An Arithmetic over GF (2^5) To Implement in ECC

A.R. Rishivarman
Pauls Engg. College
Villupuram,TN
India

B. Parthasarathy
Mailam Engg. College
Villupuram,TN
India

M. Thiagarajan
SASTRA University
Tanjore , TN
India

ABSTRACT

The potential for the use of the discrete logarithm problem in public-key cryptosystems has been recognized by Diffe and Hellman in 1976. Although the discrete logarithm problem as first employed by them was defined explicitly as the problem of finding logarithms with respect to a generator in the multiplicative group of the integers module a prime, this idea can be extended to arbitrary groups and in particular, to elliptic curve groups. The resulting public – key systems provide relatively small block size, high speed, and high security. In this paper an efficient arithmetic for operations over elements of GF(2^5) represented in normal basis is presented. The arithmetic is applicable in public-key cryptography.

General Terms
Security, Information Science

Keywords
Elliptic curve cryptography, Finite field , Simulation, public-key,

1. INTRODUCTION

In the Cryptographic schemes proposed in 90's and before are mostly based on the Discrete Logarithm Problem (DLP): Finding 'd' from the given group of elements Q and P such that ;

$$Q = d\,P \qquad (1)$$

We are concentrating only on curves over GF(2^5), where point coordinates are expressed as 5-bit vectors. The DLP in such a group is very hard as opposed to the DLP in the multiplicative group over a finite field. This means that a 173 bit key provides approximately the same security level as the 1024-bit RSA [6]. This fact is very important in applications such as chip cards, where the size of hardware and energy consumption is crucial. In algorithms such as the Elliptic Curve Digital Signature Algorithm, requires addition, multiplication and inversion over a finite field The implementation of these operations is determined by the representation of the field elements in finite field . In this work we focus on the normal basis representation. Addition over elements of GF(2^5) is implemented as a bit-wise XOR. Squaring is realized by rotation (cyclic shift) one bit to the right. Because it is so simple (one clock cycle), it is regarded as a special case. Multiplication is based on matrix multiplication over GF(2^5). In hardware, a special unit (multiplier) is necessary. The best-known algorithm for inversion in normal basis is the algorithm of Itoh, Tewhai and Tsujii [3] based on multiplication and squaring. As we can see, the main problem is an efficient normal basis multiplication. Other operations are either simple or based on multiplication, an irreducible polynomial taken for construction of the field GF(2^5) is $f(x) = x^5 + x^2 + 1$.

2. PREVIOUS WORK

In year of 1986 Massey and Omura in their publication proposed a multiplication and multiplier [5] that adopts the regularity of equations for all bits of result. From the equation for one bit of a result , equations for other bits can be derived by rotating bits of the arguments 'a 'and 'b' [2]. In this multiplier, one bit of the result is computed completely in one clock cycle. Then registers holding the arguments a and b are rotated right one bit between cycles. The computation of m bits of the result takes m clock cycles end hence this multiplication is also called bit-serial. Agnew, Mullin, Onyszchuk and Vanstone introduced a modification of the Massey-Omura multiplication [l]. They divided the equation for each bit c_i into m products;

$$c_i = P_{i,0} + P_{i,1} + \cdots + P_{i,m-2} + P_{i,m-1}$$

In the first clock cycle, the: product $P_{i,i+0}$of bit $c_i$for all $i\epsilon\langle 1, m-1\rangle$ is evaluated. In the next cycle, the product (all subscripts are: reduced mod m) of bit c_i for all $i\epsilon\langle 1, m-1\rangle$ is evaluated and added to the intermediate result, and so on. All bits of the result are successively evaluated in parallel; the computation is pipelined.

The amount of hardware is the same as for the non-pipelined Massey-Omura multiplication, but the critical path is short and constant and so the maximum achievable frequency is higher. This multiplication is widely used. The computation of an inverse element by the algorithm [3] is usually controlled by a micro program **[4].** When implementing the inversion using classical multiplier, additional registers and data transfers outside the multiplier are necessary. In this work we present a modification of the classical multiplier, which allows an efficient implementation of both the multiplication and inversion algorithms. In comparison with the micro-programmed inversion, no additional registers or data transfers outside the multiplier are necessary. We also introduce several improvements of this multiplication - inversion unit, which lead to increased performance. But in the case of addition and squaring the arithmatic presented have no performance change; the only difference is modification according to the field selected.

3. GOLOIS FIELD ARITHMETIC GF(2^5)

GF(2^5)can be viewed as a vector space of dimension 5 over the field GF(2). There are several bases known for GF(2^5) . The most common bases are polynomial bases and nonmal bases . with a polynomial basis , the field elements are represented by binary polynomials modulo an irreducible binary polynomial of degree 5 . given an irreducible polynomial

$$p(x) = x^5 + x^2 + 1 - -(2)$$

An element A $\epsilon\ GF(2^5)$ is represented either as A(α)= $\sum_{i=1}^{4} c_i \alpha^i$ or as $(c_4 c_3 c_2 c_1 c_0)$, where $c_i \epsilon GF(2)$ and α , the root of $p(x)$. Here the basis is { 1, $\alpha^1 \alpha^2 \alpha^3 \alpha^4$}. It has been proved that there always exists a normal basis for the given finite field GF(2^5) which is of the form N={$\beta, \beta^2 \beta^4 \ldots \beta^{2^4}$} where β is aroot of the irreducible polynomial $p(x) = x^5 + x^2 + 1$ over GF(2) and elements of the set are linearly independent . we say that β generates the normal basis N , or β is a normal elements of $GF(2^5)$. β will equal to $\alpha^i$for some i. Then every element A $\epsilon\ GF(2^5)$is represented as

$$A(\beta) = \sum_{i=0}^{4} a_i \beta^i, --(3)$$

Where $a_i \epsilon$ GF(2). A field element can thus be represented in a bit vector of length 5. Hence we have explained in Table 1;

Table 1: Polynomial and Normal Form

S.NO	BIT STRING	POLYNOMIAL FORM	NORMAL FORM
1	00000	0	β^{32}
2	00001	1	β^{31}
3	00010	$c_1\alpha^1$	β^1
4	00100	$c_2\alpha^2$	β^2
5	01000	$c_3\alpha^3$	β^3
6	10000	$c_4\alpha^4$	β^4
7	00011	$c_1\alpha^1 + 1$	β^{18}
8	00101	$c_2\alpha^2 + 1$	β^5
9	01001	$c_3\alpha^3 + 1$	β^{29}
10	10001	$c_4\alpha^4 + 1$	β^{10}
11	11000	$c_4\alpha^4 + c_3\alpha^3$	β^{21}
12	10100	$c_4\alpha^4 + c_2\alpha^2$	β^7
13	10010	$c_4\alpha^4 + c_1\alpha^1$	β^{30}
14	00110	$c_2\alpha^2 + c_1\alpha^1$	β^{19}
15	01100	$c_3\alpha^3 + c_2\alpha^2$	β^{20}
16	01010	$c_3\alpha^3 + c_1\alpha^1$	β^6
17	00111	$c_2\alpha^2 + c_1\alpha^1 + 1$	β^{11}
18	01011	$c_3\alpha^3 + c_1\alpha^1 + 1$	β^{27}
19	10011	$c_4\alpha^4 + c_1\alpha^1 + 1$	β^{17}
20	11100	$c_4\alpha^4 + c_3\alpha^3 + c_2\alpha^2$	β^{13}
21	11010	$c_4\alpha^4 + c_3\alpha^3 + c_1\alpha^1$	β^9
22	01110	$c_3\alpha^3 + c_2\alpha^2 + c_1\alpha^1$	β^{12}
23	10110	$c_4\alpha^4 + c_2\alpha^2 + c_1\alpha^1$	β^{28}
24	11001	$c_4\alpha^4 + c_3\alpha^3 + 1$	β^{25}
25	10101	$c_4\alpha^4 + c_2\alpha^2 + 1$	β^{22}
26	01101	$c_3\alpha^3 + c_2\alpha^2 + 1$	β^8
27	01111	$c_3\alpha^3 + c_2\alpha^2 + c_1\alpha^1 + 1$	β^{23}
28	10111	$c_4\alpha^4 + c_2\alpha^2 + c_1\alpha^1 + 1$	β^{26}
29	11011	$c_4\alpha^4 + c_3\alpha^3 + c_1\alpha^1 + 1$	β^{16}
30	11101	$c_4\alpha^4 + c_3\alpha^3 + c_2\alpha^2 + 1$	β^{14}
31	11110	$c_4\alpha^4 + c_3\alpha^3 + c_2\alpha^2 + c_1\alpha^1$	β^{24}
32	11111	$c_4\alpha^4 + c_3\alpha^3 + c_2\alpha^2 + c_1\alpha^1 + 1$	β^{15}

The following properties of a Golois field with normal basis are useful in application:

For any element A$\epsilon GF(2^5)$

$1 = A + A^2 + A^4 + .. + A^{2^4}$.

This implies that normal basis representation of 1 is (11111).

For any element A $\epsilon\ GF(2^5)$

$A^2 = \sum_{i=0}^{4} a_i \beta^{2^{i+1}} = \sum_{i=0}^{4} a_{i-1} \beta^{2^i} = (a_3 a_2 a_1 a_0 a_4)$.

3.1.$GF(2^5)$ Addition:

(a_4,a_3,a_2,a_1, a_0) ± (b_4,b_3,b_2,b_1, b_0) =(c_4,c_3,c_2,c_1,c_0), where $c_i = a_i \oplus b_i$ overGF(2)Note that in GF(2^5), Since

(a_4,a_3,a_2,a_1,a_0) +(a_4,a_3,a_2,a_1,a_0) = (0,0,0,0,0) , each element $(a_4,a_3,a_2,a_1,a_0)$is its own additive inverse. Addition and Subtraction can be implemented efficiently as component wise exclusive OR in the NB representation.

3.2.$GF(2^5)$ Squaring:

By the second property of normal basis ,squaring of an element a in NB representation is a cyclic shift operation .

3.3. $GF(2^5)$ Multiplication:

Let A and B be two arbitrary elements in $GF(2^5)$ in a NB representation and C= A.B be the product of A and B. we denote $A = \sum_{i=0}^{4} a_i \beta^{2^i}$ as a vector A=(a_0,a_1,a_2,a_3,a_4) , $B = \sum_{i=0}^{4} b_i \beta^{2^i}$ as a vector B = (b_0,b_1,b_2,b_3,b_4) where C=

(c_0,c_1,c_2,c_3,c_4) , then the last term c_4 of C is a logic function of the components of A and B, that is, c_4=

$f(a_0,a_1,a_2,a_3,a_4; b_0,b_1,b_2,b_3,b_4)$.

Since squaring in NB representation is a cyclic shift operation, we have $C^2=A^2.B^2$ or equivalently

$(c_4 ,c_0,c_1,c_2,c_3)= (a_4 ,a_0,a_1,a_2,a_3). (b_4 ,b_0,b_1,b_2,b_3)$. Hence, the last component c_3 can be obtained by the same function f that is, $c_3= f(a_4 ,a_0,a_1,a_2,a_3; b_4 ,b_0,b_1,b_2,b_3)$. By squaring C repeatedly , we get

$c_4=f(a_0,a_1,a_2,a_3,a_4; b_0,b_1,b_2,b_3,b_4)$

$c_3= f(a_4 ,a_0,a_1,a_2,a_3; b_4 ,b_0,b_1,b_2,b_3)$

:

$C_0= f(a_1,a_2,a_3, a_4, a_0; b_1,b_2,b_3, b_4, b_0)$

By the above equation define the Massey-Omura multiplier in normal basis representation. In the multiplier, the same logic function f for computing the last component of c_4 of the product 'c'

Can be used to get the remaining components c_3, c_2, c_1, c_0 of the product sequentially. In parallel architecture, we can use 5 identical logic function f for calculating all components of the products of the product simultaneously .the product of A and B in the field $GF(2^5)$is

$$C = A*B = c_0\beta^1 + c_1\beta^2 + c_2\beta^4 + c_3\beta^8 + c_4\beta^{16}$$

$$= (a_0\beta^1 + a\beta^2 + a_2\beta^4 + a_3\beta^8 + a4\beta16\times(b0\beta1+b1\beta2+b2\beta4+b3\beta8+b4\beta16)$$

Thus, we can get $c_k = \sum_{i=0}^{4}\sum_{j=0}^{4} \mu_{ij}^{(k)} a_i b_j$, $0 \le k \le 5$.

The 5×5 matrices $\mu^{(k)}(0 \le k \le 5)$ whose elements $\mu_{ij}^{(k)}$, $0 \le i,j \le 5$ can be obtained if we know the transformation between the elements of the PB and the elements of NB ,that is, the normal basis representation of the elements of the PB.

For a normal basis there always exist a multiplication table T(corresponding to the irreducible polynomial),which is given by $\beta \begin{bmatrix} \beta \\ \beta^2 \\ \vdots \\ \beta^{2^4} \end{bmatrix} = T \begin{bmatrix} \beta \\ \beta^2 \\ \vdots \\ \beta^{2^4} \end{bmatrix}$.

Corresponding to aT matrix , there always exists a matrix $\mu^{(k)}$for any $c_k$of the product c , for the given irreducible polynomial which defines the normal basis in $GF(2^5)$. After the multiplication table T is obtained , the matrix $\mu^{(k)}$can be calculated according to the above method.

3.4. $GF(2^5)$ Invertion:

We know from Fermat's theorem that for any non-zero elements $\beta, \beta^{2^5-1} = 1$, that is, $\beta^{-1} = \beta^{2^5-2}$.

$2^5 - 2 = 2^1 + 2^2 + 2^3 + 2^4$. Hence,

$\beta^{-1} = \beta^2 * \beta^4 * \beta^8 * \beta^{16}$.That is an inversion requires 4squaring 3 multiplication. This could be reduced further by alternative methods[7,8].

4. ELLIPTIC GROUP OPERATION

Elliptic group operations includes point negation, point subtraction, point doubling, and scalar multiplication.

Let GF(2^5) be a characteristic 2 finite field. Then a (non-super singular) elliptic curve E(GF(2^5)) over GF(2^5) defined by E: $y^2 + xy = x^3 + x^2 + 1$ consists of the set of solutions or points P=(x,y) for x,y $\in$ GF(2^5) .

E: $y^2 + xy = x^3 + x^2 + 1$ *in* GF(2^5)

together with an extra point *O* called the point at infinity. The number of points on E(GF(2^5)) is denoted by

E(GF(2^5)). The Hasse Theorem states that:

$2^5 + 1 - 2\sqrt{2^5} \le \#\ E(GF(2^m)) \le 2^5 + 1 + 2\sqrt{2^5}$.

It is again possible to define an addition rule to add points on *E* as the addition rule is specified as follows:

1. Rule to add the point at infinty to itself:
   O + O =O

2. Rule to add the point at infinity to any other point:
   (x,y) + O = O +(x,y)= (x,y) for all (x,y) $\in$ GF(2^5)

3. Rule to add two points with the same x–coordinates when the points are either distinct or have *x*-coordinates 0:

   (x,y) + (x,x + y)= O for all (x,y)$\in$ GF(2^5)

4. Rule to add two points with different *x*-coordinates: Let $(x_1,y_1) \in GF(2^5)$ and $(x_2,y_2) \in GF(2^5)$ be two points such that $x_1 \neq x_2$. Then $(x_1,y_1)+(x_2,y_2)=(x_3,y_3)$, where:

$x_3=\lambda^2+\lambda+x_1+x_2+a$ in $GF(2^5)$,

$y_3=\lambda.(x_1+x_3)+x_3+y_1$ in $GF(2^5)$, and

$$\lambda \equiv \frac{y_1+y_2}{x_1+x_2} \text{ in } GF(2^5).$$

5. Rule to add a point to itself (double a point): Let $(x_1,y_1) \in GF(2^5)$ be a point with $x_1 \neq 0$. Then $(x_1,y_1)+(x_1,y_1)=(x_3,y_3)$,where:

$x_3=\lambda^2+\lambda+a$ in $GF(2^5)$, $y_3=x^2{}_1+(\lambda+1)x_3$ in $GF(2^5)$,

$$\text{and } \lambda = x_1 + \frac{y_1}{x_1} \text{ in } GF(2^5).$$

The set of points on $E(GF(2^5))$ forms an abelian group under this addition rule. Notice that the addition rule can always be computed efficiently using simple field arithmetic. Cryptographic schemes based on ECC rely on scalar multiplication of elliptic curve points. As before given an integer *d* and a point $P \in GF(2^5)$, scalar multiplication is the process of adding *P to itself* d times. The result of this scalar multiplication is denoted 'dP'.

5. A PERFORMANCE IMPROVEMENT

The basic ECC operation (1) is performed by successive point additions and point doublings. Each of these operations needs 1 inversion, 2 multiplications and 1 squaring [2]. The number of clock cycles necessary for one point addition or doubling is then:

$$C_{PADD} = ([log_2 4] + w4 + 1)C_{MUL} + (5 - w4)C_{SQR}$$

$$+Const - - - (4)$$

We can reduce the number of clock cycles C_{PADD} in two ways: by speeding up the multiplications and by reducing the number of clock cycles necessary for the iterative squaring.

5.1. Serial Multiplication

The Massey-Omura need *m* clock cycles for computing all *m* bits of the result. Some authors also call them bit-serial, because they compute one bit of a result in one clock cycle. There is a digit-serial variant of the Massey-Omura multiplication (some author call it sliced or parallel). In this multiplication, *D* bits (also called a digit) are evaluated in one clock cycle. In the case of digit-serial multiplier, *D* products $C_{i,j}$ are evaluated in one clock cycle. All 5- bits of the result are then evaluated in $C_{MUL} = [5/D]$ cycles. Since more products are evaluated in one clock cycle, more combinational logic is necessary. The size of the block *COMB.LOGIC* is proportional to $D+1$. The size of other blocks remains constant. As the combinational logic becomes more complex, the length of the critical path grows proportionally to $\log D$. Since one multiplication needs 5/*D* clock cycles, the total time necessary for one multiplication is $O(5/D \times \log D)$and the total time of one inversion (or point addition on elliptic curve) is:

$$T_{PADD} = O\big((log5.(5/D)+5)logD\big) - -(5)$$

5.2. Speeding up the Squaring

Another way to improve the performance of the [3] inversion (and consequently the point addition) is to reduce the number of clock cycles necessary for the iterative squaring. Adding one or more blocks performing, long distance"rotations can reduce the number of clock cycles required for all iterative squaring .It seems that the number of clock c:ycles spent in iterative squaring is approx. $O\big(k\sqrt[k]{4}\big)$, where ***k*** is the number of rotation blocks. The total time necessary for the point addition is then

$$T_{PADD} = O\left(\left((5/D)log5 + k\sqrt[k]{4}\right).logD\right) - -(6)$$

6. CONCLUTION

Finite field $GF(2^5)$arithmetic operations include addition , subtraction, multiplication , squaring and inversion. Due to proposed field $GF(2^5)$and irreducible polynomial $f(x)=x^5+x^2+1$ both additions and subtractions can be implemented very efficiently. Multiplication in NB using the said polynomial is faster and secure. Squaring a special case of multiplication can be implemented much faster than multiplication in the NB. Also inversions in the NB with the 'almost inverse method runs', can be implemented efficiently. When comparing the arithmetic in the PB, the proposed arithmetic in NB is 20% faster [9]. More over the implementation of key sharing schemes are efficient and faster in the proposed arithmetic [10]. Thus the proposed method will give us a faster and secure cryptographic scheme in the limited environments.

Elliptic group operations include point negations, point additions, point subtraction, point doubling and scalar multiplication. The operations can be implemented very efficiently in the proposed NB.

7. LIMITATION AND FUTURE WORK

Our work has been done within the finite field GF(2^5) and the arithmetic can be generalized in GF(2^n) , stage by stage, in the future works. Also, performance of various representations [11] of the elements of the general field can be compared.

8. REFERENCES

G.B. Agnew, R.C. Mullin, I.M. Onyszchuk, and S.A.Vanstone, **"**An Implementation for a Fast Public-Key Cryptosystem," Journal of Cryptology, vol. 3, pp.63-79, 1999

IEEE 1363. Standard for Public-key Cryptography., IEEE 2000

Itoh, Teechai, and Tsujii, "A Fast Algorithm for Computing Multiplicative Inverses in GF(2^t) using normal bases," J. Society for Electronic Communications (Japan), vol.44**,** pp.31-36, 1986

P.H.W Leong and K.H. Leung, "A Micro coded Elliptic Curve Processor Using FPGA Technology," IEEE Transactions on VLSI Systems, vol. 10, no. ***5,*** pp. 550-559, Oct. 2002

J. Massey, and J. Omura, "Computational Method and Apparatus for Finite Field Arithmetic," U.S. patent number 4,587,627, 1986

I. Blake, G. Seroussi and N. Smart, "Elliptic Curves in Cryptography", Chapter 1. Cambridge University Press, Cambridge (UK), 1999

N.Koblitz, Introduction to Elliptic Curves and Modular Forms, 2nd Ed., Spinger-Verlag,1993.

N.Koblitz, Elliptic Curve Cryptosystems, Math.Compu.Vol.48,No.177, Jan,1987,pp 203-209.

A.R.Rishivarman, B.Parthsarathy, M.Thiagarajan, "An efficient performance of GF(2^5) arithmetic in the elliptic curve cryptosytem", International journal of computing and application, vol 4 no. 2 pp 111-116, 2009

A.R.Rishivarman, B.Parthsarathy, M.Thiagarajan, "A key sharing scheme over GF(2^5)", Springer-CCIS, vol 283 2012

A.R.Rishivarman, B.Parthasarathy, M.Thiagarajan, "A Montgomery representation of elements in GF(2^5) for efficient arithmetic to use in elliptic curve cryptography", International journal of advanced networking and application, vol 1 no. 5 pp 323-326, 2010