it is the fastest algorithm[11]. It is implemented by using Bruaer addition chains, which

reduces the number of multiplications and thus latency. Let us consider an element 'a'Є GF (2^m). The inverse of an element can be obtained using Itoh Tsuji algorithm as:

$$a^{-1} = a^{2\wedge m-2} \quad -(3)$$

$$a^{2\wedge m-2} = a^{2(2\wedge m-1)}$$

where (m-1) is even, $2^{m-1} -1 = (2^{m-1/2} -1)(2^{m-1/2} +1)$

where (m-1) is odd, $2^{m-1} -1 = 2(2^{m-2/2} -1)(2^{m-2/2} +1)+1$

Table 1 BRUAER ADDITION CHAIN FOR GF (2^{283})

	$\beta_{ui}(a)$	$\beta_{uj+uk}(a)$	**Exponentiation**
1	$\beta_1(a)$		a
2	$\beta_2(a)$	$\beta_{1+1}(a)$	$\beta_1^2 \times \beta_1$
3	$\beta_4(a)$	$\beta_{2+2}(a)$	$\beta_2^{2\wedge 2} \times \beta_2$
4	$\beta_8(a)$	$\beta_{4+4}(a)$	$\beta_4^{2\wedge 4} \times \beta_4$
5	$\beta_{16}(a)$	$\beta_{8+8}(a)$	$\beta_8^{2\wedge 8} \times \beta_8$
6	$\beta_{17}(a)$	$\beta_{16+1}(a)$	$\beta_{16}^{2\wedge 1} \times \beta_1$
7	$\beta_{34}(a)$	$\beta_{17+17}(a)$	$\beta_{17}^{2\wedge 17} \times \beta_{17}$
8	$\beta_{35}(a)$	$\beta_{34+1}(a)$	$\beta_{34}^{2\wedge 1} \times \beta_1$
9	$\beta_{70}(a)$	$\beta_{35+35}(a)$	$\beta_{35}^{2\wedge 35} \times \beta_{35}$
10	$\beta_{140}(a)$	$\beta_{70+70}(a)$	$\beta_{70}^{2\wedge 70} \times \beta_{70}$
11	$\beta_{141}(a)$	$\beta_{140+1}(a)$	$\beta_{140}^{2\wedge 1} \times \beta_1$
12	$\beta_{282}(a)$	$\beta_{141+141}(a)$	$\beta_{141}^{2\wedge 141} \times \beta_{141}$

The exponents are further simplified in a similar manner. The Bruaer addition chain for 283 bits is shown in Table 1. From equation 3: $a^{-1} = a^{2\wedge 283-2} = a^{2(2\wedge 283-1)}$. This requires calculation of $\beta_{282}(a)$ where $\beta_j = a\wedge(2\wedge(j)-1)$. Squaring of $\beta_{282}(a)$ is done to obtain inverse of the element 'a' in GF(2^{283}). The addition chain required for the inversion of element in GF(2^{283}) is (1, 2, 3, 4, 8, 16, 17, 34, 35, 70, 140, 141, 282). In addition chain, β_{k+j} is defined as $\beta_{k+j} = ((\beta_k)\wedge(2*j))*\beta_j$.

Thus inverse can be obtained with a maximum of $2[\log_2(m-1)]$ number of multiplication operations and the number of squaring operations required are (m-1).

III. DESIGN AND ARCHITECTURE

3.1 Finite Field Arithmetic Blocks

To obtain inverse, recursive multiplication and squaring is done. Scalability is to implement all 5 NIST recommended curves without any change in hardware. Modified multiplier is implemented with the help of Karatsuba Ofman algorithm's two configurations. For the largest curve of 571bits, second configuration is implemented two times and first configuration is also implemented two times. This reduces the critical path when compared to the critical path obtained if two 571 bits were multiplied. The input X of 571 bits is split into 3 parts each of 191 bits.

Each 191 bit is again split into 3 parts each of 64 bits i.e., second configuration is implemented two times. Each 64 bit is again split into 2 parts each of 32 bits. Each 32 bit is again split into 2 parts each of 16 bits i.e., first

configuration is also implemented two times. The 16 bit multiplier is implemented by using bit wise AND and XOR operations. Different inputs are selected by the multiplexer at each clock cycle to be the operands of the 191 bit multiplier. The output of 191 bit multipliers is registered and added. The multiplication block needs '6' multiplication operations according to the second configuration and '1' clock cycle for addition in the ADD block, which is nothing but an XOR gate as shown in Fig. 1. Thus total 7 clock cycles are required for multiplication when compared to the previous work which required 9 clock cycles as shown in Fig 4(b) and 4 (a). Multiplication in all the curves is thus implemented in same number of clock cycles. The squaring finite field operation can be achieved by interleaving zeroes in between the bits of the operand. Two operations i.e., repeated squaring and addition operation are implemented using SR block.

Reduction is done only in the SR block to reduce complexity[12],[13]. The output of the multiplier block is given as an input to the SR block through 'X', for reduction after multiplication. A multiplexer is used to select the input to the reduction module as shown in Fig 2.

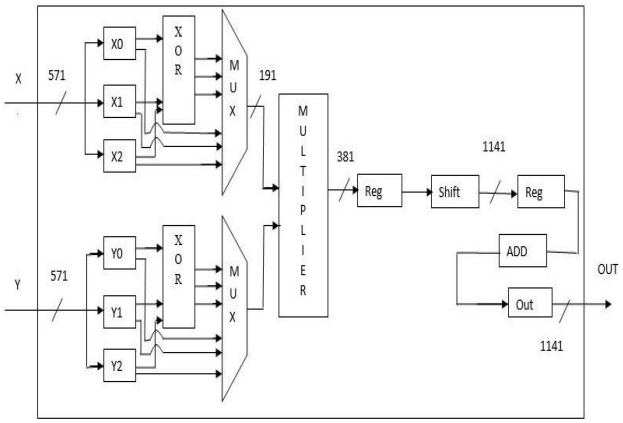

Fig. 1. Block diagram of modified multiplier

The least significant 571 bits of 'X' are given as an input to the SQ block. For repeated squaring the reduced and stored output of 'SQ' block present in 'OUT' is again fed back into the 'SQ' block. '2' clock cycles are required for performing addition and reduction operation. The repeated squaring operation requires '(t+1)' clock cycles to execute $A^{2^{\wedge}t}$ mod P.

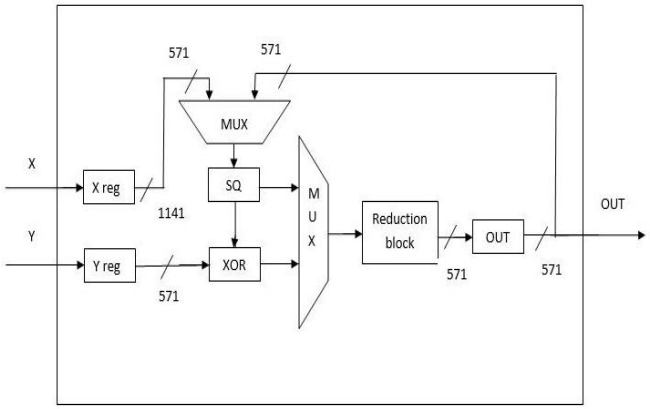

Fig. 2. Block diagram of SR block

3.2 Scalable Inversion

The block diagram of scalable inverse block is shown in Fig. 3. The inverse for all 5 NIST recommended Pseudo random curves is obtained without re-configuring the hardware by using the Bruaer addition chain. The Modified Multiplier and SR block inputs are selected with the help of multiplexers. The intermediate results are stored in a set of 571 bit registers, β_1, β_2, β_4, β_5, β_{10}, β_{20}, β_{40}, β_{80}, β_{81}, β_{162},…, β_{570} . The input to the 'SQ' block is selected by the multiplexer depending on the field.

For an element in GF (2^{163}) field, β_{162} is given as an input to 'SQ' block; for an element in GF (2^{233}) field β_{232} is given as an input to 'SQ' block; for an element in GF (2^{283}) field β_{282} is given as an input to 'SQ' block; for an element in GF (2^{409}) field β_{408} is given as an input to 'SQ' block; and an element in GF (2^{571}) field β_{570} is given as an input to 'SQ' block. Therefore, for an element in GF (2^m) field β_{m-1} is given as input to 'SQ' block. The output of the 'SQ' block is the inverse of the element.

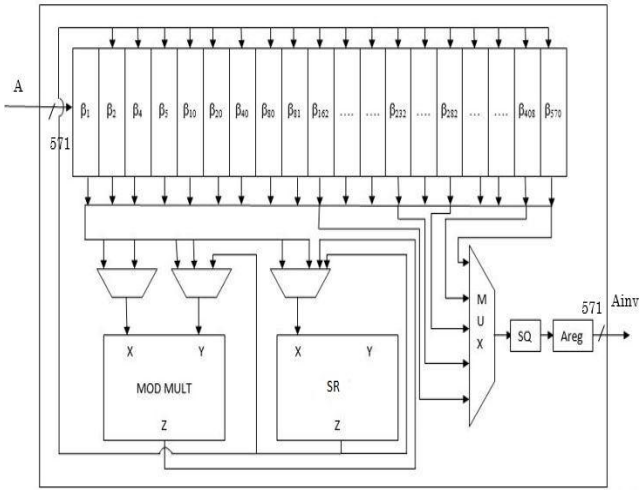

Fig. 3. Block diagram of scalable inverse block

IV RESULTS

The proposed Scalable inversion for all 5 NIST recommended Pseudo random curves is implemented by using QuestaSim Software and synthesized using Cadence EDA tools. Modified multiplier implemented in this paper takes only 7 clock cycle to perform multiplication operation when compared to the [15] which required 9 clock cycles as shown in Fig 4(b) and 4(a).

Fig. 4(a). Simulation results of multiplier in [15]

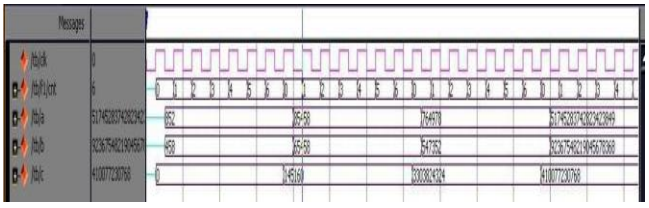

Fig. 4(b). Simulation results of modified multiplier

Table 2 shows the number of clock cycles required for computing all finite field operations. Multiplication requires 7 operations for all 5 curves. Addition with reduction is performed in 1 clock cycle. The repeated squaring operation to get A^{2^t} mod P is done in (t+1) clock cycles. Therefore, A^{2^7} mod P is obtained in 8 clock cycles. For further squaring, the result of the SR block at 8th clock cycle is again fed back to the SR block.

Table 2. CLOCK CYCLES FOR FINITE FIELD OPERATIONS

m	Multiplication	Squaring	Inverse
163	7	2	310
233	7	2	425
283	7	2	508
409	7	2	692
571	7	2	892

A comparison with the previous proposed work is done and a significant reduction in number of clock cycles is notable. Table 3 gives the total number of clock cycles required for calculation of finite field operations which shows, that the proposed work uses only '7' clock cycles for finite field multiplication and '2' clock cycles for finite field squaring.

Table 3. COMPARISON OF CLOCK CYCLES FOR FINITE FIELD OPERATIONS

m	Multiplication			Squaring			Inverse		
	[14]	[15]	This work	[14]	[15]	This work	[14]	[15]	This work
163	46	9	7	20	2	2	3654	320	310
233	78	9	7	28	2	2	7276	445	425
283	89	9	7	24	2	2	7747	530	508
409	181	9	7	36	2	2	16679	714	692

ISBN: 978-93-86171-32-0

571	332	9	7	42	2	2	28259	918	892

Synthesis of [15] and proposed work with modified multiplier is done with the Cadence EDA tool and the results are provided in Table 4. It can be seen that for the scalable inverse block when it is implemented with the modified multiplier, there is a reduction in the number of clock cycles as shown in Table 3, with very less impact on area and power consumption as shown in Table 4.

Table 4. COMPARISON OF AREA AND POWER CONSUMPTION

	[15]	**Proposed work**	**Increase**
Area	3562758.04	3565110.92	0.07%
Power(nW)	28413594.095	28417449.564	0.0135%

V CONCLUSION

This paper proposes the architecture of a scalable inverse block that can support all 5 pseudo-random curves recommended by NIST without re-configuring the hardware. The Karatsuba Ofman algorithm helps to reduce the latency by using divide and conquer technique. The proposed design uses Bruaer addition chain which reduces latency while maintaining the same area and power consumption when compared with [15]. The work presented, computes inverse of 571 bits in 892 clock cycles when compared to 918 [15]. Compared to [14] the latency of the proposed architecture is much lower especially when operating with large keys. To handle large volume of data, high speed server-side applications are required which can be achieved by this with reduced

latency and area. The scalability provided helps to establish different security level connections with different users.

REFERENCES

[1]. V. Miller, "Use of elliptic curves in cryptography," in *CRYPTO85:Proceedings of the Advances in Cryptology*. Springer–Verlag, 1986, pp. 417–426.

[2]. N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of Computation*,vol. 48, no. 177, pp. 203–209, 1987.

[3]. T. Itoh and S. Tsujii, "A Fast Algorithm for Computing Multiplicative Inverses in *GF* (2*m*) Using Normal Bases," *Information and Computation*, vol. 78, no. 3, pp. 171–177, 1988.

[4]. A. Karatsuba and Y. Ofman, "Multiplication of multi-digit numbers on automata," *Soviet Physics Doklady*, vol. 7, pp. 595–596, 1963.

[5]. T-W. Kwon, C-S. You, W-S. Heo, Y-K. Kang and J-R. Choi, "Two Implementation Methods of a 1024-bit RSA Cryptoprocessor Based on Modified Montgomery Algorithm," *Proc. of ISCAS 2001,* vol. 4, p.650-653, 2001.

[6].National Institute of Standards and Technology, "Recommended Elliptic Curves for Federal Government Use," July 1999.

[7].D. Hankerson, A. Menezes, and S. Vanstone, *Guide to Elliptic CurveCryptography*. New York, NY, USA: Springer-Verlag, 2004.

[8]. P. G. Comba, "Exponentiation cryptosystems on the IBM PC," *IBMSystems Journal*, vol. 29, no. 4, pp. 526–538, 1990.

[9]. Y. Zhang, D. Chen, Y. Choi, L. Chen, and S. Ko, "A high performance ECC hardware implementation with instruction-level parallelism over GF(2163)," *Microprocessors and Microsystems*, vol. 34, no. 6, pp. 228–236, October 2010.

[10] W. Stallings, "Cryptography and Network Security 4th Ed," Prentice Hall , 2005,PP. 58-309.

[11]. J. Lopez and R. Dahab, "Fast multiplication on elliptic curves over *GF* (2*m*) without precomputation," in *CHES99: Proceedings of the FirstInternational Workshop on Cryptographic Hardware and EmbeddedSystems*. Springer-Verlag, 1999, pp. 316–327.

[12]. J. Solinas, "Efficient Arithmetic on Koblitz Curves," *Designs, Codes andCryptography*, vol. 19, pp. 195–249, 2000.

[13]. Y. Zhang, D. Chen, Y. Choi, L. Chen, and S. Ko, "A high performance ECC hardware implementation with instruction-level parallelism over GF(2163)," *Microprocessors and Microsystems*, vol. 34, no. 6, pp. 228–236, October 2010.

[14] K. C. Cinnati Loi and Seok-Bum Ko" High Performance Scalable Elliptic Curve Cryptosystem Processor in GF(2m)" - *IEEE Trans* 2013

[15] K. C. Cinnati Loi and Seok-Bum Ko" FPGA Implementation of Low Latency Scalable Elliptic Curve Cryptosystem Processor in GF(2m)" - *IEEE Trans* 2014C.