Efficient Elliptic Curve Point Multiplication with Montgomery Ladder Algorithm

Anita Aghaie, Student Member, IEEE

Term-paper of Advanced Cryptography course, Prof. Stanislaw P Radziszowski

Abstract—Scalar point multiplication has encountered significant attention in Elliptic curve cryptography (ECC) which is gaining popularity due to providing same level security with smaller key sizes compared to traditional cryptosystems, such as Ron Rivest, Adi Shamir, and Leonard Adleman (RSA). Point multiplication (KP) in ECC is basically performed on point addition and point doubling on elliptic curves. This performance has applied in different approaches such as NAF method, Montgomery ladder algorithm, binary method that is just point addition and point doubling. In this paper, improving an algorithm of public key cryptography, Montgomery Ladder, goes under review to make an efficient Elliptic Curve point multiplication in terms of area and throughput. This algorithm computes point multiplication on Elliptic Curves such as generic curves in which it is optimized by using parallel multipliers in Digit size architectures.

Index Terms—Elliptic curve, Montgomery Ladder algorithm, point multiplication, projective coordinate.

I. INTRODUCTION

The security of Elliptic Curve Cryptography relies on the difficulty of solving the Elliptic Curve Discrete Logarithm Problem ECDLP, which states that given an elliptic curve $E$ defined over a finite field $F_q$, a point $P \in E(F_q)$ of order $n$, and a point $Q \in E(F_q)$, find the integer $k \in [0, n-1]$ such that $Q = kP$ [1]. The integer $k$ is called the discrete logarithm of $Q$ to the base $P$, denoted $k = \log_P Q$. This point multiplication is performed by repeated point addition and point doubling; for example $7P = (2((2P) + P) + P$. The scalar $k$ is used as a private key and curve’s base point $P$ is used as a public key. For ECC, we only consider those points which lie in some finite fields. To develop the fast elliptic curve multiplication, the less known algorithm of Montgomery ladder are used in the computations and this algorithm has full of generality and applies to all abelian groups [2], [3], [4]. Briefly, this algorithm is just performing multiplication point in ECC on x-coordinate of the points; therefore, it has led us to reduction the number of field operations required at each iteration in the main loop. As a matter of the mentioned fact, the Montgomery Ladder Algorithm improves the efficiency and security issues by reduction of memory requirements for elliptic curve computations and constant execution time, respectively [2].

This paper is organized as follows in which we will have a short preliminary about ECC and coordinate systems. In section III, we will verify different methods of point multiplication and clarify them by examples. In IV section, two representation of Montgomery Ladder in affine and projective coordinate will be discussed. The efficient Montgomery Ladder and its security analysis is organized in other sections. At the end we have a brief conclusion for this term paper.

II. PRELIMINARY

ECC provides an unique properties in mathematical structure which is adding two points on a specific curve and getting the result point on the same curve. This specific feature gives us an opportunity to use ECC in the cryptography due to the difficulties of finding our private key and breaking this protocol needs the advanced mathematics [5]. Elliptic curves over $GF(2^m)$ are particularly appealing due to the fact that the binary field operations are more efficiently implemented in hardware and software [6]. Here, we present a brief introduction to elliptic curves and a short explanation about the coordinate systems which are usable in the basic cryptography operations.

a. ECC

Elliptic curve cryptography is defined on elliptic curves with the group of points on these curves over finite field like $q$ which the general form of these curves are shown as following.

$$y^2 + (a_1)x.y + (a_3)y = x^3 + (a_2)x^2 + (a_4)x + a_6$$  

(1)

There are some elliptic curves such as (Eq.1) with $E$ that is simplified compared with the first equation Weierstrass equation [1]. In these curves, the neutral point $O$ is infinite ($\infty$) and it is based on cyclic group mathematics structure.

$$y^2 + x.y = x^3 + a.x^2 + b.$$  

(2)

In this equation, if the characteristic of $q$ is equal to 2 or 3 and the $a,b$ are members of $q$ and be not equal zero, such curve is called nonsupersingular [7]. The set of points on this curves satisfied Eq.1 form the group operation such as addition and multiplication to create the primitive operation of ECC. In other words, ECC is a public key cryptography method based on group of points on an elliptic curve over finite field. Elliptic curve operations such as point addition (PA) and point doubling (PD) are built up from finite field primitives. Arithmetic over finite field determines the efficiency of an
ECC-based cryptosystems. As shown in Fig. 1, the protocol of elliptic curve cryptography is based on lower layers such as field operation and curve operations. To optimize and fasten each of the ECC protocols, we should optimize the lower layer that here, we will focused on the scalar multiplication part.

b. Coordinate systems

In ECC, cryptosystems are shown via various coordinates to show their points such as affine coordinates, Projective (standard) coordinates, Projective (Jacobian) coordinates. Due to the field operations required in PA and PD such as inversion and division, an appropriate coordinate systems should be chosen. For example, PA and PD in affine coordinates need the inversion and division which are the most area and time-consuming operation in hardware implementation. Therefore the projective coordinate especially Jacobian coordinates that represent a point on the curve with three factors. (P=(x, y, z) where x, y, z \in E (GF (q)), are used to reduce the expense of division. Although, the division operation to convert back from Jacobian coordinates to affine coordinate is needed; this point arithmetic operation import less expense [7].

III. SCALAR POINT MULTIPLICATION

One of the ECC protocol’s layer is the point multiplication, Q = kP that we will verify this multiplication over binary field. In the common key generation algorithm and Diffie-Hellman algorithm, one scalar multiplication is used as a secret key; in addition, Schnorr signature verification needs double-scalar multiplication like \( k_1P_1 + k_2P_2 \) as a public scalar. This scalar point multiplication which is the combination of point addition (PA) and point doubling (PD), is performed in the various methods such as double and addition in binary method, window method, NAF and wNAF method, sliding-window method, and Montgomery Ladder algorithm. It is notable to mention that the concern in the computing scalar multiplication is timing information in which the computation \( kP \) should have the same result for public or for secret k [1].

Some fast scalar-multiplication algorithms have a running time that depends on \( k \); therefore, an attacker can measure time and deduce information about \( k \). For instance, Brumley, Tuveri, 2011[8] shows us that it is required a few minutes to steal the private key of a TLS server over the network. Consequently, we need constant-time algorithms for computing the secret key \( k \).

To explain more practical aforementioned methods of scalar multiplication, we have just applied a same example in three following methods. Assume the points of P and Q are on an Elliptic curve and k which is scalar for point multiplication, is equal 19. So the purpose is calculating of \( Q = 19P \).

Actually, the binary representation of \( k \) scalar is used as \( k = k_{m-1}2^{m-1} + k_{m-2}2^{m-2} + \ldots + k_12^1 + k_02^0 \).

1) Double and Addition in binary method. In this part for each \( k \), we should write the binary form of \( k = 19 \) to compute the addition and doubling operation for each bit. In the ECC, we have a doubling for each bit of ‘0’ and ‘1’, and for each bit ‘1’ we have an addition operation, too. As shown in this method, the time complexity is \( k/2 \) generally due to Hamming weight of \( k \) that is the number of bit ‘1’ on average.

\[
K = 19 = (10011)_2
\]

| k | P | initializing \\
|---|---|---|
| 1 | p | \\
| 0 | 2p | doubling \\
| 0 | 4p | doubling \\
| 1 | 9p | doubling and addition \\
| 1 | 19p | doubling and addition |

2) Non adjacent form (NAF) addition-subtraction. The purpose of this method is to create a binary form of \( K \) that are not adjacent nonzero bit together. In other words, two close bit ‘1’ like”11” in binary form should be changed to “10-1” which has create the \( k/3 \) time complexity in general [9]. NAF algorithm for this \( K \) is as following:

\[
19 = (10011)_2 = (1010 - 1)_2
\]
3) **Montgomery ladder Algorithm.** This method is based on the binary method and the known difference of x-coordinate of two points \( P_1 \) and \( P_2 \) [10]. In this method likewise the first approach, the point addition and point doubling are performed for each bit but here two variables are computed which the difference of these variables in each step (loop) is even that the advantages of this feature will be described later. In fact, the advantage of this method is encountering just one coordinate, x-coordinate in computations (affine coordinate) or using x and z coordinate in projective coordinate systems [7]. Finally, the last step of computation is shown the y-coordinate. Montgomery ladder in this k:

### Table II

**NON ADJACENT FORM (NAF) ADDITION-SUBTRACTION**

<table>
<thead>
<tr>
<th>k</th>
<th>A</th>
<th>B</th>
<th>A-B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>p</td>
<td>2p</td>
<td>p</td>
</tr>
<tr>
<td>0</td>
<td>2p</td>
<td>3p</td>
<td>p</td>
</tr>
<tr>
<td>1</td>
<td>9p</td>
<td>10p</td>
<td>p</td>
</tr>
<tr>
<td>-1</td>
<td>19p</td>
<td>20p</td>
<td>p</td>
</tr>
</tbody>
</table>

As seen in the above example, the algorithm of Montgomery ladder is applied as follow.

#### Algorithm 1 Montgomery Ladder Algorithm of point multiplication in ECC [7]

- **Input:** An integer \( k \geq 0 \) and a point \( P = (x, y) \in E \).
- **Output:** \( Q = kP \).

1. if \( k = 0 \) or \( x = 0 \) then output \((0,0)\) and stop.
2. Set \( k \leftarrow (k_{l-1}k_{l-2}...k_0)_2 \).
3. Set \( x_1 \leftarrow x, x_2 \leftarrow x^2 + b/x^2 \).
4. for \( i \) from \( l - 2 \) down-to 0 do
   - set \( t \leftarrow x_1^{t^2+t} \).
   - if \( k_i = 1 \) then
     - set \( x_1 \leftarrow x + t^2 + t, x_2 \leftarrow x_2^2 + t^2 + t \).
   - else
     - set \( x_1 \leftarrow x_1^2 + b/x_1^2, x_2 \leftarrow x + t^2 + t \).
5. Set \( r_1 \leftarrow x_1 + x, r_2 \leftarrow x_2 + x \).
6. Set \( y_1 \leftarrow r_1(r_1r_2 + x^2 + y)/x + y \).
7. return \((Q = (x_1, y_1))\).

This aforementioned algorithm gives us three sufficient features that leads us to have easier implementation. The first feature is the invariant value of B-A throughout the algorithm which is equal to p. The other features is being independent of two operations of point addition and point doubling in each iteration that has led to parallel computation in software implementation by using two processors. The last but not the least features is using two common operands for each point addition and point doubling in each iteration that causes the more efficient hardware implementations [6].

#### IV. DIFFERENT COORDINATE OF MONTGOMERY LADDER ALGORITHM

In this part, we verify the Montgomery Ladder Algorithm in two common coordinates: affine coordinate and standard projective coordinate. As mentioned in last sections, the affine coordinate systems enforced many division and inversion costs in group operations in ECC. Two field inversions is the cost of using affine coordinate for Montgomery Ladder algorithm which happen in each iteration as shown in the step 4 of Algorithm 2 [6]. Therefore, the projective coordinate would be more useful in such algorithms compare to affine coordinate, i.e, Montgomery Ladder substitute the most multiplications instead of divisions or inversions. Actually, the x-coordinate of each point is represented by x-coordinate over z-coordinate in projective system. Consequently, the number of multiplication for Montgomery Ladder is increased from 4 in affine coordinate to 6 in projective one where as the number of field inversion is decreased from 2 in affine to just one in new coordinate.

#### Algorithm 2 Montgomery Ladder Algorithm in affine coordinate [6]

- **Input:** An integer \( k \geq 0 \) and a point \( P = (x, y) \in E \).
- **Output:** \( Q = kP \).

1. if \( k = 0 \) or \( x = 0 \) then output \((0,0)\) and stop.
2. Set \( k \leftarrow (k_{l-1}k_{l-2}...k_0)_2 \).
3. Set \( x_1 \leftarrow x, x_2 \leftarrow x^2 + b/x^2 \).
4. for \( i \) from \( l - 2 \) down-to 0 do
   - set \( t \leftarrow x_1^t \).
   - if \( k_i = 1 \) then
     - set \( x_1 \leftarrow x + t^2 + t, x_2 \leftarrow x_2^2 + t^2 + t \).
   - else
     - set \( x_1 \leftarrow x_1^2 + b/x_1^2, x_2 \leftarrow x + t^2 + t \).
5. Set \( r_1 \leftarrow x_1 + x, r_2 \leftarrow x_2 + x \).
6. Set \( y_1 \leftarrow r_1(r_1r_2 + x^2 + y)/x + y \).
7. return \((Q = (x_1, y_1))\).

The standard projective coordinate or mixed coordinate which is the same as standard projective coordinate with the difference \( z = 1 \) to mix with affine coordinates, are used in this algorithm; therefore, the new version of \( x_A = X_A/Z_A \) and \( x_B = X_B/Z_B \) have been substituted in the addition and doubling equations [7], [6]. By using this conversion, the following equations are provided for point addition and point doubling which not include any division or inversion. The projective Montgomery algorithm written by these new coordinates are presented in some research papers [6], [7] to find the \( Q = kP \) with the standard coordinate. The key point of these algorithms of theses papers is that just only one inversion in the end of the algorithm is required to achieve the y-coordinate of the final point. For instance, in Algorithm 3, the last function, \( Mxy \), contains just one inversion; whereas, other functions such as \( MAdd \) and \( MDouble \) which are the point addition and point doubling function, do not have any inversions [6].
Algorithm 3 The Montgomery ladder point computation in projective coordinate[6]

Input: An integer \( k \geq 0 \) and a point \( P = (x, y) \in E \).
Output: \( Q = kP \).

1. if \( k = 0 \) or \( x = 0 \) then output \((0,0)\) and stop.
2. Set \( k \leftarrow (k_l \ldots k_2 k_1 k_0)2 \).
3. Set \( X_1 \leftarrow x, Z_1 \leftarrow 1, X_2 \leftarrow x^4 + b, Z_2 \leftarrow x^2 \).
4. for \( i \) from \( l - 2 \) down-to 0 do
   if \( k_i = 1 \) then
      \( M_{add}(X_1, Z_1, X_2, Z_2), M_{double}(X_2, Z_2) \).
   else
      \( M_{add}((X_2, Z_2, X_1, Z_1), M_{double}(X_1, Z_1). \)
   end if;
end loop;
5. if \( z_B = 0 \) then \( x_A := xp, z_A := yp \);
else
   \( x_A := x_A/z_A; \)
   \( z_B := x_B/z_B \).
end if;
6. return \( (Q = M_{xy}(X_1, Z_1, X_2, Z_2)) \).

V. EFFICIENT MONTGOMERY LADDER ALGORITHM

To optimize the ECC protocol, optimization of point multiplication plays an important role. Here, we are focusing on scalar multiplication over binary field in polynomial basis that creates opportunities such as improving the field operations (modular multiplication, modular division, modular squaring, and etc.) and exploiting an optimum algorithm (optimized Montgomery Ladder with using parallelism multiplications in the algorithm). The following figure show us, we can make a more efficient Montgomery Ladder algorithm by using some tools such as converting in projective coordinate, precomputation some parameters of projective algorithm, and using generic curves because Montgomery Ladder has less efficiency in other curves such as Edward curve.

As shown in Fig. 2, the left and right algorithm are Montgomery Ladder algorithm in projective coordinate but the right one is the optimum algorithm which is used the mentioned tools to make an efficient algorithm. Two multiplier as \( T_1 \) and \( T_2 \) are defined to perform each steps of loop that create the \( x \) and \( z \) coordinates and in the last if condition the \( y \)-coordinate is determined with three divisions and some addition. After executing the left algorithm, the main loop have \( m \) iteration that consumes \( m \times T \) multiplication and after that it added with the time of three division and two addition that create the time of \( 5m^2 + 8m \) [7]. However, the the execution time of the right one is reduced as \( 2m^2 + 4m \) because of using just three parallel multipliers and one inversion. The hardware implementation results of this efficient algorithm were presented in [7] to confirm the optimization of its performance in terms of time and energy consumption.

Furthermore, we can optimize the left algorithm by optimizing the field operation. In finite field operation specifically over binary field, multiplication is used in most computation. For example, there are many exponentiation in the point addition and doubling, in that the following equation (Eq.3) shows the addition of two points in the binary form in each coordinate and + with circle means XOR bitwise [4]. Hence, multiplication has a significant role in the arithmetic operation in particular cryptography systems. (Eq.3)

\[
\sum k_i a_i + \sum k_i b_i = \sum k^i (a_i \oplus b_i)
\] (3)

Different types of multipliers are in arithmetic but two main categories are involved in binary field: parallel computation and bit level computation. In parallel multiplication, the factor of fast computation and low latency (making results in one clock cycles) in spite of using high area observe the designer to apply it. Conversely the bit level multiplier are applied in existence of resources’ constrains; however, they have high latency (making results in \( m \) bit clock cycles). According to these classification there are three multiplier architecture that the digit level is a medium architecture in this spectrum. Digit-serial multiplication algorithms provide a flexible area and
time complexity. By increasing the digit size, the algorithm becomes faster but requires more hardware resources [11].

VI. SECURITY ANALYSIS

As mentioned before, one of the most challenging problem in the computation of point multiplication is timing time that will be important for secret key computation. As a matter of this fact, the algorithm with constant execution time is appropriate to solve this challenging issue. Regarded to the first mentioned feature of Montgomery Ladder algorithm that is the constant difference between two parameters, the execution time of this algorithm is also constant. As a result, this feature give us an important option of side channel resistant because the side channel attacks are able to deduce information such as secret key here by observing time or power consumption depend on the structure of algorithm [2]. Moreover, this algorithm is high secure against the different fault attacks such as C safe-error attack and M safe-error attack that the detail of this security was analyzed in [2].

VII. CONCLUSION

In this term paper, different methods of point multiplication for ECC were mentioned. One of the point multiplication algorithm, Montgomery Ladder algorithm, was verified to improve the security and speed of the elliptic curve protocol in different protocols such as Diffie-Hellman and Signature digital algorithm. The features of this algorithm give an option to use the projective coordinate and make this algorithm more efficient in hardware and software algorithms by using parallelism concept. This algorithm also reduces the memory requirements and make an easy hardware and software implementation. In addition, the constant execution time of Montgomery Ladder has caused resistant against side channel attack and fault attacks.

REFERENCES