An RNS Implementation of the Elliptic Curve Cryptography for IoT Security

Jai Gopal Pandey *†, Chhavi Mitharwal*, Abhijit Karmakar*†
*CSIR-Central Electronics Engineering Research Institute (CEERI), Pilani, India-333031
†Academy of Scientific & Innovative Research (AcSIR), CSIR-CEERI, Pilani, India-333031
{jai, abhijit}@ceeri.res.in
{chhavimitharwal}@gmail.com

***Abstract*—Public key cryptography plays a vital role in many information and communication systems for secure data transaction, authentication, identification, digital signature, and key management purpose. Elliptic curve cryptography (ECC) is widely used public key cryptographic algorithm. In this paper, we propose a hardware-software codesign implementation of ECC cipher. The algorithm has been modelled in C language. The compute-intensive components have been identified for their efficient hardware implementations. In the implementation, residue number system (RNS) with projective coordinates have been utilized for performing the required arithmetic operations. To manage the hardware-software codeign in an integrated fashion Xilinx platform studio tool and Virtex-5 xc5vfx70t device based platform has been utilized. An application of the implementation has been demonstrated for encryption of text and its respective decryption over prime fields. The design is useful for providing an adequate level of security for IoTs.**

***Index Terms*—Elliptic curve cryptography (ECC), public-key cryptography, Hardware-software codesign, residue number system (RNS), IoT Security.**

I. Introduction

Cyber physical system (CPS) as a collection of Internet of things (IoTs) provide an excellent platform for increasingly connected physical world. This ecosystem enables an integration of compute, network and physical things that work independently to provide computation, communication, information sharing, and control [1]. It brought advances in personalized health care, traffic flow management, electric power generation and many more. A number of cutting-edge artificial intelligence techniques such as , deep learning, machine learning, cognitive computing, etc. have been developed in order to realize the complete potential of CPS [2].

The rapid change in the internet-enabled technologies is said to be the next generation of the internet. The Internet is slowly becoming the active target for the hackers, in which billion of things have been interconnected and are continuing to be connected [3]. One of the significant obstacles in the efficient deployment of CPS-enabled devices is data security, which can be for infrastructure, communication network, applications and general-purpose systems [3]. The basic principle of secure communications in CPS include authentication, availability, privacy, integrity, confidentiality and non-repudiation. Here, cryptography plays a vital role [4].

Recently elliptic curve cryptography (ECC) has evolved as a potential candidate for providing security in the CPS. The main advantage of ECC cipher over the existing public-key ciphers is that it offers equal security for a smaller key size that results in reduction of processing overhead [5]. Smaller key leads to more compact implementation for a given security and provides fast computation rate, power, memory and bandwidth efficient. To implement the ECC cipher, a hardware-software codesign design approach provides an efficient solution. Here, we get the advantages of both; flexibility offered by software and performance by realizing time-consuming arithmetic component in hardware [6], [7].

In this paper an efficient implementation of the ECC algorithm with hardware-software codesign approach is proposed. This compact implementation provides an adequate level of data security for IoTs. To perform modular arithmetic operations of ECC, projective coordinates are utilized. Along with it, the concept of residue number system (RNS) is used for implementing modular arithmetic components like, point addition, doubling and multiplications. A modelling of the ECC has been performed in C language. The C code has been profiled for obtaining the compute-intensive functions of the algorithm. The identified functions have been implemented in VHDL language. To perform an integrated hardware-software codesign, Xilinx platform studio tool with its ML-507 platform have been utilized. The platform provides a *PowerPC* as an hard processor in the FPGA device. The hardware components of the algorithm have been implemented in the FPGA fabric.

Rest of this paper is organized as follows: Section II is used to discuss some of the the related work. In Section III, an overview of the elliptic curve cryptography is given. A method of the implementation and its overall architecture is proposed in Section IV. Section V is used to provide experimental results along with a comparison with an existing architecture. Finally, conclusions are drawn in Section VI.

II. Related Work

In a survey of lightweight cryptography implementations, software and hardware implementations of symmetric and asymmetric ciphers have been compared [8]. In [9], emphasis has been given to approaches for scalar multiplication over elliptic curves. Implementation of a RNS version of F_p elliptic curve point multiplier has been done in [10]. An implementation of elliptic curve point multiplication over $GF(p)$ has been

provided in [11]. Here, the architecture for *ECPM* over $GF(p)$ based on RNS has been presented.

An implementation of an elliptic curve point multiplication using digit-serial binary field operations has been done in [12]. Selected RNS bases for modular multiplication have been discussed in [13]. Research challenges in next-generation residue number system architectures have been emphasized in [14]. Related to a secure and efficient RNS software implementation for elliptic curve cryptography, a design has been provided in [15]. Implementation of text encryption using Elliptic curve cryptography is discussed in [16]. This technique avoids the costly operation of mapping and the urge to share the common lookup table between the sender and receiver. In Table I, a set of algorithms from NIST SP800-57 have been compared by showing comparable key sizes in terms of computational effort for cryptanalysis. It can be seen that the key size required for ECC is comparably shorter, which is valuable as it provides computational advantage for using ECC with a shorter key length than a comparably secure RSA. In next section, details of the ECC cipher has been described.

III. Elliptic Curve and the Elliptic Curve Cryptography

In cryptography, the variables and coefficients present in the elliptic curve equation are bound to elements in a finite field, which result in satisfying the axioms of the Abelian group [4]. Cubic equation for elliptic curves take the form which is known as *Weiertrass* equation and it is expressed as,

$$y^2 + axy + by = x^3 + cx^2 + dx + e \tag{1}$$

Here a, b, c, d and e are the real numbers and x and y take on any value in the real numbers. Here we have focused on prime curve defined over F_p, where a cubic equation is used in which the variables and coefficients take values in a set of integers from 0 to $(p-1)$ and calculations are performed over modulo p. The proposed work is based on elliptic curve that is defined over F_P. The prime curves are best suitable for software applications due to the fact that in prime curve there is no requirement for extended bidding operations [4].

A. Elliptic Curve Over F_P

The expression (1) can be modified in a form where coefficients and variables are limited to F_P as given below,

$$y^2 \bmod p = \left(x^3 + ax + b\right) \bmod p \tag{2}$$

The set $E_P(a,b)$ consist of all pair of integers that satisfy (2) along with a point at infinity O. Here the coefficients a and b and the variables x and y are all the elements of F_P. A finite abelian group is defined based on the set $E_P(a,b)$ given that $(x^3 + ax + b) \bmod p$ has no repeated factors which is equivalent to the condition $\left(4a^3 + 27b^2\right) \bmod p \neq 0 \bmod p$. The rules for addition over $E_P(a,b)$ are defined in [4].

TABLE I
Comparable Key Sizes in Terms of Computational Effort for Cryptanalysis (NIST SP-800-57)

Symmetric Key Algorithm	Digital Signature Algorithm	RSA (size of n in bits)	ECC (modulus size in bits)
80	L=1024 N=160	1024	160-223
112	L=1024 N=160	2048	224-255
128	L=1024 N=160	3072	256-383
192	L=1024 N=160	7680	384-511
256	L=1024 N=160	15360	512

B. ECC Cryptography

As shown in Fig. 1, in order to form a cryptographic system using elliptic curves, there is a need to find factors for product of two prime numbers. Here in $Q = kP$, where $Q, P \in E_P(a,b)$ and $k < p$, it is relatively easy to compute Q given k and P but it is hard to determine k given Q and P. This is the discrete logarithmic problem for elliptic curves.

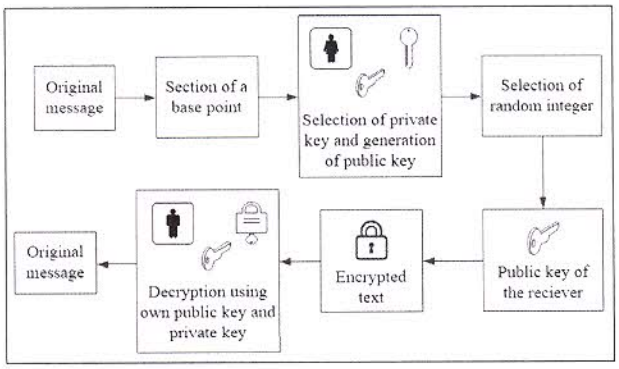

Fig. 1. Elliptic curve encryption/decryption.

Key exchange can be done through elliptic curves by selecting a large number q which is a prime number p and also by selecting elliptic curve parameters, a and b. Next, we pick a base point $G = (x_1, y_1)$ in $E_P(a,b)$ whose order is a very large number n. The sender selects the private key n_A and then generates the public key $P_A = n_A \times G$, which is a point in $E_q(a,b)$. The receiver also does the same. They both generate their own secret keys $k = n_A \times P_B$ and $k = n_B \times P_A$ respectively. These calculations produce the same results as,

$$n_A \times P_B = n_A \times (n_B \times G) = n_B \times (n_A \times G) = n_B \times P_A \tag{3}$$

In order to break this scheme, the attacker should be able to compute k given G and kG, which is very difficult [4].

In encryption, first the encoding of the plaintext message is done as (x, y) a point P_m. There is also a requirement of the point G and an elliptic group $E_q(a,b)$. Sender

selects his own private key and then generates a public key. The sender selects a random positive integer k and then produce the ciphertext C_m which consists of the pair of points $C_m = \{kG, P_m + kP_B\}$. The sender here has chosen the receiver's public key. In decryption, receiver multiplies the first point in the pair by his own private key and subtracts the result from the second point as $P_m + kP_B - n_B(kG) = P_m + k(n_B G) - n_B(kG) = P_m$. The security of ECC depends on how difficult it is to determine k if kP and P are given. This difficulty is referred as elliptic curve logarithmic problem [4].

One of the vital components of the ECC cipher is the scalar multiplication. We have here implemented the scalar multiplication using the *binary method*. This method has minimum memory requirements and relatively easy implementation. The pseudo code of binary method method is shown in Fig. 2. In the pseudo code, P signifies the point and k is an l-bit integer such that $k = \sum_{0}^{l-1} k_j 2^j$. The binary method requires $l-1$ point doublings and $W-1$ point additions, where l is the length and W is the hamming weight of the binary expansion of k.

```
intialize P, Q, k, l
Q ← O
for (j ← l-1 down to 0)
Q ← 2 X Q
if (k=1) then
Q ← Q + P
end if
end for
```

Fig. 2. The binary scalar multiplication method.

IV. A Design Approach for Elliptic Curve Cryptography (ECC) Implementation

In this section a method for an efficient implementation of the elliptic curve cryptography (ECC) is described. This method is based on an elliptic curve, where projective coordinate and residue number system are utilized. The details of the approach is described below.

A. Selection of an elliptic curve

To select an elliptic curve, the expression (1) can be modified as,

$$y^2 = x^3 + ax + b \tag{4}$$

Example of this elliptic curve equation is $y^2 = x^3 - x + 1$ and it is shown in Fig. 3. Here the values of a and b are 1 and -1 respectively.

As shown in Fig. 3, there is geometric description of addition on elliptic curve. In order to do the addition of two points P and Q that lie on the curve, a line is drawn through these, which intersects the curve at the third point of intersection that is the mirror image of the points that lie on

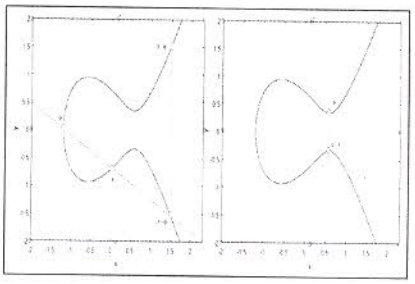

Fig. 3. Arithmetic operations on the selected elliptic curve.

the same curve. The second geometric description shown in Fig. 3. Here, a line interests the curve at infinity when it passes through the point and the negative of the same point [4]. Now we explain about the coordinate systems used in elliptic curve, followed by residue number system used in our methodology.

B. Coordinate systems in elliptic curve

An elliptic curve point P can be represented through numerous coordinate systems. The prominent coordinate systems that can be used here are affine coordinate, projective coordinate, mixed coordinate, *Jacobian* coordinate or modified *Jacobian* coordinate[17]. The objective here is here to use an efficient coordinate system for encryption and decryption so that the elliptic curve *Diffie-Hellman* (ECDH) key exchange protocol can be executed in the shortest time.

Here affine and projective coordinates are used. By this, the point addition and point doubling can be performed easily. The cost of point addition is $1I + 3M$ and for point doubling is $1I + 4M$, where I and M refer to number of inversions and multiplications respectively. Point addition and point doubling require modular inversion which is very expensive that can be avoided using projective coordinate system also known as conventional projective coordinates.

C. Residue Number System

Residue number systems (RNS) are considered because of their inherited parallelism, modularity, fault tolerance and localized carry propagation properties. Residue number systems are based on congruence relation. If q and r are the quotient and remainder respectively, when a is divided by m i.e $a = q.m + r$ then we have $a \equiv r(\mathrm{mod} m)$. Number r is called residue of a with respect to m. The set of m smallest possible values, $(0, 1, 2, ..., m-1)$, that the residue may consider is called the set of least positive residues modulo m [18].

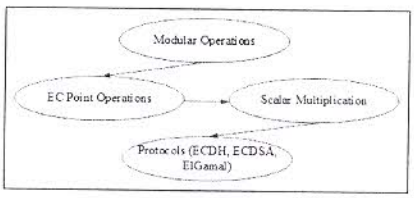

Fig. 4. The elliptic curve cryptography (ECC) operational flow.