Current Trends in Technology and Sciences
Volume : 1, Issue : 2 , (Sept.-2012) ISSN : 2279-0535

Realization of Elliptic Curve Cryptosystem Based on ECDSA

M. Ashkar Mohammed
Research Scholar, Noorul Islam Centre for Higher Education, Tamil Nadu, India, ashkarmohammed@yahoo.co.in

Dr. S Suresh Babu
Principal, TKM College of Engineering, Kollam, Kerala, India, drssbtkm@gmail.com

Abstract — The elliptic curve crypto systems are used for implementing protocols such as ECDSA digital signature scheme, EC Elgamal Encryption/ Decryption scheme, Diffie-Hellman key exchange scheme and so on. This paper analyzes the elliptic curve operations of the ECC protocol ECDSA. The steps involved in ECDSA are key-pair generation, signature generation and signature verification. The digital signature is typically created using the hash function. The transmitter sends the encrypted data along with the signature to the receiver. The receiver who knows about the senders public key can authenticate the signature using his private key. Thereby ECC ensures the secured data communication. The proposed algorithm is highly parallelizable and well adapted to VLSI implementation of elliptic curve crypto systems.

Keywords — Elliptic curve cryptography (ECC), Elliptic curve discrete logarithm problem (ECDLP), Karatsuba multiplication, message digest, Secure Hash Algorithm (SHA), Elliptic curve digital signature Algorithm (ECDSA).

1. Introduction

The elliptic curve cryptosystems (ECC) were invented around 1985 independently by Miller and Koblitz. Since their introduction a broad discussion on their security and efficiency has been carried on. It is this very efficiency that makes them so interesting for us today. This is due to the fact that information technology is developing very fast. Today we use handhelds and mobile phones as we have a need in securing communications on these devices. But several constraints like limitations in memory, computing power, bandwidth requirement etc need to be considered. What we need is a cryptosystem with small keys, and a small signature size. Efficient encryption/decryption is not so important because these operations are usually done with a private key cryptosystem.

ECC has exactly the desired properties. This comes from the fact, that there are no sub exponential algorithms for the ECDLP (elliptic curve discrete logarithm problem) known today. This means that we can use shorter keys (compared to other cryptosystems) for high security levels.

The ECDSA is the elliptic curve analog of the DSA. ECDSA was first proposed in 1992 by Vanstone in response to NIST's (National Institute of Standards and Technology) request for comments on their first proposal for DSS. Digital signature schemes are the counterpart to handwritten signatures[3]. A digital signature is a number that depends on the secret key only known by the signer and on the contents of the message being signed.

2. Elliptic Curve

The elliptic curves are not the same as an ellipse. They are named so because they are described by cubic equations similar to those used for calculating the circumference of an ellipse. An elliptic curve may be defined as a set of points on the co-ordinate planes, satisfying the equation of the form,

$$y^2[+xy] = x^3 + ax^2 + b \qquad (1)$$

The square brackets mean that the term is optional. *x* and *y* are variables, *a* and *b* are constants.

However, these quantities are not necessarily real numbers; instead they may be values from any field. For cryptographic purposes we always use a "finite" field - that is *x, y, a* and *b* are chosen from a finite set of distinct values.

3. Operations In Field $GF(2^m)$

The field $GF(2^m)$ has particularly importance in cryptography since it leads to particularly efficient hardware implementations. Elements of the field are represented in terms of a basis. Most implementations either use a polynomial basis or normal basis. A normal basis is believed to be a more efficient hardware implementation.

3.1. Addition and Squaring

The addition operation over $GF(2^m)$ is simply a bitwise exclusive OR operation. Furthermore squaring is simply a rotate left operation.

3.2. Multiplication

There are many types of multiplication in finite fields namely, Karatsuba multiplication, Comba Multiplication, etc. This paper analyses Karatsuba multiplication.

4. Karatsuba Multiplication

In 1963 A. Karatsuba and Y. Ofman discovered that multiplication of two m bit numbers can be done with a bit complexity of less than $O(m^2)$ using an algorithm now known as Karatsuba multiplication[4]. For multiplication

Current Trends in Technology and Sciences
Volume : 1, Issue : 2 , (Sept.-2012) ISSN : 2279-0535

in *GF*(2^m) the Karatsuba multiplication scheme can be applied as well.

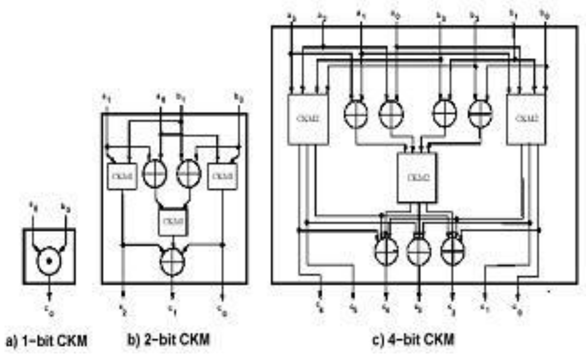

Fig.1. Hardware implementation of Karatsuba multiplication

The fundamental Karatsuba multiplication for polynomials in *GF*(2^m) is based on the idea of divide and conquer, since the operands are divided into two segments. One may attempt to generalize this idea by subdividing the operands into more than two segments. The multiplication over *GF*(2^m) is computed by a single AND operation. The hardware implementation of karatsuba multiplication is done in a recursive process as shown in figure 1.

5. Elliptic Curves Over *GF*(2^m)

This section defines a group constructed from points on elliptic curves over *GF*(2^m) and efficient implementation of operations in this group. A non-singular elliptic curve E over *GF*(2^m) , E(*GF*(2^m)) is the set of solutions to the following equation with coordinates in the algebraic closure of E.

$$y^2[+xy] = x^3 + ax^2 + b \qquad (2)$$

where *a*,*b* are in *GF*(2^m) and *b* is non-zero. Such an elliptic curve is an abelian group.

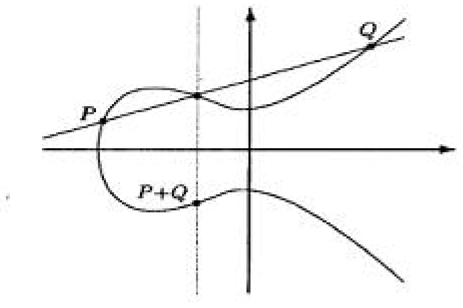

Fig.2. Elliptic Curve

6. Curve Operations

6.1. Curve Addition

The crucial property of an elliptic curve is that , the resultant point obtained by adding two points on the curve is also on the curve. The addition rule satisfies the normal properties of addition. If $P = (x1, y1)$ and $Q = (x2, y2)$ are points on the elliptic curve, the addition rule has the form:

$$(x_1, y_1) + (x_2, y_2) = (x_3, y_3) \qquad (3)$$

$$\text{where; } x_3 = L^2 + L + x_1 + x_2 + a \qquad (4)$$

$$y_3 = L(x_1 + x_3) + x_3 + y_1 \qquad (5)$$

$$L = (y_1 + y_2) / (x_1 + x_2) \qquad (6)$$

If $x_1 = x_2$ and $y_1 = y_2$ we must use instead

$$x_3 = L^2 + L + a \qquad (7)$$

$$y_3 = x_1^2 + (L + 1)x_3 \qquad (8)$$

$$L = x_1 + (y_1/x_1) \qquad (9)$$

Again, there are some other special cases which must be considered first: if $x1 = x2$ and $y2 = x1 + y1$ then the result is zero, and if either point is zero, the result is the other operand. In the case where *P* and *Q* are equal, it is called point doubling whereas if *P* and *Q* are not equal it is called point addition.

6.2. Curve Multiplication

Multiplication is defined by repeated addition, i.e,

$$Q = kP \qquad (10)$$

$$i.e.;\ Q = P + P + P + \cdots\ k \text{ times} \qquad (11)$$

This can be computed using point addition and point doubling.

7. Discrete Logarithm Problem

The Elliptic curve cryptography is based on the discrete logarithm problem applied to elliptic curves over a finite field[1]. In particular, for an elliptic curve E, it relies on the fact that it is easy to compute $Q = kP$, for *k* in *GF*(2^m) and *P*,*Q* in E. However there is currently no known sub exponential algorithm to compute *k* given *P* and *Q*. In fact the discrete logarithm problem can be used to build cryptosystems with finite abelian group. Indeed multiplicative groups in a finite field were originally proposed. However, the difficulty of the problem depends on the group, and at present, the problem in elliptic curve groups is orders of magnitude harder than the same problem in a multiplicative group of a finite field. This feature is a main strength of elliptic curve cryptosystems.

8. Secure Hash Algorithm (SHA-1)

The Hash Algorithm (SHA-1) forms the core of digital signatures. Hashing may be defined as the transformation of a string of characters into a usually shorter and fixed length value or key that represents the original message. SHA-1 mainly consists of three steps:

8.1. Formation of message digest

The formation message digest involves the appending of the message with additional bits resulting in a message digest of 512 bits. It includes the following steps.

Step1 : Append padding bits
The message is so padded so that its length in bits is congruent to 448 modulo 512. i.e. the length of padded message is 64 bits less than an integer multiple of 512 bits. Padding is always added even if the message is already of the desired length. For e.g. if the message is of length 448 bits, 512 bits are added so that length is 960 bits. Thus the number of padding is from 1 to 512 bits. The padding always consists of a 1 bit followed by 0 bits.

Step2 : Append length
A 64 bit representation of the length in bits of the original message(before padding) is appended to the result of step1(least significant byte first). If the original length is greater than 2^{64}, then only lower order bits are used. Thus the field contains the length of the original message, modulo 2^{64} The outcome of the first two steps yields a message that is an integer multiple of 512 bits.

8.2. Initialize buffers
A 160 bit buffer is used to hold the intermediate and final results of the Hash function. The buffer can be represented by five 32 bit registers (A,B,C,D,E). These are initialized to five 32 bit integers. The hexadecimal values are indicated.

Table (1) Buffer values

Word A	67 45 23 01
Word B	EF CD AB 89
Word C	98 BA DC EF
Word D	10 32 54 76
Word E	C3 D2 E1 F0

8.3. Process message digest in blocks
The heart of the algorithm consists of four rounds of processing each of which has 20 steps. The logic is illustrated in figure 3.
Each round takes place as the current input 512 bit message digest block being processed and the 160 bit buffer value as the input and updates the contents of the buffer. Each round also makes use of the additive constant K_t where t ranges from 0 to 79 including one of the 80 steps across the four rounds. In fact only four distinct values of K_t are used.

9. Operations In A Single Step Of SHA

The elementary SHA-1 operation, as shown in figure 4, involves a single 20 steps which is subjected to four rounds to obtain the complete hash process with 80 steps. In the figure 4,
K_t is an additive constant
W_t is a 32 bit word derived from the current 512 bit block
Whatever be the length of the message the final output is only of 160 bits. i.e. hash algorithm results in a message digest of 512 bits.

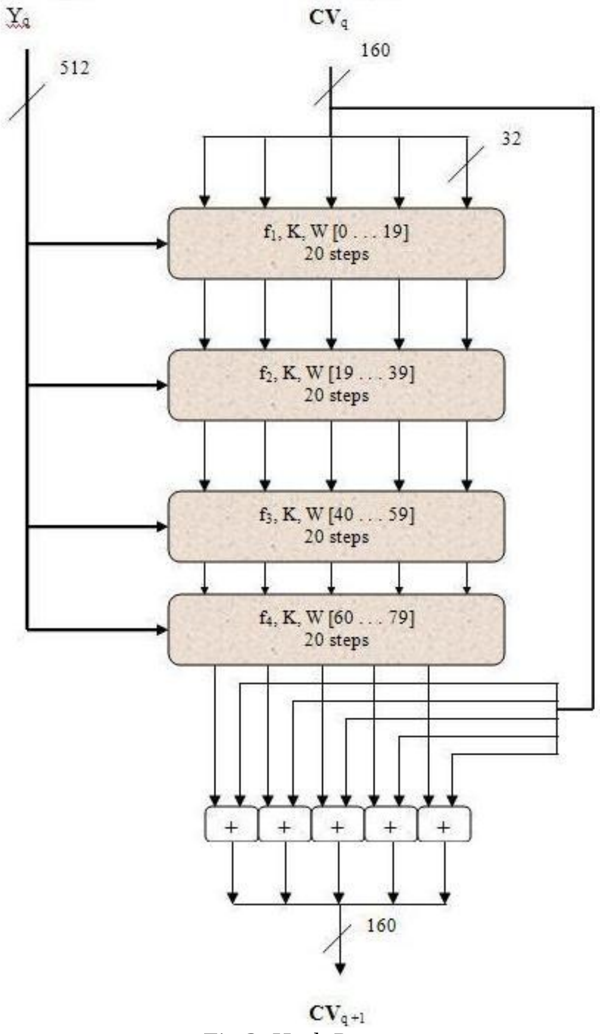
Y_q
CV_q
512
160
32
f_1, K, W [0 . . . 19] 20 steps
f_2, K, W [19 . . . 39] 20 steps
f_3, K, W [40 . . . 59] 20 steps
f_4, K, W [60 . . . 79] 20 steps
+
160
CV_{q+1}
Fig.3. Hash Process

Table (2) Functions in SHA

Ft(A,B,C,D)	Interval
(B ∧ C) ∨ (B` ∨ D)	0≤ t ≤ 19
B XOR C XOR D	20≤ t ≤ 39
(B ∧ C) ∨ (B ∨ (B∧D)) ∨ (C∧D)	40≤ t ≤ 59
B XOR C XOR D	60≤ t ≤ 79

Table (3) Initialization of K_t

Distinct Values of K_t	Interval
5A82799	0≤ t ≤ 19
6ED9EBA1	20≤ t ≤ 39
8F1BBCDC	40≤ t ≤ 59

Current Trends in Technology and Sciences
Volume : 1, Issue : 2 , (Sept.-2012) ISSN : 2279-0535

CA62C1D6	60≤ t ≤ 79

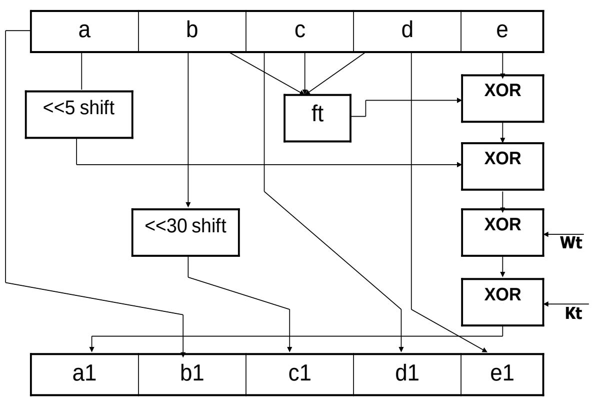

Fig.4. Elementary SHA-1 Operation

10. Elliptic Curve Cryptosystems

The elliptic curve crypto systems are used for implementing protocols such as ECDSA digital signature scheme, EC Elgamal Encryption/Decryption scheme, Diffie-Hellman key exchange scheme and so on[1]. This paper analyzes the elliptic curve operations of the ECC protocol ECDSA.

11. Elliptic Curve Digital Signature Algorithm (ECDSA)

The steps involved in ECDSA algorithm are:

11.1. Key pair generation

Suppose Alice wants to send a digitally signed message to Bob:

(a) Select a random integer, '*d*' in the interval [*0, n-1*].

(b) Compute $Q = dG$, Obtained by Karatsuba Multiplication. *Q*,*G* are points on the elliptic curve.

(c) Now Alice's key-pair is (d,Q) where *d* is the Private key and *Q* is the Public key.

11.2. Signature generation

(a) Choose a random number *k* with *k* : $1\leq k\leq n\text{-}1$.

(b) Compute $kG = (x1, y1)$ and $r = x1 \bmod n$. If $r = 0$ then go to step a.

(c) Compute $k^{-1} \bmod n$.

(d) Compute $e = SHA\text{-}1(M)$.

(e) Compute $s = k^{-1}(e + dr) \bmod n$. If $s = 0$ then go to step a.

(f) Alice signature for the message M is (r, s).

11.3. Signature verification

(a) Verifiy that r, s are integers in the inteval [*1, n-1*].

(b) Compute $e = SHA\text{-}1(M)$.

(c) Compute $w = s^{-1} \bmod n$.

(d) Compute $u1 = ew \bmod n$ and $u2 = rw \bmod n$.

(e) Compute $X = u1G + u2Q$. If $X = 0$ then reject

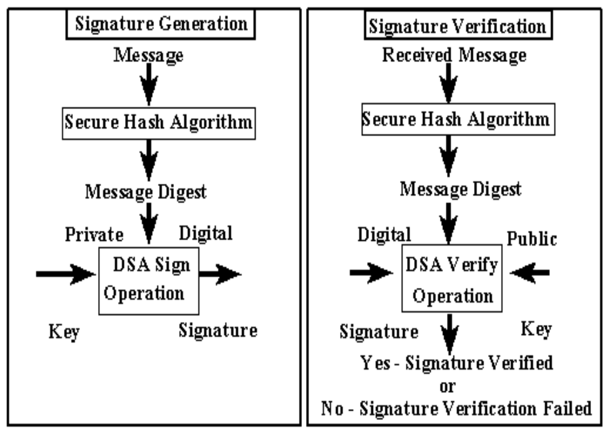

Fig.5. Digital Signature Process

the signature. Otherwise compute $v = x1 \bmod n$ where $X = (x1, y1)$.

(f) Accept the signature if and only if $v = r$.

The figure 5 shows the generation and verification of digital signatures . Both process use the Hash function of the message there by resulting in the message digest. The transmitter sends the message which may or may not be encrypted, along with the sign to the receiver. The receiver also finds the Hash of the received message and uses the received sign and the senders public key to verify the signature. The generation and verification are based on the ECDSA algorithm.

Appendixes, if needed, appear before the acknowledgment.

12. Future Scope And Conclusion

ECC offers the same level of security as that of RSA with minimum number of bits, i.e. with small keys and small signature size. Through this project we present an efficient multiplication scheme that effectively enhances the security level of data with small keys and small signature size. Also the algorithm provides improved authentification through verification of signatures at the receiving side. A high performance, generic projective coordinate algorithm proposed by Lopez and Dahab can be used[2], which is an efficient implementation of Montgomerys method for computing *kP* which requires no precomputations or special field/curve properties. It can offer the best performance for both point addition and point doubling[5]. The proposed algorithm is highly parallelizable and well adapted to VLSI implementation of elliptic curve crypto systems.

Reference

[1] N. Koblitz, A. Menezes, and S. Vanstone, "The state of elliptic curve cryptography", Designs, Codes, Cryptography, vol. 19, pp. 173-193,2000.

[2] J. Lopez and R. Dahab, “Fast multiplication on elliptic curves over GF(2^m) without precomputation”, presented at the Workshop on Cryptographic Hardware Embedded Syst. (CHES), Worcester, MA,1999.

[3] Proceedings of the 16th IEEE Symposium on Computer Arithmetic (ARITH‘03) 2003 IEEE.

[4] A. Karatsuba and Y. Ofman, “Multiplication of multidigit numbers on automata” Sov. Phys.-Dokl, Engl. transl., vol. 7, no. 7, pp. 595-596, 1963.

[5] William N. Chelton and Mohammed Benaissa, “Fast Elliptic Curve Cryptography on FPGA”, in IEEE Transactions on VLSI Systems, vol.16, Feb 2008.