International Journal of Computer Applications (0975 – 8887)
Volume 68– No.21, April 2013

Design and Implementation of Secure Stream Cipher based on Elliptic Curves on Time Shared Basis

K S Lalmohan
National Institute of Technology
NITC Post
Calicut, India

Deepthi P P
National Institute of Technology
NITC Post
Calicut, India

Sathidevi P S
National Institute of Technology
NITC Post
Calicut, India

ABSTRACT

This paper proposes the implementation of a Elliptic Curve (EC) cryptosystem which is aimed to provide secure stream ciphers, hash functions and key exchange in a time shared manner. The design of hardware efficient stream cipher based on elliptic curves proposes the use of point multiplication block on a time sharing basis for providing secure stream ciphers, hash generation and key exchange. The EC point multiplication uses the Gaussian normal bases for field arithmetic. The designs were implemented using Verilog language and the hardware implementation was done using a Field Programmable Gate Array (FPGA) device.

General Terms

Security, Data and Information Systems, Digital Systems, Cryptography

Keywords

Hash function, Elliptic curve cryptosystems, Stream cipher, Gaussian normal basis, Finite fields, FPGA

1. INTRODUCTION

E-commerce applications and emerging communication between people or agencies give rise to an important question, as how to reliably exchange confidential data via untrusted channel of communication[1]. Data transfer must be protected in the sense that it has to be ensured that exchanged documents are neither read nor modified by third parties during the data transfer. The technology which can provide this kind of protection is cryptography. The main objective of cryptography is the design and analysis of systems that ensure secure communication. The major services offered by cryptography are 1) key exchange 2) message encryption and 3) data integrity. In majority of the secure communication systems these three functions are done by three different modules implementing different algorithms like RSA[2] or ECC for key exchange, AES, DES, E0 cipher or A5 stream cipher for message encryption and SHA or HMAC algorithms for hash generation. Implementation of these independent modules increases the hardware complexity for small handheld devices where resources are limited. Since the operations are done sequentially, a hardware structure which consists of a module based on a single cryptographic primitive used on a time sharing basis for all operations will be highly acceptable for small battery powered devices like RFID tags, smart cards etc.

The widely used RSA method and public key schemes based on Elliptic Curves (EC) have gained more importance. In 1985 Elliptic Curve Cryptography (ECC) was first proposed by Miller (1986) [3] and Koblitz (1987) [4]. ECC is an attractive public-key cryptosystem based on the algebraic structure of elliptic curves over finite fields for mobile/wireless environment. Compared to traditional cryptosystems like RSA, ECC offers equivalent security with smaller key sizes, which results in faster computations, lower power consumption, as well as memory and bandwidth savings. As ECC is replacing RSA, EC point multiplication unit will be available in hardware structure of majority of communication systems. If the other two functions of message encryption and hash generation are implemented based on point multiplication then, this will reduce the hardware complexity and will be useful for mobile devices which are typically limited in terms of their CPU power and network connectivity.

The message encryption can be done with either stream cipher or block cipher. In block ciphers, the computation is done on blocks of data which results in increased buffer size. But in stream ciphers the data is encrypted bit by bit which results in lesser resource requirement. Also it is more suitable for real time operations. The major requirement for stream cipher generation is a pseudo-random number generator. A number of algorithms for pseudo-random number generation based on elliptic curve is available in literature. The one which is most suitable for hardware implementation and has increased security is reported in [5], and is used for implementation in this work. Also, a new method for hash generation based on EC point multiplication is proposed.

Underlying field arithmetic is important for the implementations of ECC[6]. For the field arithmetic in GF(2^m), two typical bases, namely polynomial basis (PB) and normal basis (NB), are used to represent field elements of GF(2^m). Each basis representation has its own advantages and disadvantages. However, hardware structures, in NB representation, may be quite different for varying choices of m, although the multiplication algorithm is basically same for each m. Unlike NB representation, PB representation provides the features of regularity, scalability, and extensibility in hardware implementations for various m.

Gaussian normal basis (GNB), a special class of normal basis[7,8], has recently received considerable attention[9]. GNB has been included in a number of standards such as IEEE 1363 [10] and NIST [11]. It is well known that GNB exists for every positive integer m that is not divisible by eight [12]. GNB is determined by an integer k and is referred to as type k GNB. The complexity of GNB multiplier in terms of timing and hardware depends on k. In other words, when there is more than one GNB for a given m, the smallest value of k yield efficient implementation of GF(2^m) multiplier. Kwon et al. [13] proposed an efficient bit level multiplication algorithm for GF(2^m) using GNB and provided VLSI architectures in the cases of GNB of type 2 and 4.

Reconfigurable hardware, such as a Field Programmable Gate Array, provides an attractive alternative to costly custom ASIC fabrication for deploying custom hardware. While ASIC fabrication requires very high non-recurring engineering (NRE) costs, an SRAM-based FPGA can be programmed after fabrication to be virtually any circuit. Moreover, the configuration can be updated an infinite number of times.

Thus, this work aims to design and implement a hardware for secure communication based on ECC with a single module of EC point multiplication used in a time sharing basis for key exchange, message encryption and hash generation there by reducing the hardware complexity.

In this paper, we propose a high performance elliptic curve cryptosystem over GF(2^m). The proposed architecture is based on standard elliptic curve point multiplication algorithm and uses GNB for GF(2^m) field arithmetic. Three major characteristics of the proposed architecture are (1) it uses fast arithmetic units based on a word-level multiplier, (2) it adopts a parallelized point doubling and point addition unit with uniform addressing mode, and (3) it utilizes benefits of GNB representation. Therefore, the proposed architecture leads to a considerable reduction of computational delay time compared with previously proposed hardware implementations.

The remainder of the paper is organized as follows: In Section 2, the mathematical background of elliptic curves is discussed, Section 3 EC based Stream ciphers are introduced. In section 4 key exchange and hash generation based on EC is presented. In Section 5, hardware structure for EC based point multiplication is discussed.

2. MATHEMATICAL BACKGROUND

An Elliptic curve E over a field K is defined by the Weierstrass equation given by

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \qquad (1)$$

where $a_1, a_2, a_3, a_4, a_6 \in K$. The points on the EC denoted as E(K) form an abelian group under addition together with an additional point denoted by ***O*** and called the 'point at infinity', which is the additive identity for the group.

Elliptic curves defined over GF(2^m) has great significance since they allow binary operations and are very suitable for hardware implementation. A Galois Field (finite field) GF(2^m) consists of 2^m elements for some integer ‘m’ together with addition and multiplication operations that can be defined over polynomials in GF(2). Elliptic curves over GF(2^m) are defined by a cubic equation in which the variables and coefficients take on values in GF(2^m). So, all mathematical operations on EC are performed using the rules of arithmetic in GF(2^m) [14,15].

Since the characteristic of the finite field GF(2^m) is 2, the equation (1) can be transformed by suitable change of variables to get the following forms

$$y^2 + xy = x^3 + a_2x^2 + a_6 \qquad (2)$$

$$y^2 + a_3y = x^3 + a_4x + a_6 \qquad (3)$$

The set **E** (a_2, a_6) consisting of all pairs of (x, y) that satisfy equation (2) together with the point at infinity ***O*** form an abelian group if $a_6 \neq 0$. This type of curves is obtained if a_1 in equation (1) is non zero. These curves are non-super singular elliptic curves. The set **E** (a_3, a_4, a_6) consisting of all pairs of (x, y) that satisfy equation (3) together with the point at infinity ***O*** form an abelian group if $a_3 \neq 0$. These curves are super singular elliptic curves. This type of curve is obtained if a_1 in equation (1) is zero. Here we will be considering the non-supersingular elliptic curves only as they provide the highest security in GF(2^m).

Rules for addition over non-super singular curves over GF(2^m) can be stated as follows:

For all points **P**, **Q** ∈ **E** (a_2,a_6),

1. **P** + **O** = **P**

2. If **P** = (x_1, y_1), then –**P** = (x_1, x_1+y_1)

3. Addition formula: If **P** = (x_1, y_1) and **Q**=(x_2, y_2), then **P**+**Q** = **R** = (x_3, y_3) is given by the ‘tangent and chord’ method

$$x_3 = (\lambda)^2 + (\lambda) + x_1 + x_2 + a_2$$
$$y_3 = (\lambda)(x_1 + x_3) + x_3 + y_1 \qquad (4)$$

where $\lambda = (y_1 + y_2)/(x_1 + x_2)$

4. Doubling formula: If **P** = (x, y), then **2P** =**R** = (x_3, y_3) is given by

$$x_3 = \lambda^2 + \lambda + a_2$$
$$y_3 = x_1^2 + \lambda x_3 + x_3 \qquad (5)$$

where $\lambda = x_1 + y_1/x_1$

Thus, adding two elliptic curve points (EC-Add) as well as doubling an elliptic curve point (EC-Double) requires one inversion and two multiplications each over the underlying finite field GF2^m.

Computing inverses is relatively expensive in comparison to multiplication in GF(2^m). In order to avoid computing inverses the point P(x,y) in affine coordinates can be converted to projective coordinate as (x,y,1). A point P(X, Y, Z) in projective coordinates can be converted to affine coordinates as (X/Z, Y/Z) provided $Z \neq 0$. Z = 0 implies a point at infinity. For projective coordinates representation of the affine points, the common denominator for X and Y coordinates are taken as Z coordinate [14] and [15]. The projective equation of the EC is given by

$$Y^2Z + XYZ = X^3 + a_2X^2Z + a_6Z^3 \qquad (6)$$

Elliptic curve addition:

Let $P = (X_1 : Y_1 : Z_1)$, $Q = (X_2 : Y_2 : Z_2)$ such that $P \neq \pm Q$ then $P + Q = R = (X_3 : Y_3 : Z_3)$ is given by

$A = Y_1Z_2 + Z_1Y_2$, $\quad B = X_1Z_2 + Z_1X_2$, $\quad C = B^2$,

$D = Z_1Z_2$, $\quad E = (A^2 + AB + a_2C)D + BC$,

$X_3 = BE$, $Y_3 = C(AX_1 + Y_1B)Z_2 + (A + B)E$, $Z_3 = B^3D$. (7)

Elliptic curve doubling:

If $P = (X_1 : Y_1 : Z_1)$ then $2P = R = (X_3 : Y_3 : Z_3)$ is given by

$A = X_1^2$ $\quad B= A + Y_1Z_1$, $\quad C= X_1Z_1$,

$D = C^2$, $\quad E= (B^2 + BC + a_2D)$,

$X_3 = CE$, $Y_3 = (B + C)E + A^2C$, $Z_3 = CD$. (8)

Thus in projective coordinates, no inversion is needed.

2.1 Point Multiplication on Elliptic Curves

If P is a point on the elliptic curve and ‘k’ is any integer, computing a new point ‘kP’ returns another point on EC. This operation is called point multiplication operation on EC. The EC point multiplication is computed by repeated point additions which is same as adding the point P to itself 'k' times and the points on an EC form an abelian group under point addition operation. The point multiplication operation can be implemented with a number of point addition and doubling operations. For example, 7P can be written as 2(2P+P) + P, which can be implemented as a combination of two doubling and two addition operation. Thus for a given point P on the EC and any integer 'k', computation of kP is easy. But, at the same time, computing k from R and P is extremely difficult. This is called the *elliptic curve discrete logarithm problem* (ECDLP). The EC operations in turn are composed of basic operations in the underlying finite field (FF or GF)[16,17]. Since computation in one direction is easy while that in the opposite direction is difficult, elliptic curve point multiplication is a one-way function. This is the underlying mathematical problem that provides security and strength to EC-based crypto-schemes. Non-supersingular curves are considered to be more secure compared to supersingular elliptic curves.

2.2 Finite Field Arithmetic

2.2.1 Galois Field Addition

If A = ($a_0, a_1, \ldots a_{m-3}, a_{m-2}, a_{m-1}$) and B = ($b_0, b_1, \ldots b_{m-3}, b_{m-2}, b_{m-1}$) are elements of GF($2^m$), then the sum C =A+B = ($c_0, c_1, \ldots c_{m-3}, c_{m-2}, c_{m-1}$), where $c_i = a_i \oplus b_i$. Therefore sum can be obtained as bitwise XORing of A and B.

2.2.2 Galois Field Squaring

Squaring of an element A in the normal basis representation is a cyclic shift operation. Hence, the hardware implementation of squaring operation requires only a shift register.

2.2.3 Galois Field Multiplication

Gaussian normal bases (GNB) for *GF* (2^m) exist whenever *m* is not divisible by 8. They include the *optimal normal bases* (ONB)[18,19], which have the simplest and most efficient multiplication possible in a normal basis. The type of a GNB is a positive integer measuring the complexity of the multiplication operation with respect to that basis. The smaller the type, the more efficient is the multiplication. For a given *m* and *T* (type T Gaussian normal basis), the field *GF* (2^m) can have at most one GNB of type *T* [20, 21]. A table of Gaussian normal bases is given in [10]. The Gaussian normal bases of types 1 and 2 have the most efficient multiplication rules for all normal bases. For this reason, they are called *optimal* normal bases.

Let p= 2m + 1 be a prime ≠ 2 and let $\mathrm{ord}_p 2$, be the order of 2 (mod p) such that gcd($2m/\mathrm{ord}_p 2$, m) = 1 where i.e. either 2 is a primitive root (mod p) or $\mathrm{ord}_p 2$ = m and m is odd. Then the element $\alpha = \beta+\beta^{-1}$ where β is a primitive *p*th root of unity in GF(2^{2m}) forms a normal basis {$\alpha_0, \alpha_1, \ldots, \alpha_{m-1}$} in GF($2^m$) which is a Gaussian normal basis of type 2 (or a type II ONB). Thus from [13] the multiplication of two elements A and B can be represented as C = AB, where

$A = \sum_{i=0}^{4} a_i\alpha_i, \quad B = \sum_{i=0}^{4} b_i\alpha_i, \quad C = \sum_{i=0}^{4} c_i\alpha_i$

We briefly explain, as an example, a word-level multiplier over GF(2^5) which has GNB of type 2. The multiplication result is as follows (see [13] for details).

$$C_0 = \underline{(a_3+a_4)b_2} + a_1b_0 + (a_1+a_2)b_3 + (a_0+a_3)b_1 + (a_2+a_4)b_4$$

$$C_1 = (a_4+a_0)b_3 + \underline{a_2b_1} + (a_2+a_3)b_4 + (a_1+a_4)b_2 + (a_3+a_0)b_0$$

$$C_2 = (a_0+a_1)b_4 + a_3b_2 + \underline{(a_3+a_4)b_0} + (a_2+a_0)b_3 + (a_4+a_1)b_1$$

$$C_3 = (a_1+a_2)b_0 + a_4b_3 + (a_4+a_0)b_1 + \underline{(a_3+a_1)b_4} + (a_0+a_2)b_2$$

$$C_4 = (a_2+a_3)b_1 + a_0b_4 + (a_0+a_1)b_2 + (a_4+a_2)b_0 + \underline{(a_1+a_3)b_3}$$

The underlined entries are the first terms to be computed. The shifted diagonal entries have the common terms of a_i's. The product C using a type II ONB in GF(2^m) for m =5 is shown in Fig.1.

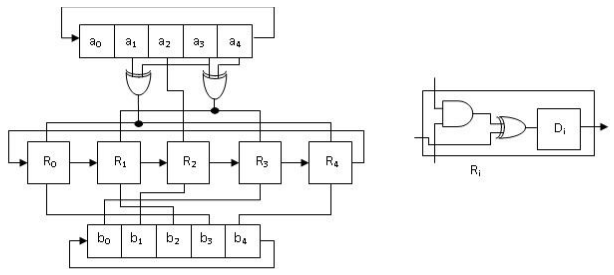

Fig 1. A multiplication circuit using GNB in GF(2^m) for m = 5

3. EC BASED STREAM CIPHERS

Stream ciphers allow real time operation, which is usually not possible with block cipher encryption. The need for buffer space is very less in stream ciphers, since data is operated bit by bit. Most critical step in the design of a stream cipher is the design of a cryptographically strong pseudorandom bit sequence generator (CSPBSG)[22, 23]. Two main approaches to implement CSPBSG are (i) using cryptographic one-way function and (ii) using Linear Feedback Shift Register (LFSR) based structures[24,25]. LFSR based systems are less complex in hardware compared to one-way function based structures. Even though one-way function based stream ciphers have increased hardware complexity compared to LFSR based structures, if the underlying one-way function is used for the implementation of some other cryptographic services such as authentication or key exchange, then the redundant hardware in the system can be reduced. The operations of key exchange, encryption and authentication are done sequentially. Thus an encryption system built using infrastructure available for key exchange or authentication is highly acceptable. This approach will reduce the overall hardware complexity of the cryptosystem.

3.1 PRBS Generation Based on One-Way Functions

A pseudorandom bit sequence generator (PBSG) is a deterministic algorithm which takes a random binary sequence of length k and outputs a binary sequence of length n»k which ‘‘appears” to be random. The input to the PBSG is called the seed, while the output is called a pseudorandom bit sequence. Generation of pseudorandom bit sequences from a one-way function *f* can be briefly explained as follows: First select a random seed S_0, and apply the function *f* to the seed to generate *f*(S_0). Now, apply a predicate B on *f*(S_0) and output

B(f(S_0)). Then, advance to the next seed S_1, and output B(f(S_1)) etc. Different EC based PBSG schemes suggested in the literature use different ways to proceed from seed for ith iteration to that for (i+1)th iteration and different predicates B, while the one way function f is the EC point multiplication operation[26].

3.2 Stream Cipher Generation Based on Elliptic Curve

Various suggestions are available in literature for EC based stream cipher generation. The properties such as periodicity, throughput and security of the key stream produced by these algorithms are thoroughly analyzed in [5]. Since Elliptic curve encryption demands mapping the message to a point on the Elliptic curve, it can be implemented only as an iterative process which is very complex. Hence methods of encryption based on EC which demand much lower hardware complexity need to be investigated. Stream cipher generation based on EC with low hardware complexity is reported in [5, 27].

The algorithm is basically an improvement of Linear Congruential Generator based on EC. It performs a random walk through points on elliptic curve starting from a secret seed point P. Random walk is performed by choosing a random integer ‘k’ given out by an LFSR and computing ‘kP’. The LFSR can be used as a counter whose count sequence is determined by the feedback polynomial. If the feedback polynomial is a primitive polynomial of degree ‘m’, the LFSR content passes through all 2^m-1 nonzero values starting from an initial value. For example, let the length of the LFSR be 5. Then, one possible primitive polynomial for being used as feedback polynomial is $x^5 + x^3 + 1$. Let the initial count in this LFSR be given as $C_0 = (c_0, c_1, c_2, c_3, c_4)$. Then for the next clock, the contents of LFSR gets modified to $C_1 = ((c_2+c_4), c_0, c_1, c_2, c_3)$ and so on. The constant used for multiplying the seed point P for i^{th} iteration is i^{th} count C_i of the LFSR. Thus by combining LFSR with EC point multiplication operation, the required randomness is ensured by the LFSR while security is ensured by the EC point multiplication. The secret integer for i^{th} iteration is i^{th} state of LFSR starting from an initial key. Once the LFSR passes through all possible states starting from the initial key, it is re-loaded by the ***x*** coordinate of the output point for the iteration i.e., X(C_nP). The outer loop can be chosen to run the required number of times ‘k’. The value of ‘k’ can be any value chosen at liberty in such a way that randomness properties of the sequence is not destroyed, the order of the point will not be revealed by the period of the sequence. Thus, the new algorithm helps to increase the security of the sequence in two different ways: (i) by removing the symmetry properties in the binary sequence (ii) by hiding the order of the seed point. The cryptanalysis of this proposed cipher can be done only by making brute-force trials on points on elliptic curve as seed point. Authors experimentally proved that this method of iteration can provide good security and randomness. Output bits at any instant are generated from X and Y coordinates of output point by applying a trace operation.

Algorithm

Let E be a non-super singular elliptic curve over GF(2^m). Let P be an affine point of order l +1 on an E. P is the secret key. L = $\log_2 (l + 1)$

```
Step 1: Load LFSR of length ‘L’ with some known non-zero integer $C_1$.
Step 2: for j = 1 to k
For $i$ = 1 to n
Get the LFSR count $C_i$.
$S_i \leftarrow C_i P$
$s(2i-1) = Tr(X\text{-co-ordinate}(S_i))$
$s(2i) = Tr(Y\text{-co-ordinate}(S_i))$
Advance the LFSR to the next count.
End for;
$C_j = X(C_nP)$
End for
Step 3: Return (s)
```

For an EC defined over GF(2^m) , the rough estimate of the number of trials to be done for finding the seed point can be given as 2^m. Thus the security of the above algorithm is O(2^m) for an EC defined over GF(2^m). Since the predicate Trace function is applied on output points S_i to generate bit stream {s}, the output points S_i cannot be traced back from output bit stream {s}. The Trace function (Tr) applied on output points to generate output bit stream, is bitwise XOR. The security of this cipher is increased further by making C_i a part of secret information. Then the security is even more stronger than solving ECDLP. This is because, even if the attacker gets point S_i, from bitstream s_i, to get back P, he should solve $P = C_i^{-1}S_i$ where C_i is not known. Thus the algorithm produces a sequence of large periodicity and good security.

4. KEY EXCHANGE AND HASH GENERATION USING EC

Two kinds of cryptosystems that implement cryptographic algorithms are private key cryptosystem and public key cryptosystem. In a private key cryptosystem both communicating entities share a secret key through a secure and authenticated channel. This secret key is used for both encryption and decryption of data. Private Key cryptography is used for the encryption of data due to its speed and reduced complexity of operations. However, it has certain shortcomings that make it unsuitable for use in today’s environment.

Key Management Problem : In a broadcast communication scenario, each user will have to communicate with many different ones. Thus, communication on a public network is not restricted to one-on-one. For a network of n users, $n(n-1)/2$ private keys need to be generated. When n is large, the number of keys becomes unmanageable.

Key Distribution Problem : With such a large number of keys that need to be generated on a network, the job of generating the keys and finding a secure channel to distribute them becomes a burden.

No digital signatures possible : A digital signature is an electronic analogue of a handwritten signature. If Alice sends an encrypted message to Bob, Bob should be able to verify that the received message is indeed from Alice. This can be done with Alice’s signature; however, private key cryptography does not allow such a feature. In contrast, public key cryptography uses two keys. Each user on a network publishes a public encryption key that anyone can use to send them messages, while keeping the private key secret for decryption. On a network of n users, it only needs n public and n private keys. Furthermore, it allows the use of digital signatures, which ensures non-repudiation. However, public key cryptography does have its drawbacks.

In truth, public and private key cryptography work best together. Public key cryptography is ideal for key distribution and management, ensuring *data integrity*, providing *authentication* and *nonrepudiation*, while private key cryptography is ideal for ensuring *confidentiality*, such as encrypting data and communication channels. Thus in this hardware implementation public key cryptography is used for key exchange and private key cryptography is used for message encryption.

4.1 Elliptic Curve Diffie-Hellman Key Agreement Scheme

In Key-Agreement schemes (KAS), users can exchange and establish keys by means of an interactive protocol. Key-agreement schemes are public-key cryptography based and are used to initiate a conversation between two introduced users. The Diffie-Hellman Key agreement scheme is implemented on Elliptic Curve defined over GF(2^m) and a point P of order n which is known to the public. Let A and B be the communicating entities. The various steps in key exchange process are

1. A chooses a random integer X, computes XP and sends the partial key to B.

2. B chooses a random integer Y, computes YP and sends the partial key to A.

3. On receipt of A's message, B computes Y(XP) = XYP.

4 .On receipt of B's message, A computes X(YP) = XYP.

Now, both A and B share a common secret, XYP which is a point on the given elliptic curve. But for an eavesdropper who sees the partial secrets, XP and YP, recovering the key 'XYP' requires solving the ECDLP (Elliptic Curve Discrete Logarithm Problem).

4.2 Hash Generation Based on EC Point Multiplication

A hash function is a one-way function that maps binary strings of different lengths to a binary string of fixed length, called the hash value. It is widely used in secure communication systems for message authentication and data integrity verification. The hash value produced is a function of all the bits of the input message and provides an error detection capability. A change in a single bit of the input message causes a change in the hash code. At the same time many messages can have the same hash value as it is a many to one mapping. Commonly used hash algorithms are MD5, SHA, HMAC etc.

As the aim of this work is to implement a secure communication system of low hardware complexity by using a single module of EC point multiplication for all the three functions on a time sharing basis, the hash function used for implementation is a point multiplication operation as explained below.

The mathematical expression of the hash function is

$H(m) = (x_1 + nx_2)P$,

where x_1 and x_2 are the residues of modular division of the message string with generator polynomials $g_1(x)$ and $g_2(x)$ and 'n' is any arbitrary integer which is kept as a constant. The operation nx_2 is implemented as a GF multiplication. P is the point on the elliptic curve which is the key generated by the Diffie-Hellman key exchange algorithm as given Fig.2.

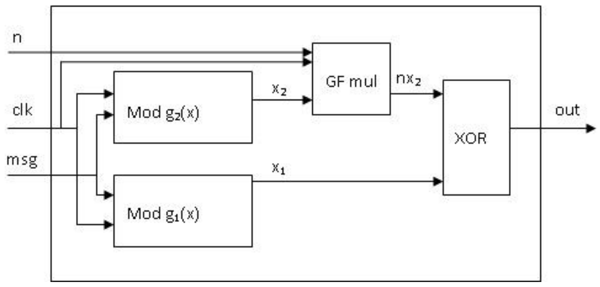

Fig 2. Hash generation unit

As the underlying one-way function is EC point multiplication and it is infeasible to find another message string of same x_1 and x_2 values, the above function satisfies all the properties to be chosen as a hash function. Thus the only additional hardware required for hash generation are two simple modular division circuits with polynomial $g_1(x)$ and $g_2(x)$. This results in a large reduction of hardware for the entire secure communication system.

5. HARDWARE STRUCTURE FOR EC POINT MULTIPLICATION

This work concentrates on hardware implementation of encryption system, integrity verification system and key exchange system. The design of complete hardware structure is explained here. The point multiplication operation is realized through point addition and point doubling operations on EC. As it is clear from the formulae for point addition equation (7) and point doubling equation (8), these operations are performed through addition, multiplication and squaring operations on the variables that are elements of the basic field over which the EC is defined. So, the hierarchy of arithmetic for EC point multiplication for a point **P** on an elliptic curve is as shown in Fig 3. Since EC over GF(2^m) are more suitable for hardware implementation, the basic operations are to be done in GF(2^m) for the implementation of point multiplication.

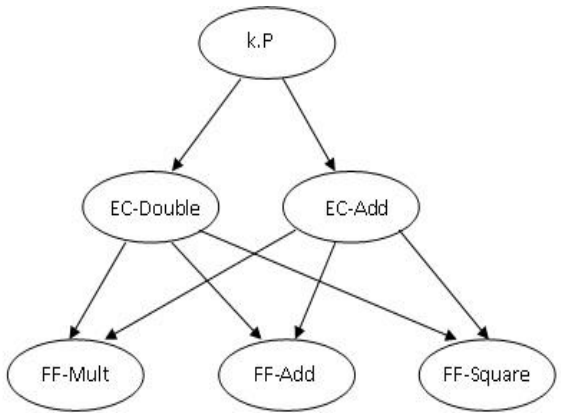

Fig 3.EC arithmetic hierarchy

The finite field addition (FF-Add) and finite field squaring (FF-Square) operations are quite simple. These operations can be done with very few clock cycles. But finite field multiplication (FF-Mult) is very costly in terms of hardware requirement. The number of clock cycles required for its computation depends on the particular architecture of the FF multiplier.

5.1 Hardware Structure for EC Arithmetic

The *EC* point multiplication is computed by repeated point additions and point doubling

Algorithm:

```
Input :An integer $k>0$, Point $P$ on EC.
Output: $Q = k.P$
Step1: Set $k=(k_l.........k_1k_0)_2$
Step2: Set $Q \leftarrow P$,
Step3: for $I$ from $l-1$ down to $0$ do
        $Q \leftarrow 2Q$,
        if $k_i = 1$
        $Q \leftarrow P+Q$,
        end if
    end for
Step4: Return$(Q)$
```

The architecture for the EC point multiplication can be obtained by combining the EC addition and EC doubling architectures as given in Fig.4.

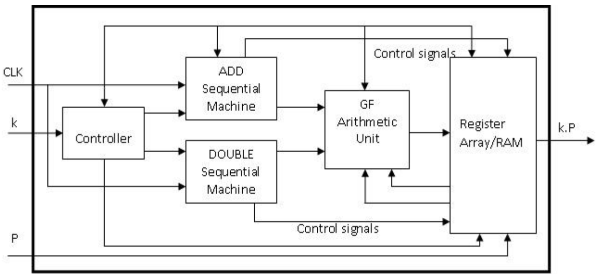

Fig 4. EC point multiplication block

If k_i is '1' then first an EC doubling operation and then an EC addition operation is to done. If k_i is '0' then only an EC doubling operation is to done. The ADD sequential machine does the sequential actions and generates the necessary control signals for the EC addition operation. Similarly the DOUBLE sequential machine does the sequential actions and generates the necessary control signals for the EC doubling operation. The intermediate results are stored in the register/RAM array. Clock division is provided for the GF multiplication as it requires many clock cycles. The EC addition operation and EC doubling operation requires additional steps which requires one or more clock cycles. A controller has been designed to generate various control signals to synchronize operations within the EC point multiplication unit. These operations include loading a new integer input into the point multiplication block, passing control between doubling and addition units, loading the result into output lines etc. The counter also synchronizes the starting and ending of a point multiplication. After the completion of point multiplication in the projective coordinates, the z coordinate of the result is passed into an inversion block, which does the field inversion. Then the inverse of z is multiplied with x and y coordinates of the result to get the point multiplied result in the affine form.

5.2 Hardware Implementation of ECPBSG

The entire working of the hardware can be divided into three different phases 1) key exchange 2) pseudo random bit sequence generation and 3) hash generation. The complete structural block diagram of the implementation of secure communication module is shown in Fig.5. The hardware requires five different clock frequencies which can be generated using clock division circuit. The hardware generates signals for handshaking process i.e. the request and grant signals. Once the initial handshaking is over the process of key exchange is initiated. The hardware randomly selects an integer stored in the memory, loads it to the LFSR. Initial seed point is the point P which is publicly known. EC point multiplication block computes the result according to the algorithm discussed before. The result is output through the port YA. Simultaneously it receives the value YB sent by the other entity through port YB. Now YB forms the seed point for next EC multiplication. The result of point multiplication with YB as the seed point forms the key. The x value of the key is loaded into the LFSR and the key acts as new seed point for EC multiplication. The output from the EC point multiplication block is now given to the trace generation circuit to get the PRBS. This output bit gets xor-ed with the message bits stored in the buffer and the resultant cipher text is obtained. Simultaneously the message bits are loaded into the hash generation unit. Once the entire cipher text is generated the LFSR is loaded with the output of hash generation unit. The x-coordinate value of the result of point multiplication is given out as the hash value of that message block. The control signals to the various blocks are generated by the controller.

5.3 Experimental Results

In this paper, we propose a design for implementation of secure stream cipher based on EC using an efficient multiplier over GF(2^{41}). Our design proposes a Stream Cipher generator, Hash generation unit, and a Key exchange module as a single hardware unit which uses the point multiplication unit on a time sharing basis controlled by a high performance controller unit thereby reducing the hardware complexity. The implementation of proposed EC cypto system was carried out using Xilinx Spartan 6 (XC6SLX45T) FPGA. The proposed architecture uses 7,263 slices and has a maximum frequency of 143MHz as shown in table 1. This hardware efficient design provides an elliptic curve cryptosystem with stream cipher of high throughput rates, integrity verification unit and key exchange as one module. Further, the proposed architecture can easily be implemented on ASICs or FPGAs.

Table 1. Performance results

	Device/Slices	Freq	Remarks
This work	XC6SLX45T, 7363	143 MHz	Time shared multiplier unit for key exchange hash and stream cipher generation

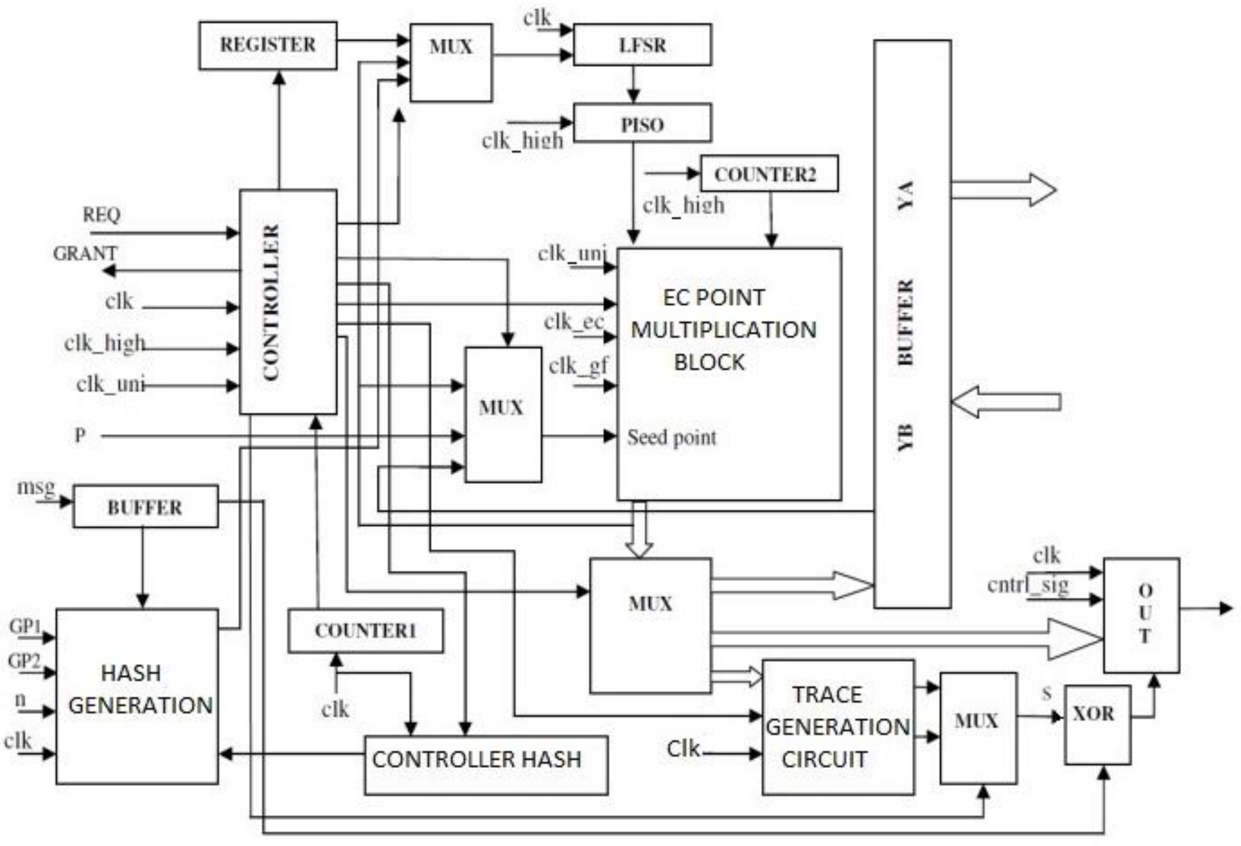

Fig 5. Structural diagram of the Hardware for Secure stream cipher with key exchange and hash generation

6. CONCLUSION

We have designed and implemented an Elliptic curve based message encryption module, key exchange module and hash generation module using verilog on FPGA Spartan 6. Even though the implementations were done over GF(2^{41}), it can be extended to any number of bits. The proposed design is an efficient implementation of EC based encryption system with good security making use of hardware for key-exchange and integrity verification. Since EC based key exchange is a popular option for key exchange in many of the modern communication systems, the proposed design is highly relevant in the implementation of secure communication system of low hardware complexity suitable for hand-held devices.

7. REFERENCES

[1] M. Ernst, B. Henhapl, S. Klupsch, S. Huss, "FPGA based Hardware Acceleration for Elliptic Curve Public Key Cryptosystems", The Journal of Systems and Software, 70 (2004), Elsevier Publishers, pp.299-313.

[2] Rivest, R.L., Shamir, A., Adleman, L.M., 1978. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems, Communications of the ACM.

[3] Miller, V., 1986. Use of elliptic curves in cryptography. In: Williams, H.C. (Ed.), Advances in Cryptology, Proceedings of the CRYPTO' 85, LNCS 218, Springer-Verlag, pp.417–426.

[4] Koblitz, N., 1987. Elliptic curve cryptosystems, Mathematics of Computation 48, pp.203–209.

[5] Deepthi P.P and Sathidevi P.S, "New Stream Ciphers Based on Elliptic Curve Point Multiplication" Computer Communications 32 (2009), Elsevier Publishers, pp.25-33.

[6] Hankerson, Menezes and Vanstone, "Guide to Elliptic Curve Cryptography", Springer-Verlag, 2004.

[7] Arash Reyhani-Masoleh, M. Anwar Hasan, Low Complexity Word-Level Sequential Normal Basis Multipliers. IEEE Transactions on Computers, vol.54 no.2, pp. 98-110, 2005.

[8] C. K. Koc and B. Sunar, "Low-complexity bit-parallel canonical and normal basis multipliers for a class of finite fields," IEEE Transactions on Computer, Vol. 47, 1998, pp. 353-356.

[9] Chang Hoon Kim, Soonhak Kwon, Chun Pyo Hong, "FPGA implementation of high performance elliptic curve cryptographic processor over GF(2^{163}), Journal of Systems Architecture, 54 (2008), Elsevier Publishers, pp.893-900.

[10] IEEE 1363, Standard Specifications for Publickey Cryptography, 2000.

[11] NIST, Recommended elliptic curves for federal government use, May 1999. <http://csrc.nist.gov/encryption>.

[12] Rudolf Lidl and Harald Niederreiter , “Introduction to finite fields and their applications”, Cambridge University Press- 2000.

[13] S. Kwon, K. Gaj, C.H. Kim, C.P. Hong, Efficient linear array for multiplication in GF(2^m) using a normal basis for elliptic curve cryptography, in: CHES 2004, Lecture Notes in Computer Science, vol. 3156, 2004, pp. 76–91.

[14] A. Menezes, “Elliptic Curve Public Key Cryptosystems”, Kluwer academic publishers, 1993.

[15] Henri Cohen and Gerhard Frey, “Handbook of Elliptic and Hyper Elliptic Curve Cryptography”, Chapman & Hall/CRC, 2006.

[16] Neal Koblitz, “A Course in Number Theory and Cryptography” Springer 2005, Second Edition.

[17] A. J. Menezes, “Applications of Finite Fields”, Kluwer Academic Publishers, Boston, Mass, USA, 1993.

[18] Shuhong Gao , Hendrik W. Lenstra, Jr., "Optimal normal bases, Designs, Codes and Cryptography, v.2 n.4, pp.315-323, December 1992.

[19] R.C. Mullin, I.M.Onyszchuk, S.A.Vanstone, R.M.Wilson, "Optimal Normal Bases in GF(p^n), Discrete Applied Mathematics, v.22 n.2, pp.149-161, February 1989.

[20] Arash Reyhani-Masoleh, Efficient Algorithms and Architectures for Field Multiplication Using Gaussian Normal Bases, IEEE Transactions on Computers, vol. 55, no.1, pp.34-47, 2006.

[21] R. Azarderakhsh, A. Reyhani-Masoleh, " A modified low complexity digit-level Gaussian normal basis multiplier", Lecture Notes in Computer Science Volume 6087, 2010, Springer-Verlag Berlin, pp 25-40.

[22] Deepthi P.P., Nithin V.S., Sathidevi P.S., “Implementation and analysis of stream ciphers based on the elliptic curves” Computers and Electrical Engg 35 (2009), Elsevier Publishers, pp.300-314.

[23] Nithin V.S., Deepthi P.P., Dhanaraj K.J., P.S. Sathidevi, “Stream Ciphers Based on the Elliptic Curves” International Conference on Computational Intelligence & Multimedia Applications ICCIMA 07, Volume IV., pp. 99-104, 13-15 December 2007 at Tamil Nadu.

[24] Dhanaraj K.J, Deepthi P.P and Sathidevi P.S, “FPGA Implementation of a Pseudo Random Bit Sequence Generator Based on Elliptic Curves”, International Congress for Global Science and Technology (ICGST) – Programmable Devices, Circuits and Systems (PDCS) Journal, Volume 7, Issue 1, pp. 23-31, May 2007.

[25] G. Gong, T.A. Berson, and D.R. Stinson, “Elliptic curve pseudorandom sequence generators”, Technical Report, University of Waterloo, December 1998.

[26] P.H.T.Beelen and J.M.Doumen, "Pseudorandom sequences from Elliptic Curves", Finite Fields with Applications to Coding Theory, Cryptography and Related Areas", Springer-Verlag 2002, pp.37-52.

[27] Menezes, P Van Oorschot and S. Vanstone, “Handbook of Applied Cryptography”**,** Second Edition, CRC Press 1996.