# Maximization of Speed in Elliptic Curve Cryptography Using Fuzzy Modular Arithmetic over a Microcontroller based Environment

*Gopinath Ganapathy, and K. Mani*

*Abstract*—**Elliptic Curve Cryptography (ECC) is a sort of public-key cryptosystem that is an alternative to other public-key algorithms like DSA, ElGamal, and Rabin. It is widely accepted because of the usage of smaller parameters than other public key cryptosystems but with same level of security. The basic building blocks of an ECC over ($F_P$) are computations of addition and scalar point multiplication kP mod m, where P is a elliptic curve point, k is arbitrary integer in the range 1 < k < ord(p), and m is a modulus. Although, several methods have been proposed for computing kP, to speed up the modular arithmetic operation which is a key operation in all the methods is not focused. To perform modular operation, normally trail division is used and the hardware implementation of such trail division is very expensive and it may slow down the process. Thus, to speed up the modular operations, a novel fuzzy modular arithmetic is taken in this paper. In fuzzy modular arithmetic, instead of performing division for modulus operation, repeated subtraction is used. Further, an algorithm running on a general computer has only limited physical security and software implementation of cryptographic algorithms is not secured in all time. To overcome, hardware encryption is thought of. Hardware encryption performs cryptographic operations with high speed and has encapsulated security. Thus, this paper focuses on hardware implementation of ECC with fuzzy modular arithmetic using AT89C51 microcontroller.**

*Index Terms*—**ECC, Fuzzy Modular Arithmetic, Fermat's Theorem, and AT89C51 Microcontroller.**

## I. Introduction

Data security and cryptographic techniques are essential for safety relevant applications. Cryptography keeps the message communication secure so that eavesdroppers cannot decipher the transmitted message. It provides the various security services like confidentiality, integrity, authentication and non-repudiation. There are several cryptographic algorithms available for both symmetric and public key cryptosystem [14]. Slow running cryptographic algorithms cause customer dissatisfaction and inconvenience. On the other hand, fast running encryption algorithms lead high speed. To achieve this, high speed custom hardware devices are needed.

Manuscript received July 27, 2009.
Dr. Gopinath Ganapathy, Professor and Head with the computer Science Department, University of Bharthidasan, Trichy, India – 620 024. (Phone : +91 9842407008; email: gganapathy@gmail.com).
K. Mani is with the computer Science Department of Nehru Memorial College, Puthanapatti, University of Bharthidasan, Trichy, India – 621 007. (Phone : +91 9443598804; email: nitishmanik@yahoo.com).

A cryptographic algorithm running on a general-purpose has only limited physical security on most operating system. Thus, hardware encryption devices are used. These devices provide high security performance and high speeds [17]. Nowadays, mobile devices are used in global communication world. These devices are interacting with other devices to perform a task and they are forming ad-hoc networks. The two key aspects of these devices are security and interoperability in the heterogeneous inter-network environment. But due to scarce resources of mobile devices, exhaustive use of cryptographic is infeasible and therefore ECC is used [12].

ECC was proposed in 1985 by Neal Koblitz and Victor Miller. It is an alternative to the established cryptosystems like RSA, ElGamal, and Rabin. It guarantees all the security services with the shorter keys. The use of shorter length implies less space for key storage, less arithmetic cost and time saving when keys are transmitted [13]. These characteristics make ECC the best choice to provide security in wireless networks. ECC has increased as evidenced by its inclusion in standards by in credited standards organizations such as American National Standards Institute *(ANSI*), Institute of Electrical and Electronic Engineers *(IEEE*), International Standards Organization *(ISO),* and National Institute of Standards and Technology *(NIST*). The security of ECC is based on the discrete logarithm problem over the points on an elliptic curve. But, it is more difficult problem than the prime number problem of RSA algorithm [3]. These two problems are closely related to the key length of cryptosystems. If the security problem is more difficult, then smaller key length can be used with sufficient security. This smaller key length makes ECC suitable for practical applications such as embedded systems and wireless applications [4].

A vast amount of research has been conducted on secure and efficient implementation since then. All focus on scalar point multiplication that is kP, where k is a scalar and P is a point on elliptic curve. Efficient hardware and software implementation of scalar point multiplication is the main research topic on ECC in recent years. To encrypt and decrypt the message using any public key cryptosystem like ECC, modular arithmetic plays a vital role and it is the key operation in modern cryptology. Even though, many algorithms have been developed for faster implementation, none of the work reveals the use of fuzzy modular arithmetic to speed up the encryption and decryption operations on ECC. Thus, this work focuses on hardware implementation of ECC using fuzzy modular arithmetic with AT89C51 microcomputer. The key idea used in fuzzy modular arithmetic is not to compute the result exactly as in the traditional modular arithmetic because the traditional modular arithmetic uses division operation for modulo reduction m. Thus, instead of the full reduction, a pseudo "fuzzy" randomized partial reduction is performed.

The rest of the paper is organized as follows. A short introduction to elliptic curve is provided in section 2. The proposed methodology for implementation of ECC using fuzzy modular arithmetic is discussed in section 3. A brief introduction to fuzzy modular arithmetic is presented in Section 4. Implementation of ECC using fuzzy modular arithmetic in AT89C51 microcomputer is provided in Section 5. Experimental results are discussed in Section 6. Finally, we draw our conclusion in Section 7.

## II. Elliptic curve

In this section, we present the definition of elliptic curve, addition and scalar point multiplication on elliptic curve, and Fermat's Theorem.

*A.1 Definition (Elliptic Curve over GF(p))* : Let $p$ be a prime greater than 3 and let a and b be two integers such that $4a^3 + 27b^2 \neq 0 \pmod{p}$. An elliptic curve E over the finite field F$p$ is the set of points $(x, y) \in Fp \times Fp$ satisfying the Weierstrass equation

$$E:\ y^2 = x^3 + a\,x + b \qquad (1)$$

together with point at infinity O [9].

The inverse of the point $P = (x_1, y_1)$ is $-P = (x_1, -y_1)$. The sum P+Q of the points $P = (x_1,y_1)$ and $Q = P = (x_2, y_2)$ (where $P,Q \neq O$ and $P = \pm Q$) is the point $R = (x_3, y_3)$ where

$$\lambda = \frac{(y_2 - y_1)}{(x_2 - x_1)},\ x_3 = \lambda^2 - x_1 - x_2,\ y_3 = (x_1 - x_3)\,\lambda - y_1. \qquad (2)$$

For P = Q, the doubling formulae are $\lambda = \frac{3x_1^2 + a}{2y_1}$,

$$x_3 = \lambda^2 - 2x_1,\ y_3 = (x_1 - x_3)\,\lambda - y_1. \qquad (3)$$

The point at infinity O plays a rule similar to that of the number 0 in normal addition. Thus, P + Q = P, and P + (– P) = O for all points P. The point on E together with the operation of "addition" forms an abelian group [2], [8]. Since the point or scalar multiplication is the main operation on ECC, the 'kP' multiplication is based on point doubling and point addition and it can be calculated by using double-and-add algorithm, which is shown in Algorithm 1 [11].

```
Algorithm 1 Elliptic Curve point Multiplication : Binary Method

Input : A point P an l-bit integer $k = \sum_{j=0}^{l-1} k_j 2^j, k_j \in \{0,1\}$
Output : Q = [k]P
1. $Q \rightarrow P$
2. for j from $l-1$ to 0 by $-1$ do :
3.    $Q \rightarrow [2]\,Q$
4.    If $k_j = 1$ then
          $Q \leftarrow P + Q$
5.    end if
6. end for
```

*A.2 Modular Multiplication Inversion*

It is done according to Fermat's theorem [16], $a^{-1} = a^{p-2} \bmod p$, if gcd $(a,p) = 1$. In this paper, we are interested the E defined over GF ($p$), p is a prime. Thus, Fermat's theorem is used to find multiplicative inverse modulo p. The multiplication inversion can be performed by modular exponentiation of $a$ by $p-2$ and it can be realized by using the square and multiply algorithm [6], [15].

## III. Methodology

The proposed methodology to speed up the modular arithmetic is shown in Fig. 1.

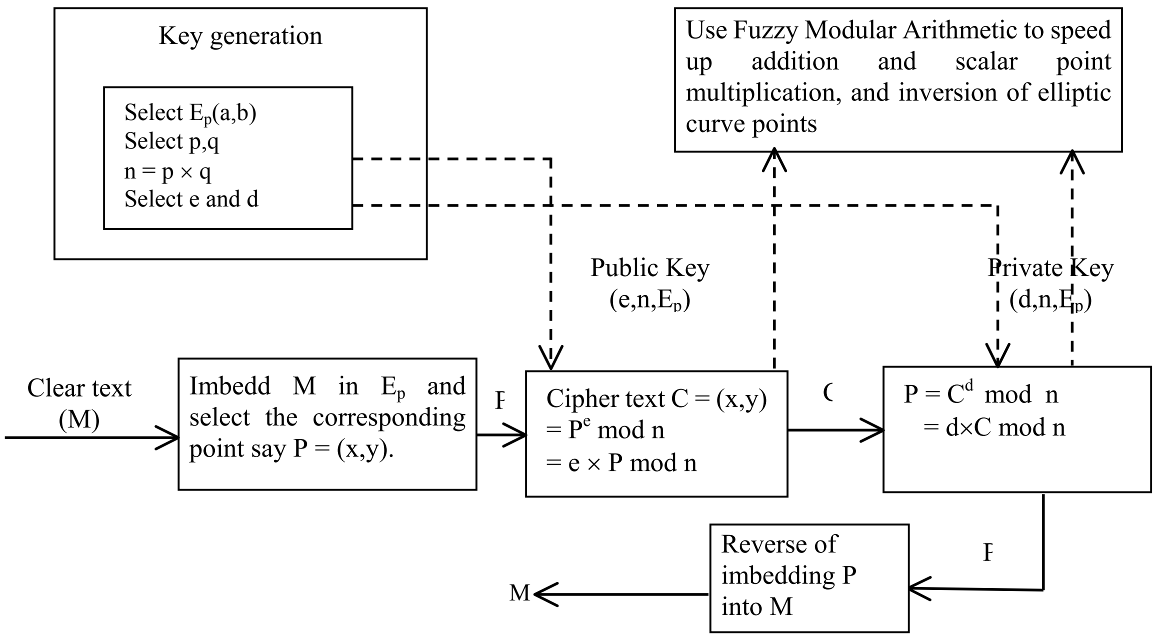

Key generation
Select Ep(a,b)
Select p,q
n = p × q
Select e and d
Use Fuzzy Modular Arithmetic to speed up addition and scalar point multiplication, and inversion of elliptic curve points
Public Key (e,n,Ep)
Private Key (d,n,Ep)
Clear text (M)
Imbedd M in Ep and select the corresponding point say P = (x,y).
P
Cipher text C = (x,y) = P^e mod n = e × P mod n
C
P = C^d mod n = d×C mod n
P
Reverse of imbedding P into M
M

Fig. 1. ECC with Fuzzy Modular Arithmetic Model

## IV. FUZZY MODULAR ARITHMETIC

The concept of fuzzy modular arithmetic for cryptographic schemes was proposed by Wael Ali. Partial reduction modulo *m* is performed in fuzzy modular arithmetic. In partial modulo reduction instead of performing division, the division is replaced by repeated subtraction. For that some random multiples of *m* is used. This approach is called fuzzy modular arithmetic because the operation used is not exact but it involves some random multiples of m from the value to be reduced modulo m. The result of the process is not the rest of the division usually computed, but another value and it can be used for calculation. The numerical example of serial fuzzy modular multiplication is shown in Fig. 2. In Fig. 2(a) the conventional multiplication of two binary integers, say A=35, and B=33 with *m*=13 is shown and the result is 11 (mod 13). Since, fuzzy modular multiplication mainly focuses on repeated subtraction instead of division for modulo reduction *m*, in this work *m* = 2 X 13 = 26 is considered and the process is shown in Fig. 2(b). It is noted that subtraction is performed whenever the bit position in B is zero. The hardware architecture for fuzzy modular multiplier is shown in Fig. 3 [1], [5].

```
A = (35)₂ =        100011                                   35 ×
B = (33)₂ =        100001                                   33
                100011                                    1155 = 11 (mod 13)
                   000000
                  000000
                 000000
                000000
               100011
               10010000011
```

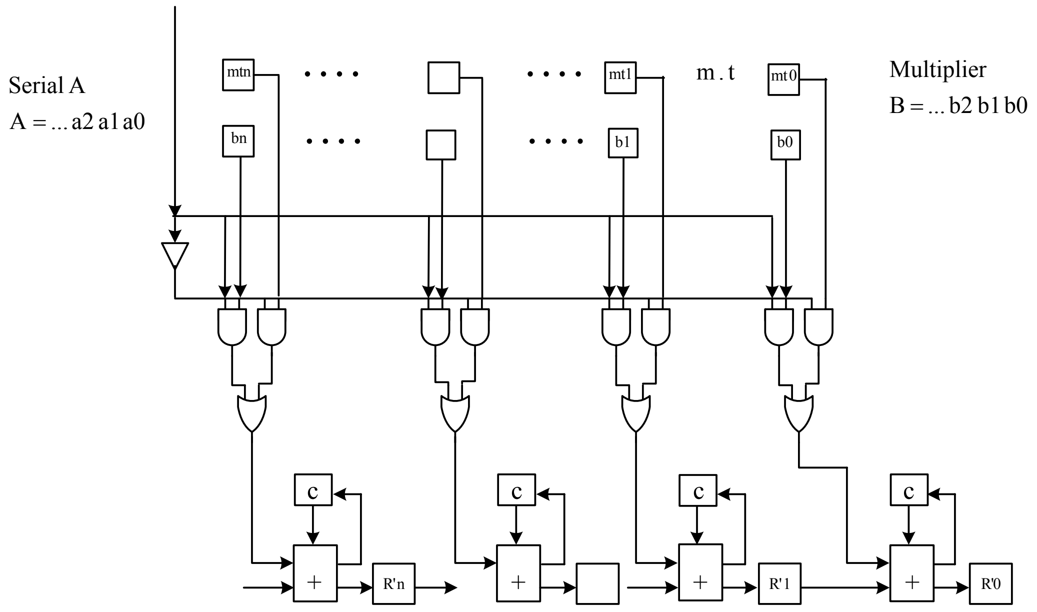
Fig. (a). Conventional Multiplication

The modulus m = 13, a multiple of m i.e., m = m·t = 13·2 = 26 is subtracted each time. The 2’s complement of m is …1100110.

```
  A = (35)₂ =          100011    |     35 ×
   B = (33)₂ =         100001    |     33
              0000000100011      |    + 35
        1111111111100110         |     – 2 × 26
        111111111100110          |   – 4 × 26
       1111111111100110          |   – 8 × 26
       111111111100110           |   16 × 26
      000000000100011            |   32 × 35
….000000000101110111             |     375 = 11(mod 13)
```

Fig. (b) Fuzzy Modular Multiplication

Fig. 2. Numerical Example of Modular Multiplication with and without Fuzzy



Fuzzy Modular Multiplication
R = A · B (mod m)
R′ = A · B – z m

Fig. 3. Simplified Hardware Architecture for Fuzzy Modular Multiplication

## V. IMPLEMENTATION OF ECC USING FUZZY MODULAR ARITHMETIC

Algorithm 2 EC point addition and doubling using Fuzzy Modular Arithmetic

```
Input : p1= (x1,y1), p2 = (x2,y2) and m
Output : p1 + p2 = p3 = (x3,y3)
1. $T_{\lambda_1} \leftarrow y_2 - y_1$
2. $T_{\lambda_2} \leftarrow x_2 - x_1$
3. $T_{\lambda_{m2}} \leftarrow T_{\lambda_2}^{m-2}$ mod m (Fermat's Theorem)
4. $T^{-1}_{\lambda_{m2}} \leftarrow T_{\lambda_{m2}} - z_1 m$ ( $z_1 m \le T_{\lambda_{m2}}$ )
5. $\lambda \leftarrow T_{\lambda_1} T^{-1}_{\lambda_{m2}}$
6. $T_{x_3} \leftarrow \lambda^2 - x_1 - x_2$
7. If $T_{x_3} > m$ then
8.     $x_3 \leftarrow T_{x_3} - z_2 m$ ( $z_2 m \le T_{x_3}$ )
   else
9.     $x_3 \leftarrow T_{x_3}$
10. end if
11. $T_{y_3} \leftarrow (x_1 - x_3)\lambda - y_1$
12. If $T_{y_3} > m$ then
13.     $y_3 \leftarrow T_{y_3} - z_3 m$
    else
14.  $y_3 \leftarrow T_{y_3}$
15. end if
```

```
Input : p1= (x1,y1), a and m
Output : 2 p1 = p3 = (x3,y3)
1. $T_{\lambda_1} \leftarrow 3x_1^2 + a$
2. $T_{\lambda_2} \leftarrow 2y_1$
3. $T_{\lambda_{m2}} \leftarrow T_{\lambda_2}^{m-2}$ mod m (Fermat's Theorem)
4. $T^{-1}_{\lambda_{m2}} \leftarrow T_{\lambda_{m2}} - z_1 m$ ( $z_1 m \le T_{\lambda_{m2}}$ )
5. $\lambda \leftarrow T_{\lambda_1} T^{-1}_{\lambda_{m2}}$
6. $T_{x_3} \leftarrow \lambda^2 - 2x_1$
7. If $T_{x_3} > m$ then
8.     $x_3 \leftarrow T_{x_3} - z_2 m$ ( $z_2 m \le T_{\lambda_{m2}}$ )
   else
9.   $x_3 \leftarrow T_{x_3}$
10. end if
11. $T_{y_3} \leftarrow (x_1 - x_3)\lambda - y_1$
12. If $T_{y_3} > m$ then
13.     $y_3 \leftarrow T_{y_3} - z_3 m$
    else
14.  $y_3 \leftarrow T_{y_3}$
15. end if
```

The execution of ECC scheme is mostly dominated by the efficient implementation of finite field arithmetic and point multiplication kP. To compute kP, several methods such as binary method, binary NAF method, and windowed NAF. have been proposed [10]. To implement these methods, efficient hardwired architecture and a platform that support the efficient computation of such methods are needed. Previous hardware work includes: the first ASIC implementation with Motorola M68008 microcomputer, reconfigurable finite-field multiplier, parallelized field multiplier [7], [18]. The main objective of this work is to perform modular operations without trail division because the hardware implementation of trial division is an expensive. Thus, this work emphasizes fuzzy modular arithmetic with no division and division is repeatedly replaced by subtraction and hence the hardware implementation is relatively easy. Moreover, to perform kP, repeated addition is used. To add two points of EC, in both formula modular multiplication inversions is involved and it takes considerable amount of time. To avoid it, Fermat's Theorem is used in this paper which is illustrated in section A.2 .The EC point addition and doubling using Fuzzy Modular Arithmetic and Fermat's theorem is given in Algorithm 2. Moreover, this work further focuses on the implementation of ECC with AT89C51 microcontroller in which the simplified fuzzy modular multiplier is embedded. AT89C51 is a low-power; high performance CMOS 8-bit microcomputer with 4K bytes of Flash programmable and erasable read only memory. The on-chip Flash allows the program memory to be reprogrammed in-system.

## VI. EXPERIMENTAL RESULT

This section summarizes the performance of the hardware implementation of ECC with and without fuzzy modular arithmetic. The software implementation of the work is performed in Visual C++ version 6.0 IDE, Advanced C version 4.0. The time taken for encryption and decryption operations of ECC for the various file sized message with and without fuzzy modular arithmetic is shown in Table I.

Table I Encryption and Decryption time of ECC with and without Fuzzy Modular Arithmetic

| File size (KB) | ECC without Fuzzy modular Arithmetic (Time in ms) | | | ECC with Fuzzy modular Arithmetic (Time in ms) | | |
|---|---|---|---|---|---|---|
| | E | D | T | E | D | T |
| 1 | 1490 | 1370 | 2860 | 1320 | 1340 | 2660 |
| 3 | 2470 | 2410 | 4880 | 1920 | 1980 | 3900 |
| 5 | 3510 | 3460 | 6970 | 2820 | 2890 | 5710 |
| 7 | 4573 | 4512 | 9085 | 3916 | 3996 | 7912 |
| 8 | 5685 | 5642 | 11327 | 4613 | 4663 | 9276 |

KB- Kilo Bytes, ms- mille seconds, E- Encryption Time, D-Decryption Time, and T- Total Time

From Table I, it is observed that the time taken for encryption and decryption with fuzzy modular arithmetic is substantially decreased compared to corresponding counterpart without fuzzy modular arithmetic. Further, it is observed that encryption time is moderately larger than decryption time in both cases.

## VII. CONCLUSION

Microcontroller based implementation of ECC is thought of and implemented. To increase the speed of encryption and decryption operations fuzzy modular arithmetic and Fermat's theorem are used. In fuzzy modular arithmetic to perform modulus operation, repeated subtraction is used instead of division and hence hardware complexity is relatively reduced than division operation. To perform inversion operation with respect to modulus, Fermat's theorem is used which is further simplified by repeated multiplication.

## REFERENCES

[1] W. Adi, "*Fuzzy Modular Arithmetic for Cryptographic Schemes with Applications for Mobile Security*", IEEE, 2000, pp. 263-265.
[2] S. B. Ors, L. Batina, B. Preneael, and J. Vandewalle, "*Hardware Implementation of an Ellipic Curve Processor over GF (p)*", in Proceedings of the IEEE Conference on Application Specific Systems users and Processors (ASSP), 2000, pp. 433-443.
[3] R. Cheung, W. Telle, and P. Cheung, "*Customizable Elliptic Curve Cryptosystems*", IEEE, TVLSI, Vol. 13(a), September, 2005 , pp.1048-1049.
[4] D.Hankerson, L. Herandez, and A. Menezes, "*Software Implementation of Elliptic Curve Cryptography over Binary Fields*", CHES, LNCS 1965, Berlin Heidelberg, Springer-Verlag, 2000, pp. 1-24.
[5] A. Hanoun, W. Adi , F. Mayer-Lindenberg, and B.Soundan , "*Fuzzy Modular Multiplication Architecture and Low Complexity IPR-Protection for FPGA Technology*", IEEE, 2006, pp. 325-326.
[6] D. Knuth, "*The Art of Computer Programming – Semi Numerical Algorithms*", Vol.2, Addison-Wesley, Third Edition, 1998.
[7] J. Lutz, and A. Hasan, "*High Performance FPGA based Elliptic Curve Cryptographic Co-processor*", In ITCC 04, International Conference on Information Technology Coding and Computing, Vol. 2, 2004, pp. 486.
[8] J. Julio Lopez, and R. Dahab, "*An Overview of Elliptic Curve Cryptography*", Reconfigurable Architecture Workshop (RAW), Nice, France, April 22, 2003, pp. 1-28.
[9] N. Koblitz, "*A Course in Number Theory and Cryptography*", Second Edition, Springer-Verlag, 1994.
[10] N. Koblitz, "*Elliptic Curve Cryptosystems*", Mathematics of Computation Vol. 48 (177), November, 1982, pp. 203-209.
[11] A. Menezes, and S.Vanstone, "*Elliptic Curve Cryptosystems and their implementation*", Journal of Cryptology 6, 1993, pp. 209-224.
[12] G. Orlando, and C. Paar, "*A High Performance Reconfigurable Elliptic Curve Processor for GF($2^m$)*", CHES, Lectures Notes in Computer Science 1965, Springer-Verlag, 2000, pp. 41-56.
[13] M. Morales, Sandoval, and Claudia Feregrino-Uribe, "*GF($2^m$) Arithmetic Modulus for Elliptic Curve Cryptography*", IEEE, 2006.
[14] A. Menezes, P. Van Oorschot, and S. Vanstone, "*Handbook of Applied Cryptography*", CRC Press, 1997.
[15] I. Blake, G. Seroussi, and N.P. Smart, "*Elliptic Curves in Cryptography*", Ser. London Math. Soc. Lecture Note Series, Cambridge Univ. Press, 1999.
[16] S. B. Ors, L. B. Batina , and J. Vandewalle, "*Hardware Implementation of a Montegomery Modular Multiplier in a systolic array*", In the 10th Reconfigurable Architecture Workshop (RAW), Nice, France, 2003.
[17] T. Wollinger, J. Guajardo, and C. Christof Paar, "*Cryptography in Embedded Systems: An Overview*", Proceedings of the Embedded World 2003 Exhibition and Conference, Design Electronik, Nuernberg, Germany, February 18-20, 2003, pp. 735-772.
[18] Yong-Je Choi, Moo-Seop Kim, Hang-Rok Lee, and Ho-Wan Kim , "*Implementation and analysis of Elliptic Curve Cryptosystems over Polynomial basis and ONB*", PWAEST, Vol. 10, December, 2005, pp. 130-134.