Proceedings of Engineering & Technology (PET)
pp.259-263
Copyright IPCO-2016

Software Implementation of ECC Using GMP Library

Anissa Sghaier[#1], Chiraz Massoud[#2], Medien Zeghid[*#3], Mohsen Machhout[#4]

[#]*Faculty of Sciences, EμELab , University of Monastir ,*
Monastir 5019, Tunisia

[1]sghaieranissa@yahoo.com

[2]massoud.chiraz@hotmail.fr

[4]machhout @yahoo.com

[*]*Higher Institute of Applied Sciences and Technology,*
Taffala city 4003 Sousse

[3]medien.zeghid@fsm.rnu.tn

***Abstract*—This paper presents an efficient software implementation of elliptic curve scalar multiplication. Computing kP need finite field point operations which are point addition and point doubling. Due to the use of big number, GMP library is used to support large integers. We implemented our design in INTEL (R) Core TM i3 CPU M380 @2.53 GHz 2.58 GHz with RAM 3 Go processor.**

***Keywords*— Elliptic Curves Cryptosystems (ECC), scalar multiplication, Discrete Logarithm (DL), GMP Library.**

I. Introduction

Nowadays, data security became one of the primordial requirements for many applications in our life. Various methods exist to protect data; they are classified into two categories, symmetric cryptography and asymmetric cryptography. Asymmetric cryptography, called also public-key cryptography, gained a big interest. It doesn't provide algorithms for data encryption and decryption, but also for digital signatures and authentication.

The elliptic curve cryptography (ECC) achieves an equivalent level of security with smaller key sizes compared with traditional asymmetric techniques such as RSA. ECC is based on scalar multiplication; it represents the most complex operation. It is based on two operations which are Point Addition and Point Doubling. These point operations are computed using arithmetic operations, such as multiplication, squaring and inversion.

There are two type of implementation, Hardware and Software. In this paper, we are interested in the Software implementation of ECC scalar multiplication. We used a specific library called Multiple Precision Arithmetic Library (GMP), it has a rich set of functions, specially for the use of cryptography. The aim of GMP library is to be faster than any other library for all operand sizes, big numbers and small numbers. In our work, we used GMP for integer arithmetic to compute scalar multiplication over GF (p).

The remainder of this paper is structured as follows. Section 2 gives briefly a mathematical background of elliptic curve cryptography. Section 3 presents our software implementation. We conclude the paper in Section 4.

II. ECC and Cryptography

In this section, we will introduce briefly elliptic curves background. Then, we will present ECC scalar multiplication and how to optimize using different coordinates system. And finally, we will give an example of Diffie-Hellman key exchange.

A. ECC Background

Elliptic Curves can be represented over prime field GF (p) or over finite field GF (2^m). In this paragraph, we will give the mathematical formulae for ECC over GF (p) and GF (2^m).

1) ECC over GF (p)

An elliptic curve E over GF (p) can be defined by as:
$y^2 = x^3 + ax + b$, where a, b in GF (p) and $4a^3 + 27b^2 \neq 0$ in GF (p). The set of Fq-rational points of E is defined as
$E(Fq) = \{(x, y) \in Fq \times Fq \,|(x, y) \text{ satisfy } E\} \cup \{O\}$
The algebraic formula for point addition and point doubling are given as follows:
$x_3 = m^2 - x_1 - x_2 \pmod p$
$y3 = m(x_1 - x_3) - y_1 \pmod p$
With:
$m = (y_2 - y_1) * (x_2 - x_1)^{-1}$, if $P \neq Q$
$m = (3x_1^2 + a) * (2y_1)$, P=Q
Where, the addition, subtraction, multiplication and division are arithmetic operations over GF (p)

2) ECC over GF(2^m)

Let E be a non-super singular curve defined over a binary field, GF (2^m):
E: $y^2+xy=x^3+ax^2+b$ where a, b in GF (2^m) and $b\neq 0$.
Let $A=\sum_0^{m-1} a_i . x^i$ and $B=\sum_0^{m-1} b_i . x^i$, a_i, b_i in {0, 1}
Then $A + B = \sum_0^{m-1}(a_i + b_i). x^i$ and A × B= A.B mod f
Let $P_1=(x_1, y_1)$, $P_2=(x_2, y_2)$

$P_1 + P_2 = P_3 = (x_3, y3)$ where
$x_3 = m^2 - x_1 - x_2 \pmod p$
$y_3 = m(x_1 - x_3) - y_1 \pmod p$

And $m = (y_2-y_1)*(x_2-x_1)^{-1} \bmod p$, if $P_1\neq P_2$
$m = (3x_1^2+a)*(2y_1)^{-1} \bmod p$, if $P_1 = P_2$

Special cases: If m is infinite, $P_3 = \infty$, and $\infty + P = P$ for all point.

From the result of 2P and 3P, we can calculate other scalar multiplications on the elliptic curve.

B. *ECC Scalar multiplication*

The scalar multiplication is the most complex operation in elliptic curve cryptography. It is based on the calculations of the form Q = k.P = P + P + P… k times. It consists of a succession of point operations.

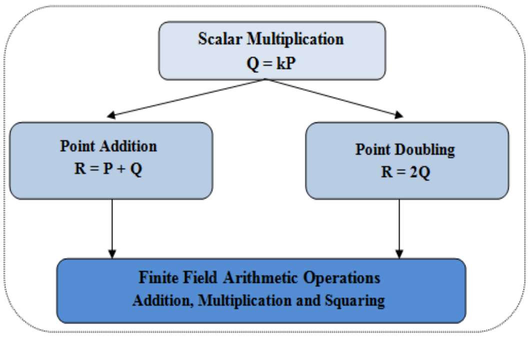

Scalar Multiplication
Q = kP
Point Addition
R = P + Q
Point Doubling
R = 2Q
Finite Field Arithmetic Operations
Addition, Multiplication and Squaring

Fig. 1 Scalar Multiplication Hierarchy

Fig. 1 present the scalar multiplication hierarchy based on Point Addition and Point Doubling, for example, if we take k=5 and P a point of the curve E, k.P= 5P = 2(2P)+P, here we need 2 doubling operations and one addition operation to compute scalar multiplication.

```
Algorithm 1: Scalar multiplication MSB First
Inputs: k= ($k_{n-1}$… $k_1$, $k_0$), with n = [$\log_2$ l]
    Output: Q = k.$P_1$

1: Z ← $P_1$
2: For i from (n-2) down to 0 do
3: Z ← 2.Z
4: If $k_i$ = 1 then
5: Z ← Z+ $P_1$
6: End if
7: End for
8: Return (Z)
```

One of the most used algorithms to perform scalar multiplication is MSB First which is a sequential algorithm. It requires m point doublings and (m-1)/2 point additions on the average. Let's take an example of 6P, 6= $(110)_2$, 6P = 2(2(P) + P).

C. *Coordinates System:*

The computation of scalar multiplication required the calculation of inversions which is very costly. The affine coordinate uses the inversions to calculate this operation or affine coordinate's inversions are very expensive. So to solve this problem we can use the projective coordinates in Lopez Dehab, such as (x.y.z), z≠0, maps to (x/z, y/z²). These coordinates systems replace inversion by the multiplication operations and then perform one inversion at the end (to obtain back the affine coordinate).

D. *Diffie-Hellman*

If we have two distinct points, P and Q on an elliptic curve E, to add the points P and Q, a line is drawn through the two points. This line will intersect the elliptic curve in exactly one more point, call -R. The point -R is reflected in the x-axis to the point R=P + Q. Any elliptic curves have an additive identity, we note O this point called also point at infinity.

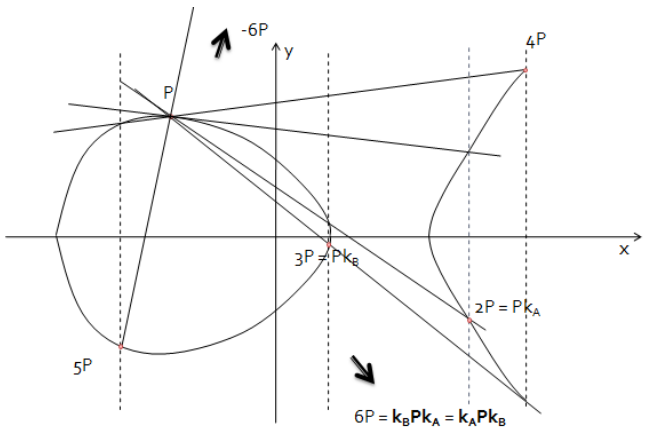


Fig. 2 Diffie-Hellman key exchange

To add a point P to itself, called doubling operation, a tangent line to the curve is drawn at the point P. If P is not 0, then the tangent line intersects the elliptic curve at exactly one other point, -R which is reflected in the x-axis to R.

Let's now take an example of Diffie-Hellman exchange, like it's mentioned in Fig. 2. Alice and Bob define an elliptic curve E, with parameters a, b and a random value k. They choose a point P, and every one select arbitrary, his private key, here k_A=2 and k_B=3. Alice and Bob compute respectively; Pk_A=2P and Pk_B=3P. They exchange keys and every one calculates k_APk_B=2×3P=6P and k_BPk_A=3×2P=6P. They found finally the same key.

III. SOFTWARE DESIGN

In this section, we will present our software implementation using GMP Library.

A. *GMP Library used with ECC*

The GNU Multiple Precision Arithmetic Library (GMP) is a free library for arbitrary-precision arithmetic, operating on signed integers, rational numbers, and floating-point numbers. There are no practical limits to the precision except the ones implied by the available memory in the machine GMP runs on (operand dimension limit is 2^{32}-1 bits on 32-bit machines and 2^{37} bits on 64-bit machines). GMP has a rich set of functions, and the functions have a regular interface. The basic interface is for C.

The main target applications of GMP are cryptography applications and research, Internet security applications, and computer algebra systems.

B. *Implementation design*

In this part, we will present how to define an elliptic curve, then how to find elliptic points and finally, we will give an example of encryption and decryption message.

1) *ECC Parameters*

Every user has a public, used for encryption/signature verification, and a private key, used for decryption/signature generation.

- Key generation: Alice and Bob agree on public :
  - Curve parameters (a, b)
  - The modulus p
  - Base point B (on the curve)
  - Pick a random integer *k* as private key
- Encryption

Let's k be a randomly chosen integer as the private key.
Alice and Bob create public/private keys respectively: Alice(n, $P_a = n * B$) and Bob (m, $P_b = m * B$)
Alice sends to Bob a pair of points: {k*B, k*Pa }
Bob represents data to send as a point *Pb*
Bob sends to Alice a pair of points: {k*B, k*Pb }
Alice takes plaintext message, M, and encodes it onto a point, P_M, from the elliptic group

- Decryption

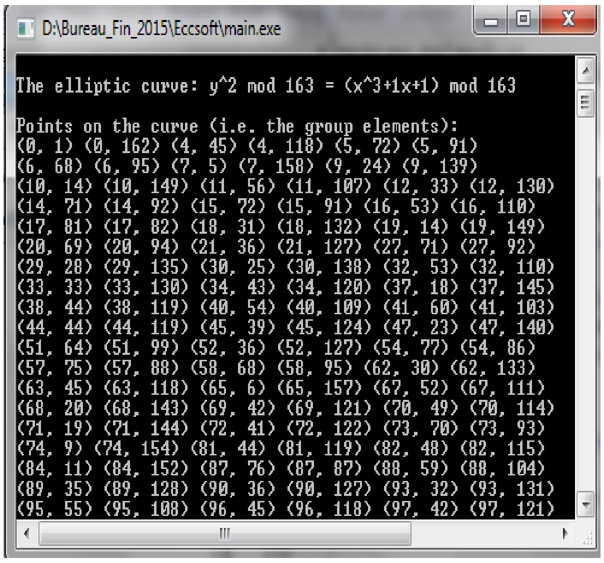
Bob decrypts the message using his private key:

$$Pa + k*B - n (k*B) = Pa + k (n*B) - n (k*B) = Pa$$

Remark: Elliptic curve discrete logarithm problem:
Given P and B, (and P= *n**B), find *n*? This is a difficult mathematical problem which let ECC very hard to be attacked.

2) *Elliptic Curve on a finite set of Integers*

The central part of any cryptosystem involving elliptic curves is the elliptic group. Using the finite fields, we can form an Elliptic Curve Group. Elliptic curve groups over F_p have a finite number of points, which is a desirable property for cryptographic purposes.



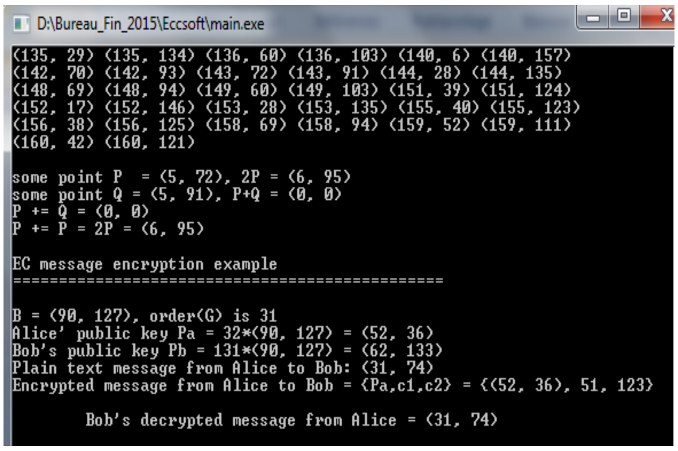
```
D:\Bureau_Fin_2015\Eccsoft\main.exe

The elliptic curve: y^2 mod 163 = (x^3+1x+1) mod 163

Points on the curve (i.e. the group elements):
(0, 1) (0, 162) (4, 45) (4, 118) (5, 72) (5, 91)
(6, 68) (6, 95) (7, 5) (7, 158) (9, 24) (9, 139)
(10, 14) (10, 149) (11, 56) (11, 107) (12, 33) (12, 130)
(14, 71) (14, 92) (15, 72) (15, 91) (16, 53) (16, 110)
(17, 81) (17, 82) (18, 31) (18, 132) (19, 14) (19, 149)
(20, 69) (20, 94) (21, 36) (21, 127) (27, 71) (27, 92)
(29, 28) (29, 135) (30, 25) (30, 138) (32, 53) (32, 110)
(33, 33) (33, 130) (34, 43) (34, 120) (37, 18) (37, 145)
(38, 44) (38, 119) (40, 54) (40, 109) (41, 60) (41, 103)
(44, 44) (44, 119) (45, 39) (45, 124) (47, 23) (47, 140)
(51, 64) (51, 99) (52, 36) (52, 127) (54, 77) (54, 86)
(57, 75) (57, 88) (58, 68) (58, 95) (62, 30) (62, 133)
(63, 45) (63, 118) (65, 6) (65, 157) (67, 52) (67, 111)
(68, 20) (68, 143) (69, 42) (69, 121) (70, 49) (70, 114)
(71, 19) (71, 144) (72, 41) (72, 122) (73, 70) (73, 93)
(74, 9) (74, 154) (81, 44) (81, 119) (82, 48) (82, 115)
(84, 11) (84, 152) (87, 76) (87, 87) (88, 59) (88, 104)
(89, 35) (89, 128) (90, 36) (90, 127) (93, 32) (93, 131)
(95, 55) (95, 108) (96, 45) (96, 118) (97, 42) (97, 121)
```

Fig. 2 Points of ECC

As an example, consider an elliptic curve over the field F_{163}. With a = 1 and b = 1, the elliptic curve equation is:
E: $y^2 = x^3 + x + 1 \bmod 163$
The field F_p uses the numbers from 0 to p - 1, and computations end by taking the remainder on division by p.

$$x = 0 \Rightarrow y^2 = 1 \Rightarrow y=1, 162 \pmod{163}$$
$$x = 1 \Rightarrow y^2 = 3 \Rightarrow \text{no solution} \pmod{163}$$
$$x = 2 \Rightarrow y^2 = 11 \Rightarrow \text{no solution} \pmod{163}$$
$$x = 3 \Rightarrow y^2 = 31 \Rightarrow \text{no solution} \pmod{163}$$
$$x = 4 \Rightarrow y^2 = 69 \Rightarrow y = 45, 118 \pmod{163}$$
$$x = 5 \Rightarrow y^2 = 131 \Rightarrow y = 72, 91 \pmod{163}$$

Then points on the elliptic curve are (0, 1) (0,162) (4, 45) (4,118) (5, 72) (5, 91) and the other points are given in Fig. 2.
As it mentioned in Fig. 2, P and Q are opposite so P + Q = O with O is a point at infinity.



```
D:\Bureau_Fin_2015\Eccsoft\main.exe
(135, 29) (135, 134) (136, 60) (136, 103) (140, 6) (140, 157)
(142, 70) (142, 93) (143, 72) (143, 91) (144, 28) (144, 135)
(148, 69) (148, 94) (149, 60) (149, 103) (151, 39) (151, 124)
(152, 17) (152, 146) (153, 28) (153, 135) (155, 40) (155, 123)
(156, 38) (156, 125) (158, 69) (158, 94) (159, 52) (159, 111)
(160, 42) (160, 121)

some point P  = (5, 72), 2P = (6, 95)
some point Q = (5, 91), P+Q = (0, 0)
P += Q = (0, 0)
P += P = 2P = (6, 95)

EC message encryption example
==============================================

B = (90, 127), order(G) is 31
Alice' public key Pa = 32*(90, 127) = (52, 36)
Bob's public key Pb = 131*(90, 127) = (62, 133)
Plain text message from Alice to Bob: (31, 74)
Encrypted message from Alice to Bob = {Pa,c1,c2} = {(52, 36), 51, 123}

        Bob's decrypted message from Alice = (31, 74)
```

Fig. 3 Encryption and Decryption Results

Fig. 2 shows an example of encryption and decryption using ECC. Alice and Bob choose a base point B (90, 127). They fixed their private keys respectively: a=32 and b=131. Alice and Bob calculated respectively: Pa=32B and Pb=131B. Alice chose a plain text (31, 74) and he encoded it onto (Pa, c1, c2). Then receiving the message from Alice, Bob decrypts it as mentioned in the previous section.

IV. Conclusions

Elliptic curve cryptography (ECC) is now very used in practice because it offers the same level of security with smaller key size comparing to other public key cryptography. In this paper, we presented a software implementation of scalar multiplication over GF (p) field using GMP library which is rich in cryptographic functions applied for all operand sizes (big or small).

References

[1] Praful Kumar Singh, and Mrityunjay Kumar Choudhary, "Scalar Multiplication Algorithms of Elliptic Curve Cryptography over GF (2^m)", International Journal of Innovative Technology and Exploring Engineering (IJITEE), ISSN: 2278-3075, Volume-3, Issue-1, June 2013

[2] Manuel Bluhm, and Shay Gueron, “Fast Software Implementation of Binary Elliptic Curve Cryptography”,, 2013

[3] Jean-Marc ROBERT, “Approches Parallèles de Multiplication Scalaire sur Courbe Elliptique Binaire”, 2014.

[4] **J**ean-Marc Robert, “Software Implementation of Parallelized ECSM over Binary and Prime Fields”, 16 Dec 2014.

[5] Software and Hardware Implementation of Elliptic Curve Cryptography