Embedment Of Montgomery Algorithm On Elliptic Curve Cryptography Over RSA Public Key Cryptography

Aswathi Thomas\textsuperscript{a,*}, Ebin M. Manuel\textsuperscript{b}

\textsuperscript{a}P G Scholar, Department of ECE, Government College of Engineering Idukki, Kerala, India
\textsuperscript{b}Assistant Professor, Department of ECE, Government College of Engineering Idukki, Kerala, India

Abstract

The Elliptic Curve cryptosystem has replaced the prevailing first generation public key algorithms – RSA and Diffie-Hellman due its shorter key size requirement. This paper presents a modified Elliptic Curve and RSA cryptosystem by incorporating a newly designed Montgomery multiplier algorithm for better efficiency. The inherent disadvantage of delay due to the large number of computations in Elliptic Curve cryptography is improved substantially by the implantation of the modified Montgomery algorithm in it. The simulation results show significant improvement in terms of speed and power.

1. Introduction

With the increase in data communication and the expansion of internet services like email, secure telephony mobile internet, e-commerce, e-banking and so on, security issue has become most important over the networks. Cryptographic systems can provide the objectives on information security: confidentiality, data origin authentication, data integrity and non-repudiation. In contrast to symmetric-key cryptosystems, public-key cryptosystems are capable of fulfilling all of the given objectives. Cryptology is the science concerned with providing secure communications. All malicious attempts to access information is prevented [1].

An authorized access is identified by a cryptographic key. There are two types of cryptographic algorithms — Symmetric key and Asymmetric key algorithms. Symmetric key cryptographic algorithms have a single key for both encryption and decryption. The asymmetric key cryptographic algorithms involve a private key and a public key. The encryption is done with the public key and the encrypted message can be only decrypted by the corresponding secret private key. Although slow and highly complex, asymmetric key cryptosystem has immense advantages. The
main advantage is that the underlying primitives used are based on well-known problems such as integer factorization and discrete logarithm problems. These problems have been studied extensively and their hardness has not been contradicted after years of research [2] [3].

Public key cryptography have one-way and trap-door properties. One-way function means easy to compute in one direction but very difficult to compute in the opposite direction. Trap-door function means the inverse function is actually easy if certain trap-door information is known. RSA cryptography uses the given properties to ensure security. The idea behind RSA is that it is relatively easy to multiply numbers but a lot more difficult to factor them. Multiplication is computed in polynomial time whereas factoring time grows exponentially proportional to the size of the number [4]. After the generation of the public and private key, the RSA encryption and decryption is just a modular exponentiation operation. This mathematical operation is represented as $C = M^e \mod n$, where $C$ is cipher text, $M$ is plain text, $e$ is the public key exponent, and $n$ is the modulus.

To use RSA or Diffie-Hellman to protect 128-bit AES keys, one should use 3072 parameters: three times the size in use throughout the Internet today. The equivalent key size for elliptic curve cryptography is just 256 bits. One can see that as symmetric key sizes increase, the required key sizes for RSA or Diffie-Hellman increase at a much faster rate than the required key sizes for elliptic curve cryptosystems. Hence, elliptic curve cryptosystems offer more security per bit increase in key size than either RSA or Diffie-Hellman public key systems [5] [6]. The mathematical operations of ECC is defined over the elliptic curve,

$$y^2 = x^3 + ax + b$$  \(1\)

Where $4a^3 + 27b^2 \neq 0$. Each value of $a$ and $b$ gives a different elliptic curve. All points $(x, y)$ which satisfy the given equation plus a point at infinity lies on the elliptic curve. The public key is a point in the curve and the private key is a random number. The security of ECC depends on the difficulty of Elliptic Curve Discrete Logarithm Problem.

2. Problem Modelling

2.1. Modular Montgomery Multiplier

In 1985, Montgomery introduced a new method for modular multiplication. The approach of Montgomery avoids the time consuming trial division that is a common bottleneck of other algorithms. Montgomery multiplication is defined as follows:

$$\text{Mont}(x, y) = xyR^{-1}\mod N$$  \(2\)

Montgomery’s method for multiplying two integers $x$ and $y$ modulo $N$, avoids division by $N$, which is the most expensive operation in hardware [7]. There is a one-to-one correspondence between each element in $x$ and $y$. This Montgomery representation allows very efficient modular arithmetic especially for multiplication. The input has a restriction $x, y < N$ and the output $T$ is bounded by $T < 2N$. As a consequence, if $T > N$, $N$ must be subtracted so that the output can be used as input of the next multiplication.

The Montgomery multiplier structure has three n-bit data inputs $X$, $Y$ and $N$, one START instruction input, one DONE output, which indicates that the operation has ended, and a l-bit RESULT output. The data path consists of a systolic array, four internal registers, a counter and a comparator. It computes the multiplications according to the Montgomery algorithm [8].

2.2. Modified Montgomery Multiplier Circuit

The systolic array uses one row of cells at a time for computation to get a linear, pipelined modular multiplier. The Montgomery structure is designed using the algorithmic state machine approach. The circuit consists of a controller and a data path. The data path consists of a systolic array, four internal registers that store $x$, $y$, $N$ and $T$ values, a counter and a comparator.

The controller stays in the idle state waiting for the Start instruction. When the Start input is set, $x$, $y$ and $N$ registers are loaded by input values, the T register and the counter are reset. In the first stage of iteration, the outputs
of the systolic array cells are written to the T register and controller goes to the next state. When the controller is in
the second stage of iteration, the counter is incremented by 1. When the counter value reaches \( l - 1 \), the comparator
sets the count end signal. Then the controller goes to the OUT state in which the value of the T register is written
to the Result output and the acknowledgment signal done is set. In the second state, the \( x \) register is shifted one bit
right and the most significant bit (MSB) of the \( x \) register is filled 0. This ensures that, during the last iteration of the
Montgomery algorithm, the value of \( x(0) \) will be 0.

The base of all the iterations in Elliptic Curve cryptosystem is point multiplication. Implantation of Montgomery
multiplication as point to point multiplier enhances the efficiency of the overall system. The speed increases naturally
as there is no trial division in Montgomery algorithm. The Systolic Array in the Montgomery multiplier architecture
is a homogenous network of tightly coupled data processing units. Each node or data processing unit independently
computes a partial result as a function of the data received from its upstream neighbors, stores the result within itself
and passes it downstream.

The equations that perform the Montgomery algorithm are carried out by the systolic array. The data given to the
systolic array is constantly updated at every clock signal till the required number of iterations are carried out. The
equations given in Montgomery algorithm are mainly addition operations. Hence to speed up the computations, an
efficient adder can be used instead of the normal ripple carry adder.

A carry-select adder is divided into sectors, each of which performs two additions in parallel, one assuming a
carry-in of zero, the other a carry-in of one. A four bit carry select adder generally consists of two ripple carry adders
and a multiplexer. At the first stage, three input numbers are added, two of which are the constantly updated values.
The values are appended with an extra bit in the MSB. The carry bit gets appended to the LSB. At the next stage, the
new values and the N value in bit mode is added. The output is determined by the lowest bit of the first stage addition.
If the value is zero then the first stage addition sum and carry is chosen with bits chosen from MSB to 1 leaving out
the last bit. If the value of the lowest bit of the first stage sum is one then the second stage sum and carry are taken
from MSB to 1. Correspondingly the chosen sum and carry is given for the next iterations.

2.3. Implementation of RSA

The significant part of RSA public key Cryptography is the modular exponentiation. The division by N required for
modulo operations is the most expensive part of the architecture. Realizing the Montgomery multiplier architecture in
RSA ensures that the time consuming trial division is avoided. The advantage of Montgomery multiplier structure is
that it speeds up the iteration process by avoiding the division part in modulo operations. Concurrently, the implan-
tation of Montgomery has reduced the area and power of the total structure making its implementation efficient in area,
power and speed. The first step in implementing RSA algorithm is the key generation.
2.3.1. Key Generation

- Select two numbers p and q.
- Compute \( n = p \cdot q \), where n is the number made public.
- Choose a large integer d which is relatively prime, such that \( \gcd(d, (p - 1)(q - 1)) = 1 \).
- The integer e is computed from p,q and d and satisfies the relation, \( ed \equiv 1 \pmod{(p - 1)(q - 1)} \).
- Public key = (n,e) and private key = (d).

After the public and private key is generated, the plain text is enciphered using the square and multiply method for modular exponentiation in the conventional way, with \( C = M^e \mod n \). Entity B encrypts a message M and converts it to cipher text for entity A by \( M = C^d \mod n \). Modular exponentiation operation is simplified into a series of modular multiplication and squaring operation.

2.4. Implementation of ECC

The Elliptic Curve is proven to be more secure due to the difficulty of breaking the discrete logarithm problem. This algorithm is much more secure than the algorithm that runs RSA – modular exponentiation. Hence for the same security, ECC needs only 163 bits of key size while comparing with 1024 bits of key size of RSA. The basic steps of implementing Elliptic Curve Cryptography is Point Multiplication, Point Addition and Point Doubling.

2.4.1. Point Multiplication

In point multiplication a point P on the elliptic curve is multiplied with a scalar k using elliptic curve equation to obtain another point Q on the same elliptic curve, i.e. \( kP = Q \). Point multiplication, along with two other basic elliptic curve operations combine to form the elliptic curve cryptography.

- Point addition, adding two points J and K to obtain another point L i.e., \( L = J + K \).
- Point doubling, adding a point J to itself to obtain another point L i.e. \( L = 2J \).

2.4.2. Point Addition

Point addition is the addition of two points P1 and P2 on an elliptic curve to obtain another point P3 on the same elliptic curve. Consider two points P1 and P2 on an elliptic curve. If \( P_1 \oplus P_2 \), then a line drawn through the points P1 and P2 will intersect the elliptic curve at exactly one more point P3. The reflection of the point P3 gives the point P4, which is the result of addition of points P1 and P2. Thus, on an elliptic curve \( P_4 = P_1 + P_2 \).

2.4.3. Point Doubling

Point doubling is the addition of a point P on the elliptic curve to itself to obtain another point 2P on the same elliptic curve. To double a point P to get 2P, i.e. to find \( L = 2P \), consider a point P on an elliptic curve. If y coordinate of the point P is not zero then the tangent line at P will intersect the elliptic curve at exactly one more point L. The reflection of the point L with respect to x-axis gives the point L, which is the result of doubling the point P. Thus \( L = 2P \).

To improve the computational efficiency of Elliptic Curve cryptosystem, Edward Curve transformations are used. Every elliptic curve over a binary field is bi-rationally equivalent to a curve in Edwards form over an extension of the field, and in many cases over the original field. A new form for elliptic curves was added to the mathematical literature recently [9] [10]. Edwards showed that all elliptic curves over number fields could be transformed to the shape \( x^2 + y^2 = c^2 (1 + x^2 y^2) \), with \((0, c)\) as a neutral element and with the surprisingly simple and symmetric addition law

\[
(x_2y_1, x_2y_2) \rightarrow \left( \frac{x_1y_2 + x_2y_1}{c(1 + x_1x_2y_1y_2)}, \frac{y_1y_2 + x_1x_2}{c(1 - x_1x_2y_1y_2)} \right)
\]

Similarly, all elliptic curves over binary finite fields can be transformed to Edwards form. Explicit formulas (i.e., sequences of additions, subtractions and multiplications) have been provided that
• compute an addition \((X_1 : Y_1 : Z_1), (X_2 : Y_2 : Z_2) \Rightarrow (X_1 : Y_1 : Z_1) + (X_2 : Y_2 : Z_2)\) using 10M + 1S.

• compute a doubling \((X_1 : Y_1 : Z_1) \Rightarrow 2(X_1 : Y_1 : Z_1)\) using 6M + 4S.

The Edward curve formula work for all pairs of input points on the curve with no exceptions for doubling or addition.

The main hindrance in Elliptic Curve Cryptography is its time consuming computations. Point Multiplication, Addition and Doubling consume time during the various iterations. The Elliptic Curve equations start from point multiplication to sequential multiplication. The adders used in sequential multiplication are carry save adder that computes both sum and carry of the same number of bits as data and carry look ahead adder that computes data ahead of the clock signal. The carry-save adder reduces the addition of 3 numbers to the addition of 2 numbers. The propagation delay is three gates regardless of the number of bits. The carry-save unit consists of n full adders, each of which computes a single sum and carry bit based solely on the corresponding bits of the three input numbers. The entire sum can then be computed by shifting the carry sequence left by one place and appending a 0 to the front (most significant bit) of the partial sum sequence and adding this sequence with RCA produces the resulting \(n + 1\)-bit value. This process can be continued indefinitely, adding an input for each stage of full adders, without any intermediate carry propagation.

Carry look-ahead adder is designed to overcome the latency introduced by the rippling effect of the carry bits. The propagation delay occurred in the parallel adders can be eliminated by the carry look ahead adder. This adder is based on the principle of looking at the lower order bits of the augends and addend if a higher order carry is generated. The base operation is point multiplication. The modified Montgomery multiplier algorithm is implanted in the multiplication operations and thereby speed is improved.

3. Simulation Results

Fig. 2. shows the simulation results for a conventional RSA encryption. 256 bits of data is encrypted with 64 bits and 256 bits of the cipher text is obtained. Modular exponentiation method is followed in this process of encryption.

Fig. 2. Cipher text obtained from RSA Conventional

Fig. 3. shows the simulation results for the output of Montgomery Multiplier Algorithm.

Fig. 3. Output of Montgomery Multiplier Structure
The simulated output waveforms for the combined Elliptic Curve Cryptography is shown in Fig. 4.

A detailed comparative study has been done on the conventional and Montgomery multiplier methods and its modified version for RSA Cryptography and Elliptic Curve Cryptography. The Montgomery modified method is more efficient in terms of speed, area and power. First a comparison is done between Montgomery conventional and modified method in terms of area, power and speed as shown in Table 1. Secondly, Table 2 shows the comparison between RSA cryptography in conventional and Montgomery multiplication method.

### Table 1. Power Delay Comparison of Montgomery multiplier and Modified structure

| Multiplier               | Montgomery | Modified Montgomery |
|-------------------------|------------|---------------------|
| Delay (Min per) (ns)    | 6.513      | 4.832               |
| Power Consumption (W)   | 0.070      | 0.069               |
| Area (no. of slices)    | 35         | 41                  |

### Table 2. Power Delay Comparison of RSA Conventional and RSA implanted with Montgomery

| RSA Public Key Cryptography | Conventional | Modified Montgomery |
|-----------------------------|--------------|---------------------|
| Delay (Min per) (ns)        | 8.869        | 6.219               |
| Power Consumption (W)       | 0.078        | 0.077               |
| Area (no. of slices)        | 1289         | 650                 |
| Area (no. of 4 input LUTs)  | 1632         | 1072                |

### Table 3. Power Delay Comparison of Different Point Addition Schemes

| Point Addition | Conventional | Montgomery | Modified Montgomery |
|----------------|--------------|------------|---------------------|
| Delay (Min per) (ns) | 44.144       | 38.761     | 30.341              |
| Power Consumption (W)  | 0.082        | 0.082      | 0.081               |
| Area (no. of slices)    | 5856         | 3480       | 3629                |

### Table 4. Power Delay Comparison of Different Point Doubling Schemes

| Point Double | Conventional | Montgomery | Modified Montgomery |
|--------------|--------------|------------|---------------------|
| Delay (Min per) (ns) | 66.395       | 47.762     | 40.864              |
| Power Consumption (W)  | 0.082        | 0.083      | 0.082               |
| Area (no. of slices)    | 6014         | 3810       | 4087                |

The set of comparisons for Elliptic Curve Point Additions and Point Doubling for the conventional and Montgomery multiplications using conventional and modified structures. Table 3 shows the comparisons for area, delay and power in Point Addition Scheme using the different structures. And finally, the comparisons for area, delay and power in Point Doubling are shown in Table 4.
4. Conclusion

In this paper, an overview of the two main cryptosystems - RSA Cryptography and Elliptic Curve Cryptography is implemented. The main constraint in both the cryptosystems is the delay due to the large number of computations. Hence, Montgomery multiplier architecture is implanted in both the cryptosystems. The modified Montgomery multiplier gives a 25.8% decrease in delay in terms of minimum period, also a 1.4% decrease in terms of power. When Montgomery multiplier algorithm is implanted in RSA public key cryptography, there is a 49.5% decrease in the used area in the form of number of slices, a 29.87% decrease in the delay in the form of minimum period and a 1.28% decrease in power. The Point Addition and Point Doubling schemes of Elliptic Curve Cryptography are implemented using Edward curve transformations, which are computationally more efficient when compared with the more familiar Jacobian transformations. From the base stage, the different adders used in addition and multiplication have been implemented to increase the speed. The implantation of modified Montgomery Multiplier in Point Addition gives 38% decrease in the area in terms of number of slices, 31.2% decrease in the delay in terms of minimum period and a 1.21% decrease in power. The implantation of modified Montgomery Multiplier in Point Doubling gives 32% decrease in the area in terms of number of slices, 38.45% decrease in the delay in terms of minimum period.

References

[1] D. Kahn (1967), “The Codebreakers, The Story of Secret Writing”, New York: Macmillan.
[2] Ivey, J. S. P., Walker S. and Davidson S. (1992), “An ultra-high speed public key encryption processor”, IEEE Custom Integrated Circuits Conference Proceedings pp. 1961-1964.
[3] Diffie W., Hellman M. E., “New directions in Cryptography” IEEE Transactions Information Theory, Vol. 22, no. 6 (1976), pp. 644-654.
[4] Rivest, R. L., Shamir, A., Adleman, L., “A method for obtaining digital signatures and public-key cryptosystems” ACM Communications, Vol. 21, no. 2 1978, pp. 120-126.
[5] Koblitz, N. (1987), “Elliptic curve cryptosystems”, Mathematics of computation, Vol. 48, No. 177, pp. 203-209.
[6] V. S. Miller (1985), “Use of Elliptic Curves in Cryptography”, Advances in Cryptology, Vol. 218, pp. 417-426.
[7] Montgomery, P. L., “Modular multiplication without trial division” Mathematics of computation, Vol. 44, no. 170 1985, pp. 519-521.
[8] Sddka Berka, Lejla Batina1, “Hardware Implementation of a Montgomery Modular Multiplier in a Systolic Array” International Parallel and Distributed Processing Symposium Proceedings, Vol. 21, no. 2  February 2003.
[9] Daniel J. and Tanja Lange, “Faster Addition and Doubling on elliptic curves”, Advances in Cryptology ASIACRYPT 2007 vol. 4833, pp 29-50.
[10] Chiara Peretti, Alessio Leoncini, Paolo Gastaldo and Rodolfo Zunino, “Edwards Curves and Extended Jacobi Quartic-Curves for Efficient Support of Elliptic-Curve Cryptosystems in Embedded Systems” International Journal for Information Security Research (IJISR), Volume 3, Issues 3 and 4, December 2013.