A Study on Finite Field Multiplication over GF (2^m) and its application on Elliptic Curve Cryptography

Ashutosh Narayan Mishra¹, Dr. R.K.Singh¹
¹Kamla Nehru Institute of Technology (KNIT), Sultanpur, Uttar Pradesh, India
Email- mishra.as123@gmail.com, singhrabinder57@gmail.com

ABSTRACT: The paper presents an extensive and careful study of finite field multiplication over GF (2^m) using polynomial basis as well as special polynomial like trinomials, pentanomial and All one polynomial (AOP). This multiplication is done by using montgomery multiplication scheme and application of it is also given. This paper focuses on different arithmetical operation on elliptic curve cryptography over GF (2^m). The parameter performance is also discussed in term of number of component, latency, space and time complexity.

Keywords: Elliptic curve cryptography, Finite Field, polynomial basis, montgomery multiplication, LSD-First, MSD-First.

1. Introduction

Finite field has received a lot of attention due to its widespread applications and computation in cryptography [1], coding theory, error control, digital signal processing [2,3]. Finite field arithmetic is used in Cryptography. Elliptic curve cryptography [4,5] and RSA[6] is two important Public Key cryptosystem. All the low-level operations are carried out in finite fields.

The mathematical model of finite field includes addition, subtraction, multiplication, division, inversion and squaring etc. Finite field arithmetic operates in prime, binary and extension field which is significant tool of public key cryptography. Addition and multiplication are two basic operations in the binary finite field GF(2^m). Addition in GF(2^m) is easily realized using m two-input XOR gates while multiplication is complex operation and its performance is counted in terms of gate count and time delay. The other operations of finite fields, such as exponentiation, division, and inversion can be performed by repeated multiplications. For hardware implementation of cryptographic application binary field is more suitable.

Finite Field as vector space, polynomial basis [7-15] normal basis [16,17] dual basis [18,19] and redundant basis is used to represent the field. Each bases have their own advantage and disadvantage, such as for square of element normal base is preferred than others. Squaring is performed in normal basis just by a cyclic right-shift, while it is performed by bit-extension through insertion of 0 between the consecutive bits followed by modular reductions to reduce the extended polynomial of degree2m-2 to degree m-1 in case of polynomial bases. Inversion is also requires less area and time-complexity in normal basis. But as our requirement is finite field multiplication, polynomial basis has better performance.

The polynomial basis multipliers have low design complexity, simplicity, modularity ,regularity, offer scalability for the fields of higher orders, and does not require a basis conversion in multiplication architecture, So, it is widely used in hardware realization of system. Systolic array architecture is prefferd in polynomial basis (PB) multiplier in which a basic cell is repeated in an array and signals flow unilaterally between neighbors.

Multiplication in PB contains two steps: partial multiplication and modular reduction. Modular multiplication is the most important arithmetic operation in ECC cryptosystem. For efficient implementation of modular multiplication, montgomery multiplication algorithm was proposed by Montgomery[20]. The montgomery multiplication algorithm does not require division operations and it performs the reduction operation depending on the least significant digit rather than the most
significant digit. The modular multiplication using Montgomery multiplication algorithm is shown by Koc and Acar [21] in GF(2^m) fields. The Montgomery multiplication is used to design an Elliptic Curve Cryptography (ECC) based crypto-processor in [22]. It is implemented with a semi-systolic array structure in [23]. Semi-systolic array structures provide low latency in comparison to systolic array implementations and require fewer latches. Also, they can be pipelined to increase the throughput of the system. An efficient architecture for bit parallel Montgomery multiplier and squarer in GF(2^m) fields generated with irreducible trinomials proposed by Wu [24]. Fournaris and Koufopavlou [25] presented both pipelined and folded architectures for the Montgomery multiplication in GF(2^m). Bajard et al. [26] provided a Montgomery multiplier over GF(pm) other than GF(2^m). An unified multiplier for GF(P) and GF(2^m) which uses Montgomery multiplication algorithm for both fields is introduced by E. Sava et al [27] in 2000. The multiplier based on word-size pipeline structure can handle operands of any size. But their pre-computed constants and transformations uses Montgomery algorithm for multiplier. [A] Chin-Wun Chiou et all present time independent Montgomery multiplication scheme in GF (2^m). Two unified multipliers for both fields which are scalable and offer faster computation of multiplication is proposed by Savas et al. [28]. Their multipliers are also based on the Montgomery algorithm and provided high-radix design for low-power and high-performance applications. Satoh and Takano [29] introduced a scalable dual-field processor for Elliptic Curve Cryptography by using the Montgomery algorithm.

In this paper, section 2 represents Fundamental of Finite Field and Elliptic curve cryptography, section 3 represents traditional bit-level Montgomery Multiplication and Time-independent Montgomery Multiplication. Section 4 represents Polynomial multiplication, this section further divided [A], [B] and [C], and this section contain both polynomial multiplication as well as Montgomery multiplication scheme using trinomial, pentanomial and all-one-polynomial, further [A], [B] and [C] is divided in to 1,2,3 part according to requirement and last section of this paper 5 is conclusion of this paper.

2. Fundamental of Finite Field and Elliptic curve cryptography:

Field is mathematical place where we can do addition, multiplication, division, inverse, square etc. Basically, A field is a set F with a multiplication and addition operation which satisfy given rules i.e. associativity and commutativity of both addition and multiplication, the distributive law, existence of an additive identity 0 and a multiplicative identity 1, additive inverses, and multiplicative inverses for everything except 0. The finite field F(2^m) is the characteristic 2 finite field containing 2^m elements. Although there is only one characteristic 2 finite field F(2^m) for each power 2^m of 2 with m≥1, there are many different ways to represent the elements of F F(2^m). Elements of F(2^m) should be represented by the set of binary polynomials of degree m-1 or less i.e.

\[ a_{m-1}x^{m-1} + \cdots + a_1x + a_0 \]

Addition: The addition is quite simple in F(2^m), C=A+B where

\[ A = a_{m-1}x^{m-1} + \cdots + a_1x + a_0 \]

\[ B = b_{m-1}x^{m-1} + \cdots + b_1x + b_0 \]

It is logical XOR operation i.e. module 2 addition. So C=a+b, where + denote bitwise XOR operation, ai and bi represent the field element of F(2^m) where i varies from 0 to m-1. To represent it, polynomial, shifted polynomial and normal basis is used.

Multiplication: Multiplication is different and complicated than addition in F(2^m). In polynomial and shifted polynomial basis C=AB mod F(x), in normal basis C=A.B represent the multiplication. Koc and Acar [21] introduced Montgomery multiplication algorithm for fast modular integer multiplication.

Square and inversion: Square is special form of multiplication. For square normal basis is preferred over polynomial, as it is circular left shift in normal basis while it is much complicated in polynomial basis. If A ∈ F(2^m) then to find A^2, such that A^-1.A=1. Itoh and Tsuji [B] proposed inversion algorithm that is used for hardware implementation.

Let F(2^m) be characteristic 2 finite field and let a,b ∈ F(2^m) satisfy b≠0 in F(2^m). Then elliptic curve E (F(2^m)) over F(2^m) defined by the parameters a,b & F(2^m) consists of the set of solutions or points P= (x,y) for x,y & F(2^m) to the equation : y^2+xy=x^3+ax^2+b with a point at infinity o.

If the number of point in elliptic curve is denoted by #E(F(2^m)) then according to Hasse theorem
\[2^m + 1 - 2 \sqrt{2^m} \leq \#E(F_{2^m}) \leq 2^m + 2 \sqrt{2^m}\]

1. Rule to add the point at infinity to itself:
   \[O + O = O\]

2. Rule to add the point at infinity to any other point:
   \[(x, y)+O = O +(x, y) = (x, y) \text{ for all } (x, y) \in E(F_{2^m})\]

3. Rule to add two points with the same x-coordinates when the points are either distinct or have X-coordinate 0:
   \[(x, y) + (x, y) = O \text{ for all } (x, y) \in E(F_{2^m}), \text{ the negative of the point } -(x, y) = (x, x+y)\]

4. Rule to add two points with different x-coordinates: Let \((x_1, y_1) \in E(F_{2^m})\) and \((x_2, y_2) \in E(F_{2^m})\) be two points such that \(x_1 \neq x_2\). Then \((x_1, y_1) + (x_2, y_2) = (x_3, y_3)\), where:
   \[x_3 = \lambda^2 + \lambda + x_1 + x_2 + a \quad \text{and} \quad y_3 = \lambda (x_1 + x_2) + x_1 + y_1 \in E(F_{2^m}) \quad \text{and} \quad \lambda = y_1 + y_2 / x_1 + x_2 \in E(F_{2^m})\]

5. Rule to add a point to itself (double a point): Let \((x_1, y_1) \in E(F_{2^m})\) be a point with \(x_1 \neq 0\). Then
   \[(x_1, y_1) + (x_1, y_1) = (x_3; y_3), \text{ where: } x_3 = \lambda^2 + \lambda + a, \quad y_3 = x_1^3 + (\lambda + 1) x_1 + \lambda = x_1 + y_1 / x_1\]

Here elliptic curve E represented by the affine coordinates \((x_1, y_1)\) and \((x_2, y_2)\), respectively. From above it is clear that inversion is required to represent rule no. 4 and 5 i.e. point add and double a point. As inversion is an expensive operation in finite field. So, we can use projective-coordinate approach proposed by Lopez and Dahab [30] to improve their performance.

Scalr multiplication of elliptic curve points is the main operation in ECC, known as point multiplication. For given an integer \(k\) and a point \(P \in E(F_{2^m})\), scalar multiplication is the process of adding \(P\) to itself \(k\) times. The result of this scalar multiplication is denoted \(kP\) or \(kP\). It may be achieved by Montgomery ladder scalar multiplication scheme using Lopez and Dahab coordinates.

3. Montgomery Multiplication:

The use of modular multiplication is the most important arithmetic tool in Finite field. Montgomery Multiplication may be applied for both GF(p) and GF(2\(^m\)). But the requirement of Montgomery algorithm is pre-computation constant and the basis conversion. The modular multiplication using Montgomery multiplication algorithm is shown by Koc and Acar [21] in GF(2\(^m\)) fields. Basically, bit-level and word-level are two type of multiplication algorithm.

Time independent Montgomery algorithm [42] shows time and space complexity reduction over bit-level time independent Montgomery multiplication algorithm.

Let \(A(x)\) and \(B(x)\) be elements in GF(2\(^m\)) generated by an irreducible polynomial \(F(x)\) of degree \(m\). The set \(\{x^0, x^1, x^2, \ldots, x^{m-1}\}\) is called the polynomial basis. Three terms \(A(x), B(x)\) and \(F(x)\) are expressed as follows:

\[A(x) = a_{m-1}x^{m-1} + \cdots + a_1x + a_0\]  
\[B(x) = b_{m-1}x^{m-1} + \cdots + b_1x + b_0\]  
\[F(x) = x^m + f_{m-1}x^{m-1} + \cdots + f_1x + f_0\]

According to conventional multiplication scheme, \(C(x)\) is given by

\[C(x) = A(x)B(x) \mod F(x)\]

\[
\text{While according to Montgomery multiplication algorithm computes } C(x) \text{ as given by} \]

\[C(x) = A(x)B(x) R^{-1}(x) \mod F(x)\]

where \(R^{-1}(x)\) is the inverse of \(R(x)\) and chosen independently element of GF(2\(^m\)) such that \(\text{GCD}(F(x), R(x)) = 1\). Efficient hardware implementation and complexity depend on the value of \(R(x)\). Let \(R(x) = x^m\) for simplicity, because using \(R(x) = x^m\) only requires ignoring the terms whose powers of \(x\) are greater than or equal to \(m\). Montgomery scalar multiplication using projective coordinates requires up to \((m-1)(6M + 3A + 5S) + (10M + 7A + 4S + I)\) clock cycles, where \(M, A, S,\) and \(I\) represent the number of clock cycles for multiplication, addition, squaring, and inversion, respectively [31]. Montgomery multiplication/squaring with low delay can reduce the overall time complexity of the scalar point multiplication and hence, increase the speed of the elliptic curve processor.
C(x) = A(x)B(x) \cdot X^{-m} \mod F(x), \quad (6)

C(x) = (a_{m-1}x^{m-1} + \cdots + a_1x + a_0)B(x) \cdot X^{-m} \mod F(x) \quad (7)

C(x) = (\ldots((a_0b(x)X^{1} + a_1b(x)X^{-1} + a_2b(x)X^{-1}) + \ldots + a_{m-1}b(x)X^{-1}) \quad (8)

In the time dependent montgomery multiplication algorithm, (equation 8 is used for computation) the latency of this semi-systolic multiplication array requires m clock cycles for the mxm multiplication. Each clock cycle takes delays of two 2-input AND gates, two 2-input XOR gates, and two 1-bit latches. Figure 1 shows detail architecture of time dependent montgomery multiplication algorithm [42].

In the time independent montgomery multiplication algorithm, latency of this semi-systolic multiplication array with mxm size also needs m + 1 clock cycles, but each clock cycle only takes one 2-input AND gate delay, one 3-input XOR gate delay, and one 1-bit latch delay. Figure 2 shows detail architecture of time independent montgomery multiplication algorithm [42]. Comparisons of time and space complexities is given in Table 1.

**Table 1:** Time and space complexities comparison for Time independent and Time-independent MM
### 4. Multiplication Scheme:

The multiplier based on polynomial basis systolic array can be divided into four categories: namely bit serial, bit parallel, hybrid or super serial, digit serial and combination of them. The performance parameter of multiplier is latency, space, power and time complexity. All one polynomial (AOP), trinomials and pentanomial are popular polynomial reported yet for multiplier. Each type of multiplier has its own advantage and specialty.

**[A] Bit-Serial Multiplier**: Only one bit of operand is processed in a bit-serial multiplier and is applicable for small systems, size, cost and bandwidth are major restrictions of such systems. It has minimum area and minimum throughput. Generally, bit-serial multipliers are slow. The bit-serial polynomial basis multipliers proposed in [32] are the classic bit-serial multipliers. Systolic array implementation of the polynomial basis multiplication has started in [33] and [34].

Basically there are two types of bit-serial polynomial basis multipliers i.e. LSB-First and the MSB-First bit-serial polynomial basis multipliers [32]. The use of Montgomery algorithm optimize the parameter performance. So, procedure of LSB-First and MSB-First bit-serial multiplier is also given.

According to conventional multiplication scheme, \( C(x) \) is given by equation (4)

\[
C(x) = A(x)B(x) \mod F(x)
\]

We can write \( C(x) \) as

\[
C = b_{m-1} A x^{m-1} + \ldots + b_1 A x + b_0 \mod F(x)
\]

or

\[
C = b_{m-1} (A x^{m-1} \mod F(x)) + \ldots + b_1 (A x \mod F(x)) + b_0 \mod F(x)
\]

If \( x \) is a root of irreducible polynomial \( f_0 \) and \( f_m = 1 \) of \( F(x) \) that satisfy given below equation

\[
f_m x^m + f_{m-1} x^{m-1} + \ldots + f_1 x + f_0 = 0
\]

\[
x^m = f_{m-1} x^{m-1} + \ldots + f_1 x + 1
\]

In this scheme LSB coordinate of \( B \) i.e. \( b_0 \) is first proceed. Architecture of this scheme is shown in figure 3. Here \( A' \) and \( C' \) are two \( m \)-bit latches, which store values of \( A^{(0)} \) and \( C^{(0)} \), respectively.

**Figure 3: LSB-First bit serial polynomial basis multiplier**

**Figure 4: MSB-First bit serial polynomial basis multiplier**

<table>
<thead>
<tr>
<th>Multipliers</th>
<th>Time-dependent Montgomery Multiplier</th>
<th>Time-Independent Montgomery Multiplier</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-input AND gate delay</td>
<td>2m</td>
<td>m + 1</td>
</tr>
<tr>
<td>2-input XOR gate delay</td>
<td>2m</td>
<td>0</td>
</tr>
<tr>
<td>3-input XOR gate delay</td>
<td>0</td>
<td>m + 1</td>
</tr>
<tr>
<td>1-bit Latch</td>
<td>2m</td>
<td>m + 1</td>
</tr>
<tr>
<td>Total Delays (unit delays)</td>
<td>16mΔ</td>
<td>(8m + 8)Δ</td>
</tr>
<tr>
<td>2-input AND gate</td>
<td>2m²</td>
<td>2m² + 3m</td>
</tr>
<tr>
<td>2-input XOR gate</td>
<td>2m²</td>
<td>0</td>
</tr>
<tr>
<td>3-input XOR gate</td>
<td>0</td>
<td>m² + m</td>
</tr>
<tr>
<td>1-bit Latch</td>
<td>4m²</td>
<td>3m² + 4m</td>
</tr>
<tr>
<td>Total Transistors</td>
<td>72m²Ω</td>
<td>(64m² + 78m)Ω</td>
</tr>
</tbody>
</table>
\[(a^{(i)} x^m + \ldots + a^{(i)} x^2 + \cdots + a_0^{(i)} x) \mod F(x) \]  

(13)

With the help of (12) and (13)

\[A^{(i+1)} = (a_{m-2}^{(i)} + a_{m-1}^{(i)} f_{m-1} ) x^{m-1} + (a_{m-3}^{(i)} + a_{m-2}^{(i)} f_{m-2} ) x^{m-2} \ldots + (a_0^{(i)} + a_{m-1}^{(i)} f_1 ) x^1 + a_m^{(i)} \]  

(14)

this module multiplies \(A^{(i)}\) by \(x\) and reduces the results by \(F(x)\).

With application of Horner’s rule equation (9) can be written as

\[C = \ldots (b_m A x \mod F(x) + b_{m-1} A x \mod F(x) + \ldots + b_1 A x \mod F(x) + b_0 A) \]  

(15)

In MSB –First bit serial scheme \(b_{m-1}\) is first proceed (equation 15). Architecture of MSB –First bit serial is shown in Figure 4.

[1] Bit-Serial Montgomery Multiplier : As Montgomery multiplication is given as \(C(x) = A(x) B(x) X^{-m} \mod F(x)\), but taking general consideration and choosing \( R = x^u, 1 \leq u \leq m \)

\[C = b_m A x^m + b_{m-1} A x^{m-1} + \ldots + b_1 A x^1 + b_0 A \]  

(16)

\[f_m x^m + f_{m-1} x^{m-1} + \ldots + f_1 x + f_0 = 0 \]  

(17)

where \( x \) is root of polynomial \( F(z) \). With condition of irreducible polynomial \( (f_0 = f_m = 1) \).

\[x^{-1} \mod F(x) = x^{m-1} + \ldots + f_2 x + f_1 \]  

(18)

Taking \( R = x^u, 1 \leq u \leq m \) and \( R = x^{m-1} \), and using equation (19), we can design the Bit-Serial Montgomery Multiplier.

[2] MSB-First Bit-Serial MM: In this scheme MSB of \( B \) i.e. \( b_{m-1} \) is first considered with one bit at each cycle. So, \( C \) is given by

\[C = b_{m-1} A x^{m-1} + \ldots + b_1 A x^1 + b_0 A \]  

(19)

so, according to MM scheme, there is requirement of pre-computation i.e. Ax^{m-1} \mod F(x) and complexity of scheme depends upon pre-computation complexity.

With use of Horner’s rule and pre-computation factor i.e. \( A^{(0)} = Ax^{m-1} \mod F(x) \),

\[A^{(i+1)} = A^{(i)} \mod F(x) \]

\[(a_{m-2}^{(i)} x^{m-2} + \ldots + a_1^{(i)} + \cdots + a_0^{(i)} x^{-1}) \mod F(x) \]  

(21)

From equation (19 and 21)

\[A^{(i+1)} = a_0^{(i)} x^{m-1} + (a_{m-1}^{(i)} + a_0^{(i)} f_{m-1} ) x^{m-2} \ldots + (a_2^{(i)} + a_0^{(i)} f_2 ) x^2 + (a_1^{(i)} + a_0^{(i)} f_1 ) \mod F(x) \]

Figure 5: MSB-First bit serial Montgomery multiplication (MM) using \( R = x^u \)

LSB of \( B \) is First proceed in LSB –First MM scheme. Table 2 provides the comparison result.

Table 2: Parameter comparison of Bit-serial multiplier over GF \((2^m)\)
For $F(z) = f_m z^m + f_{m-1} z^{m-1} + \cdots + f_1 z + f_0$

<table>
<thead>
<tr>
<th>Type</th>
<th>Algorithm</th>
<th>#XOR</th>
<th>#AND</th>
<th>#Flip</th>
<th>Latency</th>
<th>C.P.D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polynomial basis</td>
<td>LSB-First</td>
<td>2m-1</td>
<td>2m-1</td>
<td>2m</td>
<td>m</td>
<td>$T_A + T_X$</td>
</tr>
<tr>
<td></td>
<td>MSB-First</td>
<td>2m-1</td>
<td>2m-1</td>
<td>2m</td>
<td>m</td>
<td>$T_A + 2T_X$</td>
</tr>
<tr>
<td>Montgomery multiplication</td>
<td>LSB-First</td>
<td>2m-1</td>
<td>2m-1</td>
<td>2m</td>
<td>m ($u = m$)</td>
<td>$2T_A + 2T_X$</td>
</tr>
<tr>
<td></td>
<td>MSB-First</td>
<td>2m-1</td>
<td>2m-1</td>
<td>2m</td>
<td>m ($u = m-1$)</td>
<td>$T_A + T_X$</td>
</tr>
<tr>
<td></td>
<td>LSB-First</td>
<td>2m-1</td>
<td>2m-1</td>
<td>2m</td>
<td>m ($u = m-1$)</td>
<td>$T_A + 2T_X$</td>
</tr>
</tbody>
</table>

**[B]** Bit-Parallel Polynomial Basis Multiplication: In Bit-Parallel Polynomial Basis Multiplication, operand are proceed in parallel. This polynomial multiplication scheme can be improve their performance using special irreducible polynomial such as all-one polynomials, trinomials, and pentanomial. In general bit-parallel polynomial basis multiplier is given by $C = b_0 A + b_1 A x + \cdots + b_{m-1} A x^{m-1} \mod F(x)$. It is composed of some x-modules which multiply their inputs by x and reduce the results by $F(x)$ [35]. Figure 6 shows architecture of Bit-Parallel polynomial basis multiplier.

![Figure 6: Conventional Bit-Parallel polynomial basis multiplier](image)

**[1]** Bit-Parallel MM: In general MM can be formulated as $C = A \cdot B \cdot R^{-1} \mod F(x)$, where $R$ can be chosen as $R = x^u$, $0 < u \leq m$. Using equation (16) and (17) we can get a new architecture of the bit-parallel Montgomery multiplier for $u = m$. In this architecture, the AND modules multiply a field element by a bit, whereas the XOR modules add two field elements. The architecture shown in figure 7.

![Figure 7: Bit-parallel Montgomery multiplier for R=x^m](image)

This architecture of figure 7 is similar to figure 6.( bit-parallel polynomial basis multiplier. But in this case i of x^i-modules is used. Here important thing is that order of processing the coordinates of B is reverse. If u have the range of $[1, m-1]$, we can rewrite the Montgomery multiplication as

$$C = b_0 A x^u + b_1 A x^{u+1} + \cdots + b_{u-1} A x^{u-1} + b_u A + b_{u+1} A x + \cdots + b_{m-1} A x^{m-u-1} \mod F(x)$$

In this case, the main difference is that we multiply A by negative and positive powers of x to calculate the terms in (22). We can rewrite (22) as $C = C_1 + C_2$, Where $C_1 = b_0 A x^u + b_1 A x^{u+1} + \cdots + b_{u-1} A x^{u-1} \mod F(x)$ and $C_2 = b_u A + b_{u+1} A x + \cdots + b_{m-1} A x^{m-u-1} \mod F(x)$. Now,
we can design the new architecture of the general case of the MM with \( r = x^u \) as depicted in Figure 8.

Figure 8: bit-parallel Montgomery multiplier over \( GF(2^m) \) with \( R=x^u, 1 \leq u \leq m - 1 \)

Note that for \( 1 \leq u \leq m - 1 \), the number of the \( x \) and \( x^{-1} \)-modules is \( m - 1 \), as \( b_u A \) is obtained directly from \( A \).

Based on the architecture depicted in Fig. 8, the first step of the multiplication is to compute the terms \( Ax^i \mod F(x) \), for \( i \in [-u, m-u-1] \). Here, we use \( A \) to represent \( Ax^i \mod F(x) \). This can be done by using the matrix \( M \), whose columns show the PB representation of \( A \) for \( i \in [-u, m-u-1] \). So, the matrix \( M \) has \( m \) rows and \( m \) columns. Then, the MM over \( GF(2^m) \) can be formulated as

\[
[C_0, C_1, ..., C_{m-1}]^T = M[ b_0, b_1, ..., b_{m-1}]^T
\]

(23)

It is similar to the as Mastrovito multiplication [36].

[2] Bit-Parallel Montgomery Multiplier for Irreducible Trinomials

Let \( F(z) = z^m + z^k + 1 \) be an irreducible trinomial and \( x \) be the root of \( F(z) \). Then, the Montgomery factor \( r = x^u \) is obtained from the following in order to design a fast Montgomery multiplier.

\[
\begin{align*}
1, & \quad k = 1, \\
U = \{k \text{ or } k-1, & \quad k > 1
\end{align*}
\]

(24)

Entries of the matrix \( M \) will be the additions of at most two terms. that the elements of the matrix \( M \) are summations of at most two terms if \( k - 1 \leq u \leq k \). This scheme requires \( m^2 \) AND gate, \( m^2 - \frac{m}{2} \) \( k = \frac{m}{2} \) (two-input XOR gate) and \( m^2 - 1, k = \frac{m}{2} \) (two-input XOR gate). Delay is given by

\[
\begin{align*}
T_A + [\log_2(2m - u - 1)]T_X, & \quad u \leq \frac{m-1}{2} \\
T_A + [\log_2(m + u)]T_X, & \quad u > \frac{m-1}{2}
\end{align*}
\]

(25)

[3] Bit-Parallel Montgomery Multiplier for Irreducible Pentanomials

Irreducible pentanomials are special form of irreducible polynomial which are used in finite field arithmetic, e.g., [37], [38], [39], [40], and [41]. Generally, they can be formulated as

\[
F(z) = z^m + z^{k_1} + z^{k_2} + z^{k_3} + 1; 1 \leq k_1 < k_2 < k_3 < m
\]

(26)

We assume that \( R = x^u \) is the Montgomery factor. For design of bit-parallel Montgomery multipliers matrix \( M \) plays an important role. If each column of the matrix \( M \) is computed with one step of reduction, then the matrix \( M \) can be obtained faster. In this regard, we use a special type of irreducible pentanomial, known as type-II irreducible pentanomial, is defined [41], as

\[
F(z) = z^m + z^{m+2} + z^{m+1} + z^{n+1}, \quad \text{where} \quad 2 \leq n \leq \frac{m}{2} - 1
\]

(26)

Basically, this approach lead to Fast Montgomery Multiplier (FMM) and Low Complexity Montgomery Multiplier.
(LCMM) which requires less area. It can be obtained by application of Montgomery multiplication scheme as discussed above. The final result of this scheme is given below.

Let \( R = x^n \) (\( u = n \)) be the Montgomery factor. The fast bit-parallel Montgomery Multiplier (FMM) using type-II irreducible pentanomial of degree \( m \) requires time complexity of \( T_A + (1 + \lfloor \log_2(m + n) \rfloor) T_X \), if \( n > \frac{m-1}{2} \) and \( T_A + (1 + \lfloor \log_2(2m - n-2) \rfloor) T_X \), if \( n < \frac{m-1}{2} \), also it requires \( m^2 \) two-input AND gates and \( m^2 + 3m - 9 \) two-input XOR gates.

Again let \( R = x^n \) (\( u = n \)) is used as the Montgomery factor, the low complexity bit-parallel Montgomery multiplier (LCMM) using the type-II irreducible pentanomial \( F(z) \) requires time complexity of \( T_A + (1 + \lfloor \frac{\log_2 (2m - 4u - 9)}{2} \rfloor) T_X \), if \( u < \frac{m-1}{2} \) and \( T_A + (1 + \lfloor \frac{\log_2 (2m - 4u - 9)}{2} \rfloor) T_X \), if \( u > \frac{m-1}{2} \). Also it requires \( m^2 \) two-input AND gates and \( m^2 + 2m - 3 \) two-input XOR gates.

A low latency systolic montgomery multiplier for Finite Field \( GF(2^m) \) based on Pentanomials is proposed by P.K.mehar[43] in which pre-computed addition technique is used and this design have low latency less area-delay and power-delay complexities.

[C] Digit-Serial Polynomial Basis Multiplication

In a digit-serial multiplier, the bits are grouped as digits and at each cycle, one digit is processed. In Digit-serial multiplication scheme \( m \) bit word is broken into \( n = \lceil m/D \rceil \) digits. By choosing different size of digit, it can be possible to compensate the gap between the speed and the amount of required hardware, with increment of digit size more hardware is required. In general we will start from the LSB of the operand \( B \), i.e., \( b_0 \), and group \( D \) consecutive bits as a digit, where \( D \geq 2 \) to be the digit size. So

\[
B = \sum_{i=0}^{n-1} B_i \cdot x^i
\]

where \( B_i \) is given as

\[
B_i = \begin{cases} \sum_{j=0}^{n-1-Di} b_{Di+j} \cdot x^j, & 0 \leq i \leq n-2 \\ \sum_{j=n-1-Di}^{m}, & i=n-1 \end{cases}
\]

So according to polynomial basis multiplication scheme \( C \) is given as

\[
C = A \cdot \sum_{i=0}^{n-1} B_i \cdot x^i \mod F(x)
\]

\[
= AB_{n-1} x^{(n-1)D} + \ldots + AB_1 x^D + AB_0 \mod F(x)
\]

As above it can also divided into LSD-First digit-serial polynomial basis multiplication and MSD-First digit-serial polynomial basis multiplication. In first case \( B_0 \) and \( B_{n-1} \) is proceed in later case. The equation for LSD-First digit-serial polynomial basis multiplication and architecture (Figure 9) is given as

\[
C = B_{n-1} (Ax^{(n-1)D} \mod F(x)) + \ldots + B_1 (Ax^D \mod F(x)) + B_0 \mod F(x)
\]

Figure 9: The LSD-first digit-serial polynomial basis multiplier

This multiplier requires latency of \( n + 1 \) clock cycles and critical path delay \( D(T_A + T_X) \), \( D(3m-2)-m+1 \) two-input AND gates and \( Dx(3m-2)-m+1 \) two-input XOR gates and \( (2m+D-1) \) latches.
And for MSD-First digit-serial polynomial basis multiplication equation and architecture (Figure 10) is given as
\[ C=((B_{n-1}A \mod F(x)) x^D + B_{n-2}A) \mod F(x) x^D + \ldots + B_1A) \mod F(x) x^D + B_0A \mod F(x) \] (33)

Figure 10: The MSD-first digit-serial polynomial basis multiplier

This multiplier requires latency of \( n + 1 \) clock cycles and critical path delay \( D(T_A + T_X) + T_X \cdot D \cdot (3m - 2) \) two-input AND gates and \( D \cdot (3m - 2) \) two-input XOR gates and \( (2m + D - 1) \) latches.

We can improve the performance of Digit serial multiplication by using Shifted polynomial Basis (SPB) scheme. Using Shifted polynomial basis scheme and using equation (4) SPB multiplication is given as,
\[ C = B_0 A x^{-v} + B_1 A x^{-(n-2)} + \ldots + B_{n-1} A x^{-(n-1)D-v} \mod F(x) \] (34)

By choosing the appropriate value of \( v \), we may lead different technique (algorithm). If we choose \( v = (n-1)D \), above equation can be written as
\[ C = B_0 A x^{-v} + B_1 A x^{-(n-2)} + \ldots + B_{n-1} A \mod F(x) \] (35)

It leads to MSD Digit-Serial multiplication scheme in which \( B_{n-1} \) is first proceeded.

Figure 11: The MSD-first digit-serial SPB multiplier.

Using equation (34) Shifted polynomial basis multiplication can be written as
\[ C = B_0 A x^{n} + B_1 A x^{D-v} + B_{[n/2]} A x^{[n/2]D-v} + \ldots + B_{n-1} A x^{-(n-1)D-v} \mod F(x) \] (36)

By choosing \( v = [n/2]D \),
\[ C = B_0 A x^{([n/2]D)} + B_1 A x^{D-[n/2]D} + \ldots + B_{[n/2]} A x^{D-[n/2]D} + B_{[n/2]+1} A x^{D} \ldots + B_{n-1} A x^{-[n-2]D} \mod F(x) \] (37)

It leads in to two parts i.e. positive part of \( x \) and negative part of \( x \).

\[ C = C^+ + C^- \] where \( C^+ \) is digit serial –SPB with \( [n/2] \) of LSD of \( B \), while other is \( n-[n/2] \) MSD of operand of \( B \).
\[ C = B_0 A x^{[n/2]D} + B_1 A x^{D-[n/2]D} + \ldots + B_{[n/2]-1} A x^{D} \mod F(x) \] (38)
Equation (37, 38, 39) represent Hybrid Digit-Serial SPB Multiplication scheme. By using different basis, we can get better result. Table 3 provides comparison result of Time complexity for Digit-Serial multiplier over GF (2^m)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Type</th>
<th>Critical Path delay</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSD-first</td>
<td>SPB</td>
<td>D(T_A + T_X)</td>
<td>n + 1</td>
</tr>
<tr>
<td>Hybrid</td>
<td>SPB</td>
<td>D(T_A + T_X)</td>
<td>[n/2] + 2</td>
</tr>
</tbody>
</table>

This result can improve using systolic and semi-systolic architecture, using digit-serial systolic multiplication scheme (montgomery algorithm) over all-one-polynomial (AOP) over GF(2^m) have latency (2N-1) clock cycle, where N=[m/L], m is the word size and L is the digit size [43].

5. Conclusion:
This paper covers architectures and multiplication scheme for binary finite field for implementing the elliptic curve cryptography. Here particularly, we focused on polynomial basis, but include trinomial, pentanomial and all-one-polynomial. We gave a comprehensive summary of finite field arithmetic in cryptography, covering all multiplication scheme and algorithm in order to create time and space efficient implementation of finite field operation. There is a lot of possibility to improve the performance of cryptographic system by using semi-systolic model, special polynomial, montgomery algorithm and different bases. It is also possible to create single hardware architecture that supports several different field and bases at a cost of improvement in multiplication algorithm.

Reference:


[44] Jiafeng Xie, Jian jun He, and Pramod Kumar Meher, “Low Latency Systolic Montgomery Multiplier for Finite Field GF(2^m) Based on Pentanomials”, IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, VOL. 21, NO. 2, FEBRUARY 2013