PIECC: Point Inversion algorithm for Elliptic Curve Cryptology to Secure IoT Data Communication

Padmashree M.G.
University Visvesvaraya College of Engineering, Bangalore University, Bengaluru, India

Arunalatha J.S.
University Visvesvaraya College of Engineering, Bangalore University, Bengaluru, India

Venugopal K.R., *Fellow, IEEE*
Bangalore University, Bengaluru, India

ABSTRACT

In the Internet of Things (IoT), the internet-connected objects send the Collected data and act on the received data. Encryption controls a large number of structured and unstructured data protection during transmission. Inadequate memory and processing capacity of IoT devices demand Elliptic Curve Cryptography (ECC) for simple, secure functionalities. Scalar Multiplication frequently uses Modular Inversions that impact significantly on ECC-based applications with low resource usage with the enhancement of reliable IoT System availability. The Point Inversion algorithm for Elliptic Curve Cryptology (PIECC) enhances security and reduces the Computation time of Modular Point Inversion of Elliptic Curve using High-Speed Split Multiplication and Squaring. The use of limited intermediate registers for Cryptographic functions optimizes the Storage. The proposed algorithm reduces the Computation Time of the Cryptographic operations in terms of Clock cycles using chain Fermat-based Inversion compared with High-Speed multiplication and Product Scanning algorithms with lower Space Complexity.

Keywords

Elliptic Curve Cryptography, Internet of Things, Modular Inversion, Montgomery Curve, Security of data

1. INTRODUCTION

The varying worldwide development in Modern Technology shows the converging of computation and communication. The distributed Smart devices with a remote connection substitute the Personal Computers with a wired network in almost all the sectors. It necessitates information protection and security measures. Data security and confidentiality are currently the requisite for Banking applications (mobile, SMS, UPI, *etc.,*) on Phones, wearable Healthcare devices, work-from-home, *etc*. Internet of Things (IoT) drives the concept of bringing the world together via universal connectivity. The fundamental requirement of IoT is to secure information and data communication providing robust availability with optimized resource usage.

The embedded systems are highly domain-specific; The domains expand for such systems. The purpose of the washing machine is to inlet and outlet water at a controlled time using a programmed microcontroller. The fully automatic washing machines provide the option of fuzzy, pre-set hot water washing and other fabric-dependent programming features. The cell phone acts as a router and the Television screen as a Smart screen using Hotspot and WifiDirect facility. Every device *viz.,* laptop, printer, refrigerator provides functionality and maintenance status. The internet of Things provides connectivity of the underlying devices with least or without human intervention. The connected embedded devices transfer information and need network security in a resource-constrained environment. The small key size with a high level of protection causes Elliptic Curve Cryptography [1] effective in an IoT scenario with reduced Storage, and Time overhead minimizes the power consumption. It increases the life span of IoT devices.

The primary component of IoT is Device-to-Device communication. Wireless sensor networks are application-specific. The sensor nodes or motes design in compliance with the application. IoT and Device-to-Device communication use existing motes. The functionality of sensors integrates into the chipset of the specific device leads to a device-specific implementation.

The Sensor nodes of the IoT and WSN nodes vary in IP connectivity. Cooja simulator implements an IP stack into sensor nodes *e.g.,* TELOSB, Tmote, Micaz, *etc.,* Sensor nodes in IoT are IP-enabled with a 6lowPAN IP address allowing them to interact with remote Edge-to-Edge motes. The characteristics of the sensor node or mote are non-replaced battery, low data transfer capacity, restricted computational and operational efficiency. IoT environment requires the highest performance of the nodes consuming the least energy.

Motivation: Finite Field Arithmetic is extensively used in numerous fields *viz.,* Combinatorics, Coding, String Theory, and Cryptology, Logic Gate Theory, *etc.,* A wide range of Public-Key Crypto Applications implemented are over a high-order Finite Fields [2]; accordingly, multiplying and dividing operations influence the processing time of encrypting and decrypting functionality. Thus, designing and developing a faster approach to carry out such operations is essential. High Speed Split Multiplier (HSSM) for Elliptic Curve Cryptography [3] uses Split Multiplier with confined registers to carry out Cryptographic operations in IoT. The faster Inversion algorithm accelerates the implementation of the Scalar Multiplication. The reduction in computation cost and storage complies with Micaz mote of 4KB RAM. The Point Inversion algorithm for Elliptic Curve Cryptology (PIECC) to secure Data Communication in IoT reduces the Computation Time. It thereby decreases the processing time of the Encryption and Decryption in the Public Key Infrastructure of IoT.

Laporta *et al.,* [4] described a Modular Multiple Word Inversion enhancing Stein Greatest Common Divisor with De Bruijn Contiguous Subsequence and Montgomery arithmetic. Hyper Thread [5] and Reverse Product Scanning [6], [7]enhanced the existing systems. Many researchers worked on Binary Edward, Twisted Edward, and Montgomery curve [8], [9], [10] over modular Non-CoPrime [11],Gaussian Integer [12] and Prime Field [13].

Contributions: The key contributions of this work are to enhance:

i) security by reducing the Computation time of Modular Point Inversion of Elliptic Curve using High-Speed Split Multiplication and Squaring.
ii) performance by optimizing the Storage using limited intermediate registers of motes for Cryptographic functions.

Organization: The paper is structured as follows: Section 2 abstracts the Related Works of the modular Inversions for Elliptic Curve Cryptography in an IoT environment. Section 3 describes the Background Work, and Section 4 presents the Proposed Point Inversion algorithm for Elliptic Curve Cryptology (PIECC) for secure Data Communication in IoT Algorithm. The Performance Analysis of the proposed system is in Section 5. Section 6 contains the Conclusions.

2. RELATED WORKS

Fermat's little theorem (FLT) over Prime Fields reduces Squaring and Multiplication in modular Inversion. Gallin *et al.,* [14] presented a Pipelined Modular Multipliers for Hyper Elliptic Curve Cryptography of Field Programmable Gate Arrays with Digital Signal Processor blocks. The pipelines are filled using Dynamic Threads. The modified 128 bits algorithm computes parallel independent intermediate results and uses Block RAMs to produce smaller and faster circuits compared to the Multiplier with Threads of [5]. The performance is distinct and non-generalized, excluding the non-square quadrate blocks of the Digital Signal Processor, focusing on the single word parameters applicable for Digital Signal Processor functions of the Field Programmable Gate Array.

Ding *et al.,* [15] presented a Two-stage pipelined Montgomery Modular Multiplication using a Two-stage Karatsuba multiplication [9] which is extended to Three and Four stage multipliers for unbounded positive operands [15]. The partitioning of the operands into small chunks reduces the core multiplication by occupying all the pipeline stages. The cropped multipliers and registers reduce the Look-Up Table (LUT) cost. Feng *et al.,* [16] proposed a multi-path Number Theoretic Fourier Transformation algorithm using Quantum Computational cryptanalysis train and test over Polynomial Rings. The computational complexity is directly proportional to the degree and inversely proportional to the paths. The use of the modified Caching technique and the Round Constant speeds up the product computation.

Leelavathi *et al.,* [17] implemented and processed ECC with 8 Montgomery Curve Point Product obtained using Urdhva Triyambhagyam Vedic Multiplication technique. The 194 bits implementation overflows the Spartan-3E resource limitations. Li *et al.,* [18] presented a framework for a Koblitz Elliptic Curve Point Multiplier (ECPM) with Heterogeneous Scalar conversion. The parallel and pipelined executions with a three-level channeled Finite Field accumulator of the functional unit increase the performance of the Addition operation. The heterogeneous delayed reduction and dual τ-adic Nonadjacent Form method reduce the resource usage. Inversion is performed using continuous Addition operations.

Wenger *et al.,* [19] implemented Four Elliptic Curve multiplier with an Accumulator and performed Elliptic Curve Multiplication of 4 Bytes accessed in 2 sets of a Word using eight registers accelerating the Functional Unit. Abdulrahman *et al.,* [20] presented a Radix-Eight Multiplier method to assess the EC multiplier by pre-computing results of doubling and addition operations for three bits of the scalar in parallel without using the Look-Up Table. The Prime Extended Twisted Edward Curve with Eight multiplier implementation performs better compared to the Montgomery with resistance to Simple Power Analysis (SPA) attack. The scheme outperforms in a multiple processor parallel execution environment.

Salarifard *et al.,* [21] proposed architecture for an Elliptic Curve Point Multiplier using a Fixed base Comb to decrease the quantity of Addition Operations performed and the time complexity. The Static base algorithm requires the Prior Multiplication operation to complete, introducing an extra operating cost. A Two-level pipelined Karatsuba Ofman Point Multiplier with overlapped operations decreases the complexity and latency, increasing the throughput. Ali *et al.,* [22] designed a 521-bit Modular Multiplier using Matrices and Vectors applying dual and triple partitions over a Four Byte result for the static and varied base operands in two stages. The number of Arithmetic operations and the time complexity are reduced by avoiding the overflow compromising the efficiency of the algorithm.

Roy *et al.,* [10] presented an improved Montgomery Elliptic Curve Point Multiplier [23] that incorporates the compressed circuitry allowing two sets of multiplier functionalities performed simultaneously using Carry links, LUTs, and the scheduler. Amiet *et al.,* [24] designed an ECPM processing framework for the operands to the maximum of 0.5Kbits encompassing a Finite Field bounded by 1Kbits. The implementation uses DSP blocks of FPGA, and parallelization of Addition and Doubling increases the efficiency. The Modular Arithmetic functionalities execute in parallel. Fermat's method is applied in the Modular Inversion operation. The Elliptic Curves of Parameter Size more than 256bits are not supported.

Rashidi *et al.,* [25] employed two-Field parallel Multiplier Selim *et al.,* [26] NIST Montgomery Modular Multiplication Al-Asli *et al.,* [27] resists Internal Attack. Loi *et al.,* [28] designed and implemented a NIST ECC processing framework using DSP blocks and parallel execution to reduce the time complexity and memory usage. The multiplier and the Inverter use the same functional block without parallelization.

Xu *et al.,* [29] constructed a Modular Inversion over 192 bit NIST and Chinese Elliptic Curves resistive to SPA and Heuristic Lattice-based Decoding Attacks. The Montgomery Point reduction decreased the number of multiplication operations performed to minimize the Time complexity.

Choi *et al.,* [30] Hardware-based NIST ECC processor. A repeated Modular Prime number reduction over multiple intermediate products of size greater than the operand size and half the size of Final Result used. The Computation cost reduced is directly proportional to the size of the Intermediate results using a 0.5Kbits register. Gu *et al.,* [11] generalized the McLaughlin Multiplier of Residue Number System with unpaired Secondary Prime applying

the Chinese Remainder Theorem. The Montgomery representation allows the same coefficients used for varied sizes.
Researchers have designed Scalar Multiplication with Inversion algorithms for specific Elliptic Curves providing mathematical models and validations for various motes in an IoT scenario.

3. BACKGROUND WORK

Researchers have designed Scalar Multiplication with Inversion algorithms for specific Elliptic Curves providing mathematical models and validations for various motes in an IoT scenario. Liu *et al.,* [7] implemented MoTE Curves to secure IoT device communication. The Curve models resist the Power Analysis attack. Table 1 gives the comparison of related background works on the Elliptic Curve Cryptographic operations.

3.1 Elliptic Curve Cryptography

The Elliptic Curve Point representation of the data provides tamper-resistance data transfer. The secure ECC is assured by the Discrete Logarithmic Property (DLP). An Elliptic Curve is a smooth, Planar, Projective, Algebraic Curve with a single genus, characterized by two Curve Coefficients that belong to a Finite Field. An Elliptic Curve satisfies the properties of an Abelian Group over a Multiplication with a Commutator $\mathcal{O}$, the Point at Infinity in the Projective Plane [3]. Cryptographic Applications require adequate security, which is provided by the Elliptic Curves. If the selected Elliptic Curve satisfy the characteristics reported in *SafeCurve* [31], then the security is assured. Montgomery Elliptic Curves E_m represented by the equation (1).

$$E_m : B.Y^2 = X^3 + A.X^2 + X, \tag{1}$$

where A, B curve coefficients belong to a Finite Field Prime F_p with $(A^2 - 4).B$ resulting in a non-zero number. Twisted Edwards Curve E_t/F_p is given in equation (2).

$$E_t : A.X^2 + Y^2 = 1 + D.X^2.Y^2 \tag{2}$$

where $A,\ D \in F_p$ and $A.D.(A - D)$ a non-zero integer.
The Additive Policy of Elliptic Curve [3] states that sum of any two Points P_i, P_j on the E_t curve results in a Curve Point subject to $\sqrt{A}$ is a positive Integer with $A \in F_p$ and $\sqrt{D}$ is a real number *i.e.,* $\forall_{P_i,P_j \in E}\{P_j + P_j \in E : if\ \sqrt{A} \in Z^+\ and\ \sqrt{D} \notin Z^+\}$.
The properties of the Montgomery and Twisted Edward Curves allow the Weierstrass Elliptic Curve Point representation.

3.2 Fermat-Euler theorem

The Fermat-Euler theorem is given in the equation (3), (4) is used to Inverse the Prime number exponentiation representation of an Elliptic Curve.

$$\forall_{a \in N^+} \exists_b \{|a = \rho * b\} \tag{3}$$

where ρ is Prime, N^+ is the set of Positive Integer Numbers and $a, b, \rho \in N^+$; *i.e.,*

$$\forall_{a \in N^+} \left\{ \begin{matrix} a^\rho \equiv a\ mod\ \rho & \iff & a\ mod\ \rho \neq 0 \\ a^{\rho-1} \equiv 1\ mod\ \rho & \iff & (a-1)\ mod\ \rho = 0 \end{matrix} \right\} \tag{4}$$

3.3 Problem Statement

The Encryption/Decryption technique provides secure information exchange in a stable framework. The resource-restricted IoT devices demand the utilization of Elliptic Curve Cryptographic (ECC) techniques. The Scalar Multiplication significantly affects the ECC usage in minimizing the Computational Complexity and the Storage overhead of Modular Inversions, as the ECC operations involve the Fundamental Modular Arithmetic. Thus, upgrading an ECC enhances the Security and Privacy of the IoT Application.
Given a collection of IoT devices of specified configuration, to design a Point Inversion algorithm for Elliptic Curve Cryptology to secure IoT Data Communication, the objectives are as follows:

i) Enhance the security with decreased computation time using the High-Speed Split Multiplication based Squaring algorithm for Modular Point Inversion of Elliptic Curve Cryptography.

ii) Optimize the data and code Storage, improving the performance of IoT devices using limited intermediate registers to carry out Cryptographic functions.

3.4 Assumptions

The heterogeneous IoT devices or the Sensor nodes:

i) Handle unexpected failure.

ii) Report on session failure.

iii) Portable within a network range of varied topologies.

iv) Scalable, independent, and re-programmable.

4. PROPOSED POINT INVERSION ALGORITHM FOR ELLIPTIC CURVE CRYPTOLOGY (PIECC) FOR SECURE DATA COMMUNICATION IN IOT

The Cryptographic Elliptic Curve Point Encrypt and Decrypt functions perform Point Addition, Doubling, Inversion, *etc.,*. The Prime numbers chosen depend on the Curve properties to satisfy. There is no division arithmetic function for the prime group, a division is performed by finding Inversion of the denominator and then multiplying with the numerator. Table 2 provides the definition of the notations used in the Section.

4.1 Selection of Curves

A Prime Field number is represented as $p = 2^\alpha - \beta$ [3] where, $\beta \leq 2^\omega$ (ω, mote Word-length), $p \cong 1mod4$ and $\beta \cong 5mod8$.
Let the equation (5) given below represent a 159 bit Montgomery Curve

$$E_m159 : -319156.y^2 = x^3 + 3191566.x^2 + x \tag{5}$$

and Twisted Edwards curve given in the equation (6).

$$E_t159 : -x^2 + y^2 = 1 + d.x^2.y^2, \tag{6}$$

where d = 8372259163916347487045608834894170521 976562663492.

4.2 Modular Squaring

The Modular Squaring function performs multiplication of two same operands based on HSSM.

Table 1. Comparison of Recent Related works

Article	Approaches	Advantage	Disadvantage
Gallin *et al.*, [14] (2019)	Pipelined Modular Multipliers Dynamic Threads.	computes parallel independent intermediate results; uses Block RAMs	Distinct, non-generalized; exclude non-square quadrate blocks; focus on the single word parameters
Leelavathi *et al.*, [17] (2019)	ECC using Urdhva Triyambhagyam Vedic Multiplication	8 Montgomery Curve Point Product	194 bits implementation overflows the Spartan-3E resource limitations.
Amiet *et al.*, [24] (2016)	Finite Field bounded by 1Kbits	ECPM for the operands to the maximum of 0.5Kbits; DSP blocks and parallelization of Addition and Doubling increases the efficiency.	Parameter Size more than 256bits not supported.
Loi *et al.*, [28] (2015)	NIST ECC processing framework using DSP blocks	parallel execution reduce the time complexity and memory usage	Multiplier, and Inverter use the same functional block without parallelization.

Table 2. Table of Notations

Symbol	Definition
p	Prime Field number
ω	mote Word-length
E_mX	X bit Montgomery Curve
E_tX	X bit Twisted Edwards curve
E_m	Montgomery Curve
E_t	Twisted Edwards curve
$\mathcal{P}_1 = (\mathcal{P}_1[\eta-1], \cdots \mathcal{P}_1[1], \mathcal{P}_1[0])$	Operand of η-words
$\mathcal{P}_2$	Product of 2η-words
T_i	Temporary variables
μ, ν	Iteration control variables
a	Finite Field element a

Function 1 Squaring of two numbers.

Input: Two operands $\mathcal{P}_1 = (\mathcal{P}_1[\eta-1], \cdots \mathcal{P}_1[1], \mathcal{P}_1[0])$
Output: Product $\mathcal{P}_2 = \mathcal{P}_1 \times \mathcal{P}_1 = (\mathcal{P}_2[2\eta-1], \cdots \mathcal{P}_2[1], \mathcal{P}_2[0])$

```
1.  $T_1 \leftarrow \mathcal{P}_2[0] \leftarrow 0$
2.  for $\mu$ from 1 to $\eta$ do
3.      for $\nu$ from 0 to $\mu/2-1$ do
4.          $T_1 \leftarrow T_1 + \mathcal{P}_1[\nu]\mathcal{P}_1[\mu-\nu-1]$
5.      end for
6.      $\mathcal{P}_2[\mu-1] \leftarrow T_1 \; mod \; 2^{\omega}; T_1 \leftarrow T >> \omega$
7.  end for
8.  for $\mu$ from $\eta$ to $2\eta-3$ do
9.      for $\nu$ from $\mu-\eta+1$ to $(\mu-1)/2$ do
10.         $T_1 \leftarrow T_1 + \mathcal{P}_1[\nu]\mathcal{P}_1[\mu-\nu]$
11.     end for
12.     $\mathcal{P}_2[\mu] \leftarrow T_1 \; mod \; 2^{\omega}; T_1 \leftarrow T >> \omega$
13. end for
14. $\mathcal{P}_2[2\eta-2] \leftarrow T_1 \; mod \; 2^{\omega}; \mathcal{P}_2[2\eta-1] \leftarrow 0$
15. $T_1 \leftarrow 0$
16. for $\mu$ from 0 to $\eta-1$ do
17.     $T_1 \leftarrow T_1 + \mathcal{P}_1[\mu]\mathcal{P}_1[\mu] + 2(\mathcal{P}_2[2\mu+1]2^{\omega} + \mathcal{P}_2[2\mu])$
18.     $\mathcal{P}_2[2\mu] \leftarrow T_1 \; mod \; 2^{\omega}; T_1 \leftarrow T >> \omega$
19.     $\mathcal{P}_2[2\mu+1] \leftarrow T_1 \; mod \; 2^{\omega}; T_1 \leftarrow T >> \omega$
20. end for
21. return $(\mathcal{P}_2[2\eta-1], \cdots, \mathcal{P}_2[1], \mathcal{P}_2[0])$
```

The Function 1 plays a vital role in the Modular Inversion algorithm as the frequency of multiplying in others is more. So the efficiency of this Function increases the efficiency of incorporated algorithms in many folds. The symmetrical factor of the large Integer Multiplicand and Multiplier in the Square Function, $\mathcal{P}_1$ provides better performs compared to asymmetric Integer Multiplication function. In a regular Multiplication based Squaring, all interjacent outcomes of the form $\mathcal{P}_{1_\kappa} \times \mathcal{P}_{1_\nu}$ with $\nu! = \kappa$ computed twice. The Function 1 computes these interjacent outcomes merely one time and later shifts words to the right to obtain the Doubled result by shrinking excess processing cost.
The two nested loops in the Function compute the Double of interjacent outcomes $\mathcal{P}_{1_\kappa} \times \mathcal{P}_{1_\nu}$ equivalent loops in multiplication algFuncorithm used in this HSSM. The initial and the final interjacent outcomes are attached to the interleaved blocks, and reduction in the frequency of inner blocks repetition differs the Squaring Function from the regular Doubling. The modified exit-control statements reduce the overall Multiplications carried out by the dual repetitive blocks from η^2-2 to $(\eta^2-\eta)/2$.

4.3 Modular Reduction

The Multiplication and Squaring of two numbers of bit length η results in 2η length of the result. Since the value of η is large, subtracting the prime number from the result consumes much time. The Reduction is a process of minimizing the result of 2η bit length to η bit length less than the prime number chosen, which is the same as reducing the result belong to the selected Prime Field.

Function 2 Reducing 2η bit length result.

Input: Prime number p, $\mathcal{Z} = \mathcal{A} * \mathcal{A}$, a 2η-word.
Output: Reduced product Z' having η-word. $(\mathcal{A} * \mathcal{A})$ modulo p.

```
1: $Z_H \leftarrow \mathcal{A}[2m], \mathcal{A}[2m-1], \cdots, \mathcal{A}[m]$
2: $Z_L \leftarrow \mathcal{A}[m-1], \mathcal{A}[m-2], \cdots, \mathcal{A}[0]$
3: Write the Prime number $p$ as
       $2p = 2(2^k - C) = 2^n - d$ where $d = 2 * C$
4: $Z' \leftarrow Z_H.d + Z_L$
5: while $Z' > p$ do
6:     $Z' \leftarrow Z' - p$
7: end while
8: return $Z'$
```

The Modular Reduction Function can be implemented in two methods efficiently. In Function 2, the first method, the value of $\mathcal{Z}_H$ is left-shifted n times, where n is the bit length of the integer d. The intermediate results and the value $\mathcal{Z}_L$ are added and reduced further using subtraction. In Function 3, the second method, the usual multiplication algorithm is used instead of the left shift, and further reduction is obtained by subtraction.

Function 3 Calculating $\mathcal{Z}$ in Reduction function.

```
Input: $\mathcal{Z}_H, \mathcal{Z}_L$ and $d = 2 * C$.
Output: $\mathcal{Z}'$, where $\mathcal{Z}' = \mathcal{Z}_H * d + \mathcal{Z}_L$.
1: Convert to binary form.
2: while $d > 0$ do
3:     left shift $\mathcal{Z}_H$ $log(d)$ times.
4:     $\mathcal{Z}' \leftarrow \mathcal{Z}' + \mathcal{Z}_H$
5:     $d = d/2$
6: end while
7: return $\mathcal{Z}'$
```

4.4 Modular Inversion

Modular Inverse of an integer a is an integer x such that $a.x \equiv 1\ (mod\ p)$ Since the elements of the Prime Field p are from 0 to $p-1$, the modular Inversion, *i.e.,* reducing the integer obtained after multiplying and squaring to an integer in the Prime Field p.
Modular Inversion is the most consuming process in the encryption algorithm. The Extended Euclidean Algorithm and Inversion *via* exponentiation based on Fermat's little theorem are two principal approaches for evaluating an inversion in F_p. A Fermat-based inversion in F_p with $p = 2^k - c$ costs about $k-1$ squaring and no more than $2.log_2(k-1)$ multiplications.
For the 223 bit Prime Field with prime number $p = 2^{223} - 235$ the optimized chain Fermat-based inversion algorithm is shown in Algorithm 1.

Algorithm 1 Calculating Inversion for $p = 2^{223} - 235$

```
Input: Finite Field element a.
Output: The inversion $a^{-1}$ in $F_p$ as $t = a^{p-2}\ mod\ p, p = 2^{223} - 235$.
1: $a_2 \leftarrow a^2$
2: $a_9 \leftarrow (a_2)^4.a$
3: $x_1 \leftarrow 2; t \leftarrow (a_2.a_2)^{x_1}.a_9$
4: $x_2 \leftarrow x_3 \leftarrow 2^5; t \leftarrow (t)^{x_2}.t$
5: for $i$ from 1 to 4 do
6:     $x_3 \leftarrow (x_3)^2; t \leftarrow (t)^{x_3}.t$
7: end for
8: $x_3 \leftarrow (x_2)^2; t \leftarrow ((t)^{x_3}.t)^{x_2}$
9: $x_3 \leftarrow x_2.4; t \leftarrow ((t)^{x_3}.a_9)^2.a$
10: return $t$
```

From the given integer a, which belongs to the Finite Field, the algorithm has to calculate the value a^{p-2} where $p = 2^{223} - 235$. The algorithm calculates intermediate powers and reduces at each step by calling the modular reduction function. The intermediate results are stored in the array to prevent re-calculating the values calculated earlier, thus increasing speed. The modular Reduction of any number results in values being in the finite field save the memory required to store non-reduced values. The squaring and multiplications are obtained by the respective functions described before.
The total number of squaring and multiplications required to calculate Inversion for the prime number $p = 2^{223} - 235$ is 173 squaring and 12 multiplications. As the squaring of a number is considerably faster than multiplying the number twice, Squaring Function is called wherever feasible to make it more efficient.
The algorithm 1 described, calculates the Inversion of any number belonging to the Finite Field chosen significantly faster than most previously implemented versions. The inversion algorithms for the prime numbers $p = 2^{191} - 19$, $p = 2^{159} - 91$ and $p = 2^{255} - 19$ follows the same methodology with few required changes required accordingly.

Algorithm 2 Calculating Inversion for $p = 2^{159} - 91$

```
Input: Finite Field element a.
Output: Inversion $a^{-1}$ in $F_p$ as $t = a^{p-2}\ mod\ p, p = 2^{159} - 91$.
1: $a_2 \leftarrow a^2$
2: $a_4 \leftarrow (a_2)^2$
3: $a_9 \leftarrow (a_4)^2.a$
4: $a_{15} \leftarrow (a_9.a_4.a_2)$
5: $a_{31} \leftarrow (a_{15})^2.a$
6: $x_1 \leftarrow 2^4; t \leftarrow (a_{31})^{x_1}.a_{15}$
7: $x_2 \leftarrow (x_1).2; t \leftarrow (t)^{x_2}.t$
8: $x_3 \leftarrow x_1.x_2; t \leftarrow (t)^{x_3}.t$
9: $x_3 \leftarrow (x_3)^2.2; t \leftarrow (t)^{x_3}.t$
10: $x_3 \leftarrow (x_3)^2; t \leftarrow (t)^{x_3}.t$
11: $x_3 \leftarrow (x_3)^2; t \leftarrow (t)^{x_3}.t$
12: $x \leftarrow (x_1).2^3; t \leftarrow (t)^x.a_{31}.a_2$
13: return $t$
```

Algorithm 3 Calculating Inversion for $p = 2^{191} - 19$

```
Input: Finite Field element a.
Output: The inversion $a^{-1}$ in $F_p$ as $t = a^{p-2}\ mod\ p, p = 2^{191} - 19$.
1: $a_2 \leftarrow a^2$
2: $a_4 \leftarrow (a_2)^2$
3: $a_7 \leftarrow a_4.a_2.a$
4: $a_9 \leftarrow (a_4)^2.a$
5: $a_{15} \leftarrow (a_9.a_4.a_2)$
6: $a_{31} \leftarrow (a_{15})^2.a$
7: $x_2 \leftarrow 2^5; t \leftarrow (a_{31})^{x_2}.a_{31}$
8: $x_1 \leftarrow 2^3; t \leftarrow (t)^{x_1}.a_7$
9: $x_3 \leftarrow (x_2)^2.x_1; t \leftarrow (t)^{x_3}.t$
10: $x_3 \leftarrow (x_2)^4.x_1; t \leftarrow (t)^{x_3}.t$
11: $x_3 \leftarrow (x_3)^2; t \leftarrow (t)^{x_3}.t$
12: $x_3 \leftarrow (x_3)^2; t \leftarrow (t)^{x_3}.t$
13: $x_3 \leftarrow (x_1)^2.2; t \leftarrow (t)^{x_3}.(a_{31})^4.(a_7)^2$
14: return $t$
```

Fermat's little theorem states that Any prime number perfectly divides one of the powers - 1 of any progression, *i.e.,* $\exists_t \ni [\rho|(a^t - 1)]$. The exponent of this power is a sub-multiple of the given prime number - 1, *i.e.,* $t|\rho - 1$. The first power that satisfies the property, all those whose exponents are multiple of the exponent of the first, still satisfy the property.
Consider an Integer $a = 3$ and a Prime $\rho = 5$, then, $3^5 = 243$ and $243 - 3 = 240 = 5 \times 48$, *i.e.,* $3^5 \equiv 3\ mod\ 5$ and $3^4 = 81$ and $81 - 1 = 80 = 5 \times 16$, *i.e.,* $3^4 \equiv 1\ mod\ 5$. Similarly, let $a = 3$ and $\rho = 11$, then, $3^{11} = 177147$ and $177147 - 3 = 177144 = 11 \times 16104$, *i.e.,* $3^{11} \equiv 3\ mod\ 11$ and $3^{10} = 59049$ and $59049 - 1 = 59048 = 11 \times 5368$, *i.e.,* $3^{10} \equiv 1\ mod\ 11$.

Thus, $31 = 2^5 - 1$, hence, a^{31} is represented as a^{2^5-1} given in the equation (7)

$$a^{31} = a^{2^5-1} \tag{7}$$

In general,

$$a^{(2^r-1)2^s} a^{(2^s-1)} = a^{(2^{(r+s)}-2^s)+(2^s-1)} = a^{(2^{r+s}-1)} \tag{8}$$

$$a^{(2^5-1)2^5} a^{(2^5-1)} = a^{(2^{(5+5)}-2^5)+(2^5-1)} = a^{(2^{10}-1)} \tag{9}$$

$$a^{(2^{80}-1)2^{80}} a^{(2^{80}-1)} = a^{(2^{(80+80)}-2^{80})+(2^{80}-1)} = a^{(2^{160}-1)} \tag{10}$$

$$a^{(2^{160}-1)2^{40}} a^{(2^{40}-1)} = a^{(2^{160+40}-2^{40})+(2^{40}-1)} = a^{(2^{200}-1)} \tag{11}$$

$$\begin{aligned} \left(a^{(2^{215}-1)2^7} a^{91}\right)^{2^1} a^1 &= a^{2^{(215+7+1)}-2^{(7+1)}+(9)(2)+1} && (12)\\ &= a^{2^{223}-256+19} && (13)\\ &= a^{2^{223}-237} && (14) \end{aligned}$$

For the prime number $p = 2^{159} - 91$, the algorithm 2 follows the same procedure. The algorithm requires 155 squares and 15 multiplications to calculate all the intermediate powers to find the inverse of the given number. All the intermediate values contributed to reduced multiplication values and squared values calculated by the Brute-Force method. This comparison is to determine whether all the intermediate calculations and reducing perform correctly. The Inversion algorithm is then used to calculate the inversion of the denominator in point addition and point doubling. As there is no division arithmetic function for the prime group, a division is performed by finding Inversion of the denominator and then multiplying with the numerator.

Algorithm 4 Calculating Inversion for $p = 2^{255} - 19$

```
Input: Finite Field element a.
Output: The inversion $a^{-1}$ in $F_p$ as $t = a^{p-2}\ mod\, p, p = 2^{255} - 19$.
 1: $a_2 \leftarrow a^2$
 2: $a_4 \leftarrow (a_2)^2$
 3: $a_7 \leftarrow (a_4)^2.a$
 4: $a_{15} \leftarrow (a_7)^2.a$
 5: $x_1 \leftarrow 2^3; t \leftarrow (a_{15})^{x_1}.a_2.a$
 6: $x_2 \leftarrow (x_1).2; t \leftarrow (t)^{x_2}.t.a$
 7: $x_3 \leftarrow x_1.x_2; t \leftarrow (t)^{x_3}.t$
 8: $x_3 \leftarrow (x_3)^2.2; t \leftarrow (t)^{x_3}.t$
 9: $x_3 \leftarrow (x_3)^2; t \leftarrow (t)^{x_3}.t$
10: $x_3 \leftarrow (x_3)^2; t \leftarrow (t)^{x_3}.t$
11: $x \leftarrow (x_1).2^3; t \leftarrow (t)^x.a_{31}.a_2$
12: return $t$
```

The Algorithms 2, 3, 1 and 4 describes the Point Inversion for $p = 2^{159} - 91$, $p = 2^{191} - 19$, $p = 2^{223} - 235$ and $p = 2^{255} - 19$ respectively. These individual Algorithms call Multiplication, Squaring, and Reduction algorithms. The number of times multiplication and squaring called and efficiency of Multiplication and Squaring decides the efficiency of the Inversion algorithm. The intermediate calculations are stored, and complex calculations break into sub-problems to increase efficiency. All intermediate calculations are stored as computationally expensive and used when required. The algorithms for $p = 2^{191} - 19$ and $p = 2^{159} - 91$.

Table 3. Comparison of Inversion Algorithms over Existing Schemes

bits	RPSM	LHS	PIECC
160	$158Sqr + 12Mul$	$158Sqr + 11Mul$	$155Sqr + 15Mul$
192	$190Sqr + 08Mul$	$190Sqr + 12Mul$	$167Sqr + 16Mul$
224	$222Sqr + 15Mul$	$222Sqr + 13Mul$	$173Sqr + 12Mul$
256	$254Sqr + 08Mul$	$254Sqr + 11Mul$	$122Sqr + 16Mul$

The number of Multiplication and Squaring in the Inversion Algorithms are as given in Table 3. PIECC uses 173 Squaring, and 12 Multiplication operations for 224 bit; and 122 Squaring and 16 Multiplying operations. A 50% reduction in the number of Squaring operations compared to LHS and RPSM reduces the Computation time of PIECC algorithm.

4.5 Scalar Point Multiplication

Scalar point multiplication multiplies a fixed point on the selected elliptic curve and performs scalar multiplication κ times. Scalar point multiplication is one of the time-consuming processes in elliptic curve cryptology. In this step, the message is represented by any point on the elliptic curve. Then the selected Point is multiplied κ times, which is the Elliptic Curve Discrete Logarithm problem. Key generation and Diffie-Hellman key exchange require one Scalar multiplication. In Diffie-Hellman key exchange, κ is kept secret. The computation of $\kappa.P$ is the same for public and secret κ. The problem with the standard, straightforward implementation of scalar multiplication is safety. An attacker can measure time and can try and deduce information about κ. Montgomery Ladder Algorithm used for performing Scalar multiplication. The safety and execution of the algorithm depend on the value of κ. The steps involved are shown in algorithm 5.

Algorithm 5 Calculating Scalar Point Multiplication.

```
Input: An integer $\kappa > 0$ and a Point $\mathcal{P}_0$.
Output: $\mathcal{P}_3 = \kappa\mathcal{P}_0$.
 1: set $\kappa \leftarrow (\kappa_{l-1}, \kappa_{l-2}, \cdots, \kappa_1, \kappa_0)$
 2: set $\mathcal{P}_1 \leftarrow 0; \mathcal{P}_2 \leftarrow \mathcal{P}_0$
 3: for $i$ from $l-1$ downto 0 do
 4:     if $\kappa_i = 1$ then
 5:         $\mathcal{P}_1 \leftarrow \mathcal{P}_1 + \mathcal{P}_2; \mathcal{P}_2 \leftarrow 2\mathcal{P}_2$
 6:     else
 7:         $\mathcal{P}_2 \leftarrow \mathcal{P}_2 + \mathcal{P}_1; \mathcal{P}_1 \leftarrow 2\mathcal{P}_1$
 8:     end if
 9: end for
10: $\mathcal{P}_3 \leftarrow \mathcal{P}_1$
11: return $\mathcal{P}_3$
```

The algorithm calculates $\kappa.\mathcal{P}_0$ by repeated point addition and point multiplication. For point addition and point multiplication, the standard formula requires Addition, Subtraction, Multiplication, Reduction, and Inversion. Hence Scalar Point Multiplication calls corresponding arithmetic functions whenever required. The algorithm computes the point multiplication in a fixed time, thus providing immunity against time-based attacks. This algorithm performs the same number of point additions and doubles regardless of the value of κ. The algorithm does not leak any information through timing or power.

5. PERFORMANCE ANALYSIS

The Elliptic Curve Cryptosystem designed consists of Sensor devices or the mote as the fundamental component. These devices are equipped with 4-16 KiloBytes of main memory and 256 KiloBytes of nonvolatile memory. In Elliptic Curve Cryptosystem, the selected Curve Parameters and Prime numbers are in Binary representation. The system encodes data into a Point on the Elliptic Curve. The required number of Inversion and Reduction is performed over the Elliptic Curve Point. The Scalar multiplication is performed to encrypt the data in Elliptic Curve Diffie-Hellman Cryptography. The encrypted data is sent from the Sender to the Receiver.

Computation Time: The reduction in Computation Time of Multiplication, Squaring, Inversion, and Scalar Multiplication used in the Elliptic Curve Crypto-system shows the better performance of PIECC over the existing algorithms: Liu [7], and Reverse Product Scanning methods [6] with Fermat-based Inversion. The Computation time of the Functions and Algorithms is given in Table 5.

Computation time of Multiplication: Fig. 1 shows the time taken to perform Multiplication over Elliptic Curves of 160, 192, 224, 255 bits. The multiplication function of PIECC over the 192-bit curve consumes 2632 Clock Cycles, LHS consumes 2706 Clock Cycles, and RPSM consumes 3831 Clock Cycles for 192 bits. The processing time of multiplication in PIECC is 2.7% better than LHS and 36% better than RPSM for 256 bits. The use of the High-Speed Split multiplier reduces the computation Time of multiplication.

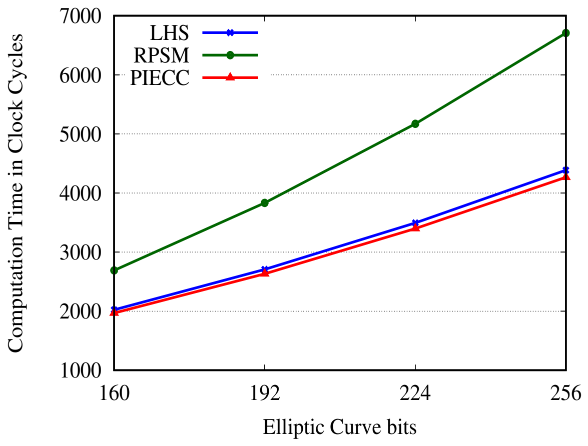

Fig. 1. Computation Time of Multiplication LHS, RPSM and PIECE Elliptic Curves

Computation time of Squaring: Fig. 2 shows the time taken to perform Scalar Multiplication over Elliptic Curves of 160, 192, 224, 255 bits. The Scalar multiplication algorithm of PIECC over the 192-bit curve consumes 1603 Clock Cycles, LHS consumes 1651 Clock Cycles, and RPSM consumes 2457 Clock Cycles for 192 bits. The processing time of multiplication in PIECC is 49% better than LHS and 69% better than RPSM for 256 bits. The use of the High-Speed Split multiplier, HSSM based Inversion reduces the computation Time of Scalar Multiplication.

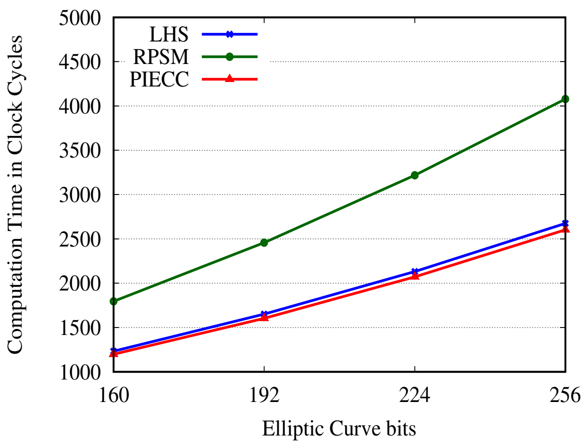

Fig. 2. Computation Time of Squaring in LHS, RPSM and PIECE Elliptic Curves

Computation time of Inversion: Fig. 3 depicts the time of Inversion over Elliptic Curves of 160, 192, 224 and, 255 bits. The Inversion algorithm over the 192-bit curve uses 309813 Clock Cycles, LHS uses 346162 Clock Cycles, and RPSM uses 497478 Clock Cycles for 192 bits. For 224 bits curve, the Point Inversion algorithm takes 399059 Clock Cycles, LHS takes 518504 Clock Cycles, and RPSM takes 791946 Clock Cycles. The processing time of Inversion in PIECC is 46.6% better than LHS and 64.5% better than RPSM for 256 bits. The use of limited High-Speed Split multiplier and HSSM-based Squaring reduces with reduced code decreases the computation Time of Inversion.

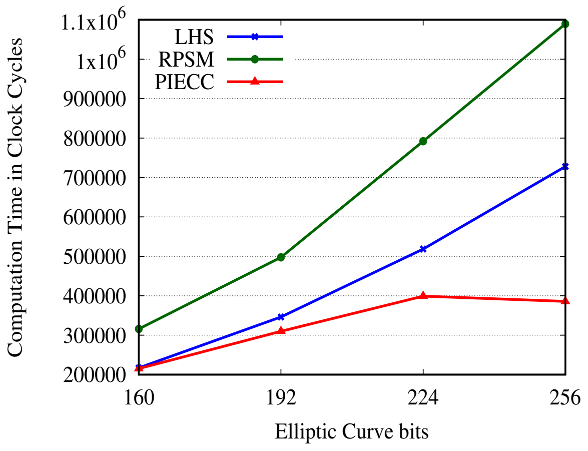

Fig. 3. Computation Time of Inversion in LHS, RPSM and PIECE Elliptic Curves

Computation time of Scalar Multiplication: Fig. 4 shows the time taken to perform Scalar Multiplication over Elliptic Curves of 160, 192, 224, 255 bits. The Scalar multiplication algorithm of PIECC over the 192-bit curve consumes Clock Cycles, LHS consumes Clock Cycles and, RPSM consumes Clock Cycles for the 192 bits. The processing time of multiplication in PIECC is 49% better than LHS and 69% better than RPSM for 256 bits. The High-Speed Split

Table 4. Comparison of Computation Time of Inversion Algorithms over Existing (in Clock Cycles) Schemes

curve	**Multiplication**			**Squaring**			**Inversion**			**ScalarMultiplication**		
bits	LHS[7]	RPSM[6]	PIECC	LHS[7]	RPSM[6]	PIECC	LHS[7]	RPSM[6]	PIECC	LHS[7]	RPSM[6]	PIECC
160	2022	2690	1967	1233	1795	1198	217056	315890	215195	1638000	2296115	1550781
192	2706	3831	2632	1651	2457	1603	346162	497478	309813	2433000	3616027	2079402
224	3494	5170	3398	2131	3218	2071	518504	791946	399059	3489000	5756432	2564260
256	4386	6707	4266	2675	4078	2602	727696	1089468	385700	4798000	7919036	2428489

multiplier and HSSM based Inversion reduces the computation Time of Scalar Multiplication.

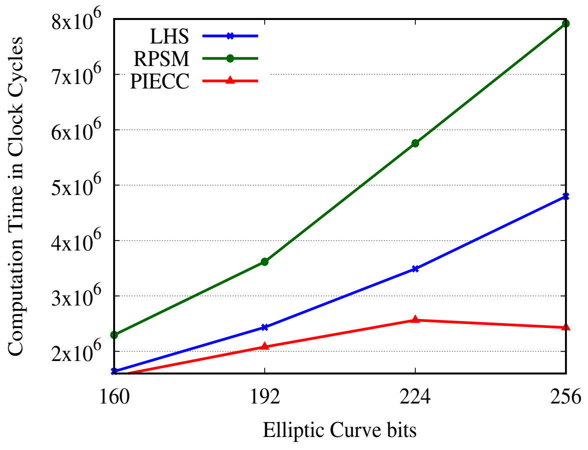

Fig. 4. Computation Time of Scalar Multiplication in LHS, RPSM and PIECE Elliptic Curves

Memory usage: The reduction in Memory usage of PIECC in Fig. 5 shows the better performance of PIECC over the existing algorithms: Liu [7], and Reverse Product Scanning methods [6] with Fermat-based Inversion. The PIECE uses 412Bytes, LHS uses 446Bytes, and RPSM uses 1174Bytes of RAM. The PIECE uses 11434Bytes, LHS uses 12156Bytes, and RPSM uses 19000Bytes of ROM. The PIECC uses ROM 5.5% lesser than LHS and 39.8% than RPSM. The RAM used by PIECE is 7.6% less than LHS and 64.9% less than RPSM. The memory usage is reduced by minimizing the intermediate memory usage for data and the code.

6. CONCLUSIONS

In an IoT environment, devices store and transmit data over communication media must be protected from illegitimate usage. Encryption controls for Data security at rest and in transit are essential. Elliptic Curve Cryptography enhances IoT security providing secure data communication in the Internet of Things Application. The proposed Point Inversion algorithm for Elliptic Curve Cryptology uses High-Speed Split Multiplier and HSSM based Squaring in Scalar Multiplication to secure IoT Data Communication. The PIECC optimizes the data and code Storage, improving the performance of IoT devices using limited intermediate registers to carry out Cryptographic functions. The proposed algorithm is almost two times faster than LHS and three times than RPSM. The PIECC uses 6% less memory than the LHS and 41% than RPSM by the reduction in intermediate memory usage and code. The Modular Inversion for other Curves can apply Factorization over Montgomery-friendly Prime Representation.

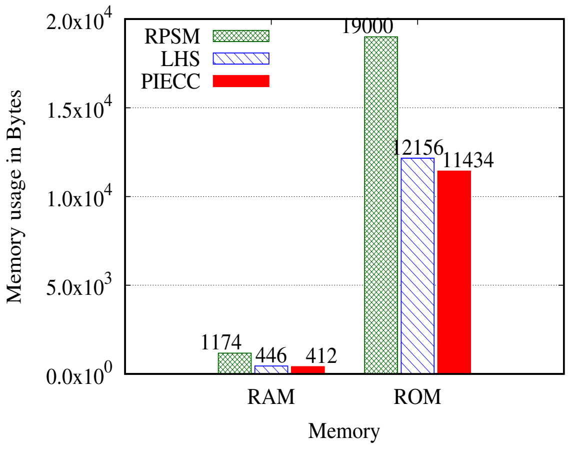

Fig. 5. Memory usage of PIECC in LHS, RPSM and PIECE Elliptic Curves

7. REFERENCES

[1] A. R. Omondi, *Cryptography Arithmetic*. Springer, 2020, vol. 77.

[2] S. Kondo and T. Watari, "String-Theory Realization of Modular Forms for Elliptic Curves with Complex Multiplication," *Springer Journal on Communications in Mathematical Physics*, vol. 367, no. 1, pp. 89–126, 2019.

[3] M. G. Padmashree, J. S. Arunalatha, and K. R. Venugopal, "HSSM: High Speed Split Multiplier for Elliptic Curve Cryptography in IoT," *in Proceedings of the Fifteenth International Conference on Information Processing (ICInPro-2019)*, pp. 123–127, December 2019.

[4] M. Laporta and A. Pizzirani, "A Binary Algorithm With Low Divergence for Modular Inversion on SIMD Architectures," *Springer Journal on Ricerche di Matematica*, vol. 63, no. 1, pp. 187–199, 2014.

[5] G. Gallin and A. Tisserand, "Hyper-Threaded Multiplier for HECC," *in Proceedings of the Fifty First IEEE Asilomar Conference on Signals, Systems, and Computers*, pp. 447–451, October 2017.

[6] Z. Liu, H. Seo, J. Großschädl, and H. Kim, "Reverse Product-Scanning Multiplication and Squaring on 8-Bit AVR Processors," *in Proceedings of the Sixteenth Springer International Conference on Information and Communications Security*, pp. 158–175, 2015.

[7] Z. Liu, X. Huang, Z. Hu, M. K. Khan, H. Seo, and L. Zhou, "On Emerging Family of Elliptic Curves to Secure Internet of Things: ECC Comes of Age," *IEEE Transactions on Dependable and Secure Computing*, vol. 14, no. 3, pp. 237–248, 2017.

[8] M. Bedoui, B. Bouallegue, B. Hamdi, and M. Machhout, "An Efficient Fault Detection Method for Elliptic Curve Scalar Multiplication Montgomery Algorithm," *in Proceedings of the IEEE International Conference on Design and Test of Integrated Micro and Nano-Systems*, pp. 1–5, 2019.

[9] J. Ding and S. Li, "Broken-Karatsuba multiplication and its application to Montgomery modular multiplication," *in Proceedings of the Twenty Seventh International Conference on Field Programmable Logic and Applications*, pp. 5–8, 2017.

[10] D. Basu Roy and D. Mukhopadhyay, "High-Speed Implementation of ECC Scalar Multiplication in GF(p) for Generic Montgomery Curves," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 7, pp. 1587–1600, 2019.

[11] Z. Gu and S. Li, "A Generalized RNS Mclaughlin Modular Multiplication with Non-Coprime Moduli Sets," *IEEE Transactions on Computers*, vol. 68, no. 11, pp. 1689–1696, 2019.

[12] M. Safieh and J. Freudenberger, "Montgomery Modular Arithmetic over Gaussian Integers," *in Proceedings of the Twenty Fourth IEEE International Conference on Information Technology*, pp. 6–9, 2020.

[13] W. Yu, K. Wang, B. Li, and S. Tian, "Montgomery Algorithm over a Prime Field," *IEEE Chinese Journal of Electronics*, vol. 28, no. 1, pp. 39–44, 2019.

[14] G. Gallin and A. Tisserand, "Generation of Finely-Pipelined GF(P) Multipliers for Flexible Curve based Cryptography on FPGAs," *IEEE Transactions on Computers*, vol. 68, no. 11, pp. 1612–1622, 2019.

[15] J. Ding and S. Li, "A Low-Latency and Low-Cost Montgomery Modular Multiplier based on NLP Multiplication," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 7, pp. 1319–1323, 2020.

[16] X. Feng, S. Li, and S. Xu, "RLWE-Oriented High-Speed Polynomial Multiplier Utilizing Multi-lane Stockham NTT Algorithm," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 3, pp. 556 – 559, 2020.

[17] G. Leelavathi, K. Shaila, and K. R. Venugopal, "Elliptic Curve Crypto Processor on FPGA using Montgomery Multiplication with Vedic and Encoded Multiplier over GF (2^m) for Nodes in Wireless Sensor Networks," *in Proceedings of the Thirteenth IEEE International Conference on Industrial and Information Systems*, no. 978, pp. 207–210, 2019.

[18] L. Li and S. Li, "High-Performance Pipelined Architecture of Point Multiplication on Koblitz Curves," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 65, no. 11, pp. 1723–1727, 2018.

[19] E. Wenger and J. Großschädl, "An 8-bit AVR-based Elliptic Curve Cryptographic RISC Processor for the Internet of Things," *in Proceedings of the Forty Fifth IEEE/ACM International Symposium on Microarchitecture Workshops*, pp. 39–46, 2012.

[20] E. A. Abdulrahman and A. Reyhani-Masoleh, "New Regular Radix-8 Scheme for Elliptic Curve Scalar Multiplication without Pre-Computation," *IEEE Transactions on Computers*, vol. 64, no. 2, pp. 438–451, 2015.

[21] R. Salarifard, S. Bayat-Sarmadi, and H. Mosanaei-Boorani, "A Low-Latency and Low-Complexity Point-Multiplication in ECC," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 9, pp. 2869–2877, 2018.

[22] S. Ali and M. Cenk, "Faster Residue Multiplication Modulo 521-bit Mersenne Prime and an Application to ECC," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 8, pp. 2477–2490, 2018.

[23] D. Mukhopadhyay and D. B. Roy, "Revisiting FPGA Implementation of Montgomery Multiplier in Redundant Number System for Efficient ECC Application in GF(p)," *in Proceedings of the Twenty Eighth IEEE International Conference on Field-Programmable Logic and Applications*, pp. 323–326, August 2018.

[24] D. Amiet, A. Curiger, and P. Zbinden, "Flexible FPGA-Based Architectures for Curve Point Multiplication over GF(p)," *in Proceedings of the Nineteenth IEEE Euromicro Conference on Digital System Design, DSD 2016*, pp. 107–114, 2016.

[25] B. Rashidi, "Low-Cost and Fast Hardware Implementations of Point Multiplication on Binary Edwards Curves," *in Proceedings of the Twenty Sixth IEEE Iranian Conference on Electrical Engineering*, pp. 17–22, 2018.

[26] M. Selim Hossain and Y. Kong, "FPGA-based Efficient Modular Multiplication for Elliptic Curve Cryptography," *in Proceedings of the Twenty Fifth IEEE International Telecommunication Networks and Applications Conference*, pp. 191–195, 2015.

[27] M. Al-Asli, M. E. Elrabaa, and M. Abu-Amara, "FPGA-Based Symmetric Re-Encryption Scheme to Secure Data Processing for Cloud-Integrated Internet of Things," *IEEE Journal on Internet of Things*, vol. 6, no. 1, pp. 446–457, 2019.

[28] K. C. C. Loi and S. B. Ko, "Scalable Elliptic Curve Cryptosystem FPGA Processor for NIST Prime Curves," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 11, pp. 2753–2756, 2015.

[29] S. Xu, H. Gu, L. Wang, Z. Guo, J. Liu, X. Lu, and D. Gu, "Efficient and Constant Time Modular Inversions over Prime Fields," *in Proceedings of the Thirteenth IEEE International Conference on Computational Intelligence and Security*, vol. 2018-January, pp. 524–528, 2018.

[30] P. Choi, M. K. Lee, J. H. Kim, and D. K. Kim, "Low-Complexity Elliptic Curve Cryptography Processor Based on Configurable Partial Modular Reduction over NIST Prime Fields," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 65, no. 11, pp. 1703–1707, 2018.

[31] SafeCurves. [Online]. Available: https://safecurves.cr.yp.to/