ELLIPTIC CURVE CRYPTOGRAPHY AND ITS APPLICATIONS

Moncef Amara [1] *and Amar Siad* [2]

Université Paris-8, laboratoire LAGA, Saint-Denis / France.
[1] amara_moncef@yahoo.fr
[2] siad@math.univ-paris13.fr

ABSTRACT

The idea of Elliptic Curve Cryptography (ECC), and how it's a better promise for a faster and more secure method of encryption in comparison to the current standards in the Public-Key Cryptographic algorithms of RSA is discussed in this paper. The Elliptic Curve Cryptography covers all relevant asymmetric cryptographic primitives like digital signatures and key agreement algorithms. The function used for this purpose is the scalar multiplication $k.P$ which is the core operation of ECCs. Where k is an integer and P is a point on an elliptic curve. This article explains the role of ECC in the network security. ECC's uses with smaller keys to provide high security and high speed.

1. INTRODUCTION

ECC is a kind of public-key cryptosystem like RSA. But it differs from RSA in its quicker evolving capacity and by providing attractive and alternative way to researchers of cryptographic algorithm. The security level which is given by RSA, can be provided even by smaller keys of ECC. For example, the 1024 bit security strength of a RSA could be offered by 163 bit security strength of ECC. Other than this, ECC is particularly well suited for wireless communications, like mobile phones and smart cards. EC point of multiplication operation is found to be computationally more efficient than RSA exponentiation.

Elliptic Curve Cryptography is a relatively new cryptosystem, suggested independently, from the second half of 19th century, by Neals Koblitz [1] and Victor Miller [2]. At present, ECC has been commercially accepted, and has also been adopted by many standardizing bodies such as ANSI, IEEE [3], ISO and NIST [4]. Since then, it has been the focus of a lot of attention and gained great popularity due to the same level of security they provide with much smaller key sizes than conventional public-key crypto-systems have.

The ECC covers all relevant asymmetric cryptographic primitives like digital signature (ECDSA), key exchange and agreement protocols. Point multiplication serves as the basic building block in all ECC primitives and is the computationally most expensive operation.

[1] Thanks to LAGA laboratory - University of Paris 8 and Paris 13.

The aim of this work is to explain the role of ECC in the network security, to discuss the comparison with the current standards in the Public-Key Cryptographic algorithms like RSA and to present EC point multiplication processor, intended to the conception of the cryptographic applications, like digital signature (ECDSA) and key agreement (Diffie-Hellman) protocol.

The paper is organized as follows. After a brief introduction, an overview of elliptic curve in cryptography is given in section 2. We present in Section 3, mathematical background on elliptic curve over finite-field $GF(2^m)$. The point multiplication method is explained in Section 4, and Elliptic Curve applications are presented in Section 5. The EC Point multiplication processor is given in Section 6. In Section 7, performance of elliptic curves is presented. Finally, conclusion is summarized in Section 8.

2. ELLIPTIC CURVE CRYPTOGRAPHY

Elliptic curves defined over a finite-field (Figure.1) provide a group structure that is used to implement the cryptographic sch-emes. The elements of the group are the rational points on the elliptic curve, together with a special point O (called the "point at infinity").

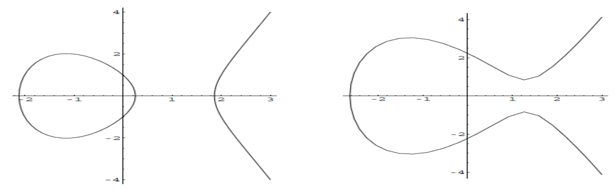

Fig. 1. Graphs of elliptic curves $y^2 = x^3 - 4x + 1$ (on the left) and $y^2 = x^3 - 5x + 5$ (on the right) over $\mathbb{R}$ [5].

A major building block of all elliptic curve cryptosystems is the scalar point multiplication, an operation of the form $k.P$ where k is a positive integer and P is a point on the elliptic curve. Computing $k.P$ means adding the point P exactly $k - 1$ times to itself, which results in another point Q on the elliptic curve. The inverse operation, i.e., to recover k when the points P and $Q = k.P$ are given, is known as the *Elliptic Curve Discrete Logarithm Problem* (ECDLP). To date, no subexponential-time algorithm is known to solve the ECDLP in a properly selected elliptic curve group. This makes Elliptic Curve Cryptography a promising branch of public-key cryptography which offers similar security to other ”traditional” DLP-based schemes in use today, with smaller key sizes and memory requirements, e.g., 160 bits instead of 1024 bits (as shown in Table.1).

Table 1. Key length for public-key and symmetric-key cryptography [6].

Symmetric-key	ECC	RSA/DLP
64 bit	128 bit	700 bit
80 bit	160 bit	1024 bit
128 bit	256 bit	2048-3072 bit

3. BACKGROUND MATHEMATICS

3.1. Groups and fields

A group $(G, +)$ consists of a set of numbers G together with an operation $+$ that satisfies the following properties.

1. Associativity:$(a+b)+c = a+(b+c)$ for all $a, b, c \in G$.
2. Identity: there is an element $0 \in G$ such that $a+0 = 0 + a$ for all $a \in G$.
3. Inverse: for every $a \in G$, there exists an element $-a \in G$ such that $(-a) + a = a + (-a) = 0 \in G$.

A field $(\mathbb{F}, +, \times)$ is a set of numbers F together with two operations and that satisfies the following properties.

1. $(\mathbb{F}, +)$ is an abelian group with identity 0.
2. $(\times)$ is associative.
3. there exists an identity $1 \in F$ with $1 \neq 0$ such that $1 \times a = a \times 1 = a$ for all $a \in F$.
4. the operation $\times$ is distributive over $+$, i.e., $a \times (b + c) = (a \times b) + (a \times c)$ and $(b + c) \times a = (b \times a) + (c \times a)$ for all $a, b, c \in F$.
5. $a \times b = b \times a$ for all $a, b \in F$.
6. for every $a \neq 0, a \in F$ there exists an element $a^{-1} \in F$ such that $a^{-1} \times a = a \times a^{-1} = 1$.

If the field has a finite set of elements, it is called a finite (or Galois) field. Numbers in the field $\mathbb{F}_2$ can be represented by $\{0, 1\}$ and numbers in $\mathbb{F}_{2^n}$ can be represented as n-bit binary numbers.

3.2. Elliptic curves over $\mathbb{F}_{2^n}$

In this section, a group operations on elliptic curves over $\mathbb{F}_{2^n}$ is described.

A nonsupersingular elliptic curve E over $\mathbb{F}_{2^n}$, $E(\mathbb{F}_{2^n})$ is the set of all solutions to the following equation [7].

$$y^2 + xy = x^3 + a_2x^2 + a_6 \tag{1}$$

where $a_2, a_6 \in \mathbb{F}_{2^n}$, and $a_6 \neq 0$. Such an elliptic curve is a finite abelian group. The number of points in this group is denoted by $\#(E(\mathbb{F}_{2^n}))$.

Curve addition: If $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ are points on the elliptic curve [i.e., satisfy (1)] and $P \neq -Q$ then $(x_3, y_3) = R = P + Q$ can be defined geometrically (Figure.2).

In the case that $P \neq Q$ (i.e., point addition), a line intersecting the curve at points P and Q and must also intersect the curve at a third point $-R = (x_3, -y_3)$.

If $P = Q$ (point doubling), the tangent line is used (Figure.3).

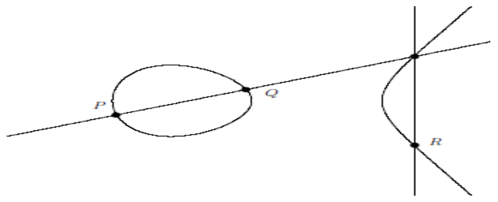

Fig. 2. Group law of elliptic curve (point addition).

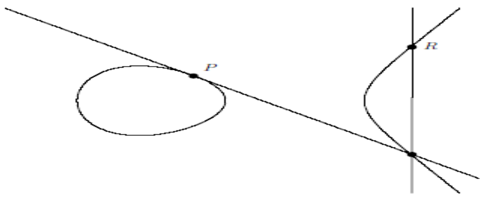

Fig. 3. Group law of elliptic curve (point doubling).

For E given in affine coordinates:
if $P \neq Q$

$$\begin{aligned} x_3 &= \lambda^2 + \lambda + x_1 + x_2 + a \\ y_3 &= \lambda(x_1 + x_3) + x_3 + y_1 \\ \text{où } \lambda &= \tfrac{(y_2+y_1)}{(x_2+x_1)} \end{aligned} \tag{2}$$

if $P = Q$

$$\begin{aligned} x_3 &= \lambda^2 + \lambda + a \\ y_3 &= x_1^2 + (\lambda + 1)x_3 \\ \text{où } \lambda &= x_1 + \tfrac{y_1}{x_1} \end{aligned} \tag{3}$$

4. ELLIPTIC CURVE POINT MULTIPLICATION

There are different ways to implement point multiplication: binary, signed digit representation (NAF), Montgomery method,... A scalar multiplication is performed in three different stages. At the top level, the method for computing the scalar multiplication must be selected, in the second level, the coordinates to represent elliptic points must be defined. From this representation, the add operation is defined. Possible coordinates are: affine, projective, Jacobeans and L'opez-Dahab. The lower level, but the most important, involves the primitive field operations on which the curve is defined. Basic field operations are sum, multiplication, squaring and division.

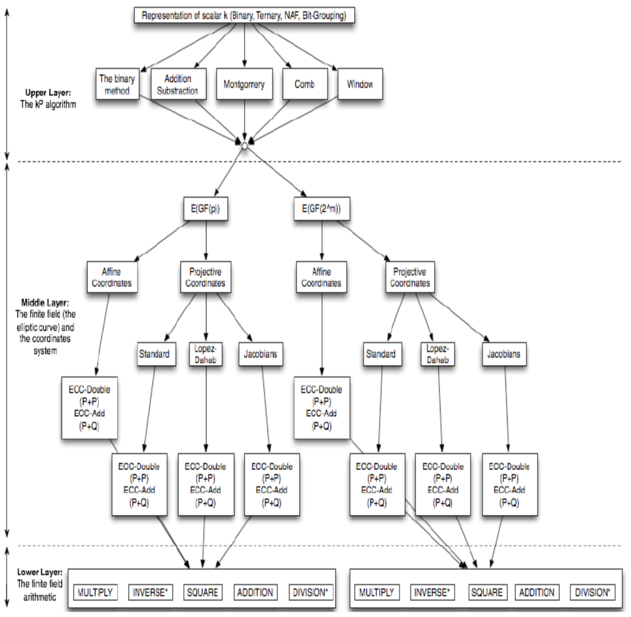
Representation of scalar k (Binary, Ternary, NAF, Bit-Grouping)
The binary method
Addition Substraction
Montgomery
Comb
Window
Upper Layer: The kP algorithm
E(GF(p))
E(GF(2^m))
Affine Coordinates
Projective Coordinates
Middle Layer: The finite field (the elliptic curve) and the coordinates system
Standard
Lopez-Dahab
Jacobians
ECC-Double (P+P) ECC-Add (P+Q)
Lower Layer: The finite field arithmetic
MULTIPLY
INVERSE*
SQUARE
ADDITION
DIVISION*

Fig. 4. Different method to compute scalar multiplication $k.P$

4.1. Binary method

The most simplest and straightforward implementation is the binary method (as shown in Algorithm.1); The binary method scans every bit of scalar k and, depending on its value, 0 or 1, it performs an ECC-DOUBLE operation or both a ECC-DOUBLE and an ECC-ADD operation. Algorithm.1 scans every bit of k from right to left. This allows to perform the operations ECC-DOUBLE and ECC-ADD in parallel.

For an elliptic curve defined on $\mathbb{F}_{2^m}$ using affine coordinates, the operations ECC-ADD and ECC-DOUBLE are performed according to equations (2) and (3) respectively. The operation ECC-ADD requires one inversion, two multiplications, one squaring and eight additions. The operation ECC-DOUBLE requires five additions, two squaring, two multiplications and one inversion, all of them, operations on $\mathbb{F}_{2^m}$.

Algorithm 1 *Binary method: right to left* [7]

```
Input: $P(x, y)$, $x, y \in GF(2^m)$, $k = (k_{m-1}, k_{m-2}, \ldots, k_0)$
Output: $R = k.P$
1: $R \leftarrow 0$
2: $S \leftarrow P$
3: for $i \leftarrow 0, m-1$ do
4:     if $k_i = 1$ then
5:         if $R = 0$ then
6:             $R \leftarrow S$
7:         else
8:             $R \leftarrow R + S$
9:         end if
10:    end if
11:    $S \leftarrow 2S$
12: end for
13: return $R$
```

5. APPLICATIONS OF ECC

5.1. Diffe-hellman

The Diffie-Hellman protocol is the basic public-key cryptosystem proposed for secret key sharing. If A (Alice) and B (Bob) first agree to use a specific curve, field size, and type of mathematics. They then share the secret key by process as follows. We can see that we just need scalar multiplication in order to implement the Diffie-Hellman protocol.

Algorithm 2 *Diffie-Hellman Protocol*

```
1: $A$ and $B$ each chose random private key $k_a$ and $k_b$
2: $A$ and $B$ each calculate $k_aP$ and $k_bP$, and send them
   to opposite side.
3: $A$ and $B$ both compute the shared secret $Q = k_a(k_bP) = k_b(k_aP)$.
```

5.2. Elliptic curve digital signature algorithm

EC Digital Signature Algorithm is the elliptic curve analogue of the DSA, this protocol needs not only the elliptic curve operations, such as scalar multiplication, field multiplication and field inverse multiplication, but also integer multiplication, inverse operation, modular operation and a hash function. In the ECDSA, A (Alice) generates the signature with his secret key and B (Bob) verifies the signature with A's public key. Algorithm.3 is the ECDSA protocol which A signs the message m and B verifies A's signature.

Algorithm 3 *ECDSA Protocol*

```
Key generation : (A)
1: Select a random integer $d$ from $[1, n-1]$.
2: Compute $Q = d.P$.
3: A's public key is $Q$; A's private key is $d$.
Signature generation : (A)
1: Select a random integer $k$ from $[1, n-1]$.
2: Compute $k.G = (x_1, y_1)$ and $r = x_1$(mod n).
3: If $r = 0$ then go to step 1.
4: Compute $k^{-1}$(mod n).
5: Compute $s = k^{-1}(SHA-1(m) + dr)$(mod n).
6: If $s = 0$ then go to step 1.
7: Send $m$ and $(r, s)$, which is A's signature for the message $m$, to B.
Signature verification : (B)
1: Verify that $r$ and $s$ are integers in $[1, n-1]$.
2: Compute $e = SHA-1(m)$.
3: Compute $w = s^{-1}$(mod n).
4: Compute $u_1 = e * w$(mod n) and $u_2 = r * w$(mod n).
5: Compute $u_1P + u_2Q = (x_1, y_1)$ and $v = x_1$(mod n).
6: If $s = 0$ then go to step 1.
7: Accept the signature if and only if $v = r$.
```

6. AN ELLIPTIC CURVE PROCESSOR

Figure.5. shows a structure of ECC processor. It consists of a main control block, an ECC ADD and Double block and an ECC block for arithmetic operations. To implement Elliptic curve protocols, we should use curves which are a SEC-2 recommendation [8].

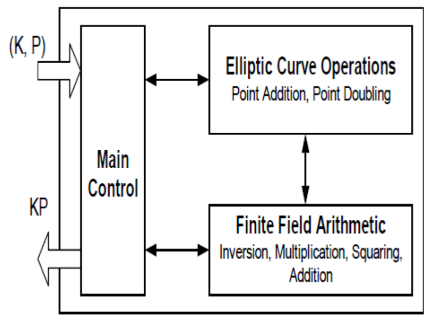
(K, P)
KP
Main Control
Elliptic Curve Operations
Point Addition, Point Doubling
Finite Field Arithmetic
Inversion, Multiplication, Squaring, Addition

Fig. 5. Elliptic curve point multiplication processor.

7. PERFORMANCE OF ECC AND COMPARAISON

The inverse operation of ECC which known as the Elliptic Curve Discrete Logarithm Problem (ECDLP) gets harder, faster, against increasing key length than do the inverse operations in Diffie Hellman and RSA. As security requirements become more stringent, and as processing power gets cheaper and more available, ECC becomes the more practical system for use. And as security requirements become more demanding, and processors become more powerful. This keeps ECC implementations smaller and more efficient than other implementations.

ECC can use a considerably shorter key and offer the same level of security as other asymmetric algorithms using much larger ones. Moreover, the difference between ECC and its competitors in terms of key size required for a given level of security becomes dramatically more pronounced, at higher levels of security.

7.1. Comparison

Comparison between the two asymmetric cryptographic algorithms such as RSA and ECC, Same level of security, data sizes, encrypted message sizes and computational power. But ECC have smaller keys than other cryptographic algorithms (RSA).

Table 2. Equivalent Key Sizes for ECC and RSA.

ECC Key Size (bits)	RSA Key Size (bits)	Key Size ratio
160 bit	1024 bit	1:6
224 bit	2048 bit	1:9
256 bit	3072 bit	1:12
512 bit	15360 bit	1:30

8. CONCLUSION

This is paper gives a clear idea of Elliptic curve cryptography; definition, mathematics and a brief comparative point between ECC and RSA, ECC's advantages and some application of ECC like ECDSA and ECDH. Finally the performance and security of ECC is discussed.

9. REFERENCES

[1] N. Koblitz, "Elliptic Curve Crytosystems," *Mathematics of Computation*, Vol.48, pages 203-209, 1987.

[2] V. S. Miller, "Use of Elliptic Curves in Cryptography," *Advances in Cryptology - CRYTO '85*, Lecture Notes in Computer Science, vol. 128, Springer-Verlag, pages 417-426, 1985, Hugh C. Williams (Ed.)

[3] IEEE P1363, "Standard Specifications for Public Key Cryptography," 2000.

[4] Digital Signature Standard (DSS), *Federal Information Processing Standards Publication 186-2*, National Institute of Standards and Technology. 2000.

[5] Lejla BATINA, "Arithmetic and Architectures for Secure Hardware Implementations of Public-Key Cryptography," *PhD thesis*, KATHOLIEKE UNIVERSITEIT LEUVEN, December 2005.

[6] Sandeep S. Kumar, "Elliptic curve cryptography for constrained devices," *PhD thesis*, Ruhr-University Bochum, June 2006.

[7] D. Hankerson, A. Menezes, and S. Vanstone, "Guide to Elliptic Curve Cryptography," *Springer*, 2004.

[8] SEC 2, "Recommended Elliptic Curve Domain Parameters. Standard for Efficient Cryptography," *The SECG Group*, 2000.