Implementation of Architecture and Algorithm for Point Multiplication Based On Koblitz Curve

R.Divya¹, S.Kalpana²
II-M.E(CS)¹, AP/ECE², Dhanalakshmi Srinivasan Engineering College, Perambalur.
Dhanalakshmi Srinivasan Engineering College, Perambalur.

ABSTRACT
In recent years, elliptic curve cryptography (ECC) has gained widespread exposure and acceptance, and has already been included in many security standards. Engineering of ECC is a complex, interdisciplinary research field encompassing such fields as mathematics, computer science, and electrical engineering. In this paper, we survey ECC implementation issues as a prominent case study for the relatively new discipline of cryptographic engineering. In particular, we show that the requirements of efficiency and security considered at the implementation stage affect not only mere low-level, technological aspects but also, significantly, higher level choices, ranging from finite field arithmetic up to curve mathematics and protocols.

Index Terms- Cryptography, ECC, Efficiency Protocols, Security.

I INTRODUCTION
Nowadays, Radio Frequency Identification (RFID) technology surrounds us in several forms. RFID typically refers to wireless single chip, passive or active transponders operating at frequencies from 120 kHz (low-frequency) to 10 GHz (microwave, semi-active or active tags). They are used for several applications such as, military application, Internet services, Telephone Calls etc. In symmetric key method, key can be stored in one node only so the hackers can easily hack the network. Public key method is advantages as compare to symmetric key method. In several applications Elliptic Curve Cryptography (ECC) is considered. It is one of the public key cryptography schemes. The main advantage of ECC is that it offers similar security levels as other approaches but it employs smaller key sizes. The security of ECC relies on the difficulty of solving the elliptic curve discrete logarithm problem (ECDLP). The main arithmetic computation of ECC is an operation denoted as point multiplication. The performance of point multiplication is determined by finite field arithmetic computations. In this paper, We consider Koblitz Curve method. We represent the field elements using Gaussian normal basis (GNB). This basis provide cheap multiplications, squaring and low area complexity.

II KOBILITZ CURVE
A. Gaussian normal basis: A normal basis can be constructed by finding the normal element GF(2^m). Gaussian Normal Basis is attractive because it provides cheap multiplication, squaring and low area complexity. For this operation multipliers are used. Various multipliers are present such as, Bit-Level parallel-in parallel-out(BL-PIPO), Bit-Level serial-in parallel-out(BL-SIPO) etc. in BL-PIPO multiplier in which all the coordinates of both input operands should be present throughout the multiplication operation. The BL-SIPO normal basis multipliers are advantageous for applications where one of the input operands is available in a bit-serial fashion.

Let R=(r_0, r_1, ……r_m), S= (s_0, s_1, ……s_m) be two field elements in GF(2^m). T will be the product of R and S (i.e) T=RS. In this R input is in serial manner and S input is in parallel manner. The structure of BL-SIPO GNB multiplier can be easily modified for a BL-PIPO architecture. In this case, both operands R and S should be available throughout the multiplication process. In ECC method PIPO can be used for that three registers are used to store the values. BG has a serial-input parallel-output architecture and similar to the other architectures,
for real applications, one needs to have both input operands stored in a register (shift register) during the multiplication process and, hence, it requires three registers as well.

III POINT MULTIPLICATION

Let $E(GF(2^m))$ be the group of points on an elliptic curve over a binary extension field. The point $(u,v)$ that satisfy the elliptic curve equation together with a special point called the point at infinity. The group operation $(u_3,v_3)=(u_1,v_1)+(u_2,v_2)$ is called point addition. If $u_1=u_2$ and $v_1=v_2$ point addition is called point doubling.

In previous, elliptic curve method is used. An elliptic curve is the set of points that satisfy a specific mathematical equation. The equation for an elliptic curve looks something like this:

$$y^2 = x^3 + ax + b$$  \hspace{1cm} (1)

And the elliptic curve representation is as follows as,

![Elliptic curve](image)

The point addition on Koblitz curves can be performed in different coordinates including affine, projective, and mixed coordinates. Note that it is possible to define point operations on Koblitz curves with more than three different coordinates (e.g., affine, standard projective, Jacobian projective, Lopez-Dahab, and different mixes of them) but we consider only three alternatives in this paper. In affine coordinates, point addition $(u_3,v_3)=(u_1,v_1)+(u,v)$ is computed as follows,

$$\lambda = \frac{v_1 + v}{1 + u}$$  \hspace{1cm} (2)

$$u_3 = \lambda^2 + u_1 + u + a$$  \hspace{1cm} (3)

$$v_3 = \lambda(u_3 + u) + u_3 + v$$  \hspace{1cm} (4)

In affine cost inversion(I), multiplication (M), squaring(S), addition(A) values are used.

Inversion can be efficiently done by using Fermat’s Little Theorem.

IV COORDINATE SELECTION

For our resource-constrained targets, we focus on minimizing the area as much as possible. Projective coordinates, where a point is represented with three coordinates $(u,v,w)$, are commonly used for improving the speed of point multiplication because they allow trading expensive inversions to cheaper multiplications. On the other hand, traditional affine coordinates, where a point is represented with two coordinates $(u,v)$, require simpler control structure and fewer registers to store the points and temporary variables and, as a result, lead to simpler and smaller (but, of course, slower) implementations. Consequently, a point addition (or subtraction) requires $12M$ in affine coordinates and $8 M$ in mixed coordinates. We conclude that this reduction of less than one-third in the expected latency is not significant enough to support using mixed coordinates in resource-constrained implementations, because it would come at the expense of increased number of registers and more complex control structure. Hence, we use affine coordinates in our implementation. Addition in affine coordinates is spent in computing the inversion. Similarly as in many other hardware implementations, we choose to compute inversions via exponentiations based on Fermat’s Little Theorem because it leads to a simpler and smaller implementation compared to Extended Euclidean Algorithm when a multiplier and a squarer are already available. The IT scheme requires $162$ squaring and $9$ multiplication in $GF(2^m)$.

The recently introduced Dimitrov-Jarvinen (DJ) algorithm requires the same amount of multiplications and squaring but allows using fewer registers in the implementation compared to the IT scheme. There are also several other recent proposals that improve inversion computations. They focus mainly on fast increases area requirements and makes them impractical for extremely constrained applications. DJ algorithm computes an inversion in binary field with the smallest possible multiplication, squaring and temporary variables. In DJ algorithm two registers
are used whereas in IT algorithm three registers are used.

V. FIELD OPERATION
The main field operations are as follows as, addition, squaring, multiplication, certain consecutive operation. A. Addition: T=R+S. First R is loaded to Z by selecting R with S_A and setting S_1=1 and S_2=0. Second S is selected with S_A and the addition is computed by setting S_1=1 and S_2=2 which results in Z=R+S. Finally Z is stored into T. B. Squaring: R>>1. First, R is loaded to Z by selecting R with S_A and setting S_1=1 and S_2=0. Second the first operand of the adder is set to zero by selecting S_1=2 and the second operand is selected by selecting S_2=3. Now Z includes 0+Z^2=R^2. Finally Z is stored into T. C. Certain consecutive operations: When already includes the operand for the next operation or the result of an operation is not needed after the next operation, loads and saves from and to the registers can be avoided, respectively. It is also possible to combine two operations into a single operation in certain cases. And the comparison of affine coordinates and mixed coordinates are as follows as. In previous, it was estimated that mixed coordinates would give an improvement of about one-third in latency with the expense of increased area. In order to shed more light on this issue, we derived an algorithm for point addition using mixed coordinate. Point multiplication would take, on average, approximately 76,100 clock cycles using mixed coordinates which is a 29% improvement over the variant with affine coordinates. The area improvement is significant despite the fact that the algorithm utilizes similar resource sharing that was used for affine coordinates the number of registers increases with two 163-bit registers which is a 33% increase.

VI. IMPLEMENTATION
This process can be implemented by using Field programmable gate array. And it may also implemented by using Application specific integrated circuit. FPGAs especially find applications in any area or algorithm that can make use of the massive parallelism offered by their architecture. One such area is code breaking, in particular brute-force attack, of cryptographic algorithms. FPGAs are increasingly used in conventional high performance computing applications where computational kernels such as FFT or Convolution are performed on the FPGA instead of a microprocessor.

Fig 2 Initial Output
In Fig 2 initial output diagram is mentioned. In that S(X) and S(Y) are two input values. K be the key value. After starting the process values of S(X) and S(Y) are changed. In that DONE can be used to represent the end of encryption process. After encryption process the values of input cannot be changed. And the final output can also be represented in hexadecimal format. The final output can be represented in Fig 3. In this 163 bit key values are used for this process. And the input also 163 bit values.

Fig 3 Final output
During the encryption process value of inputs can be changed and the output can be represented in Fig 4.

![Variation of inputs](image)

**Fig 4 Variation of inputs**

**VII CONCLUSION**

In this, we have proposed an efficient implementation of point multiplication on Koblitz curves for extremely con-strained applications such as RFIDs and sensor networks. For that we are using GNB multipliers, it provides cheap multiplication and low area complexity. Fewer register can be used for the process. Here we conclude that the koblitz curve is perfect for the secure application. In this GNB multiplier is used. In future multiplier can be changed and the level of security can be increased. And also reduce the resource requirements in future. Elliptic curve cryptosystems unremarkably need standard number arithmetic additionally to elliptic curve operations. whereas this demand for 2 varieties of arithmetic (binary field and standard integer) creates challenges for all systems mistreatment binary curves in forced environments, the challenges will become bigger for Koblitz curves as a result of conversions between integers and radic representations is also required too. Hence we conclude that the koblitz curve is perfect for secured applications and also for RFID. And the output can be implemented in Field programmable gate array.

**REFERENCES**


