IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received November 15, 2019, accepted December 2, 2019, date of publication December 9, 2019, date of current version December 23, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2958491

FPGA Implementation of High-Speed Area-Efficient Processor for Elliptic Curve Point Multiplication Over Prime Field

MD. MAINUL ISLAM[1], (Student Member, IEEE), MD. SELIM HOSSAIN[2], (Member, IEEE), MOH. KHALID HASAN[1], (Student Member, IEEE), MD. SHAHJALAL[1], (Student Member, IEEE), AND YEONG MIN JANG[1], (Member, IEEE)

[1]Department of Electronics Engineering, Kookmin University, Seoul 02707, South Korea
[2]Department of Electrical and Electronic Engineering, Khulna University of Engineering and Technology, Khulna 9203, Bangladesh

Corresponding author: Yeong Min Jang (yjang@kookmin.ac.kr)

This work was supported in part by the Ministry of Science and ICT (MSIT), South Korea, through the Information Technology Research Center (ITRC), and in part by the the Institute for Information and Communications Technology Promotion (IITP) under Grant 2018-0-01396.

ABSTRACT Developing a high-speed elliptic curve cryptographic (ECC) processor that performs fast point multiplication with low hardware utilization is a crucial demand in the fields of cryptography and network security. This paper presents field-programmable gate array (FPGA) implementation of a high-speed, low-area, side-channel attacks (SCAs) resistant ECC processor over a prime field. The processor supports 256-bit point multiplication on recently recommended twisted Edwards curve, namely, Edwards25519, which is used for a high-security digital signature scheme called Edwards curve digital signature algorithm (EdDSA). The paper proposes novel hardware architectures for point addition and point doubling operations on the twisted Edwards curve, where the processor takes only 516 and 1029 clock cycles to perform each point addition and point doubling, respectively. For a 256-bit key, the proposed ECC processor performs single point multiplication in 1.48 ms, running at a maximum clock frequency of 177.7 MHz in a cycle count of 262 650 with a throughput of 173.2 kbps, utilizing only 8873 slices on the Xilinx Virtex-7 FPGA platform, where the points are represented in projective coordinates. The implemented design is time-area-efficient as it offers fast scalar multiplication with low hardware utilization without compromising the security level.

INDEX TERMS Elliptic curve cryptography (ECC), elliptic curve point multiplication (ECPM), twisted Edwards curve, side-channel attacks (SCAs), field-programmable gate array (FPGA).

I. INTRODUCTION

Internet of Things (IoT) security has become a crucial issue in the present scenario of the Internet world. With the rapid development of wireless communication, the demand for IoT security is increasing day by day. Public key cryptography (PKC) or asymmetric cryptography [1], [2] is an excellent solution to fulfill the demand as it provides a key-agreement protocol between two sensor nodes in a wireless network and prevents unauthorized accesses to sensitive data during transmission over the network [3]–[5]. The two widely accepted PKC algorithms for cryptographic applications are Rivest-Shamir-Adleman (RSA) [6] and elliptic curve cryptography (ECC) [1]. RSA, proposed by Rivest, Shamir, and Adleman, is based on integer factorization, whose encryption strength depends on the key sizes taken. ECC, first introduced by Koblitz [7] and Miller [8] independently, is based on discrete logarithms, whose encryption strength is much difficult to break. To provide the same level of security, ECC requires a shorter key length than RSA. For example, 160-, 224-, and 256-bit ECC encryption keys provide equivalent security as 1024-, 2048-, and 3072-bit RSA encryption keys, respectively. The advantage of smaller key sizes is that they make ECC the best suited for the high-speed cryptographic processors as well as resource-constrained IoT devices. The authentication protocols for wireless sensor

The associate editor coordinating the review of this manuscript and approving it for publication was Yu-Chi Chen.

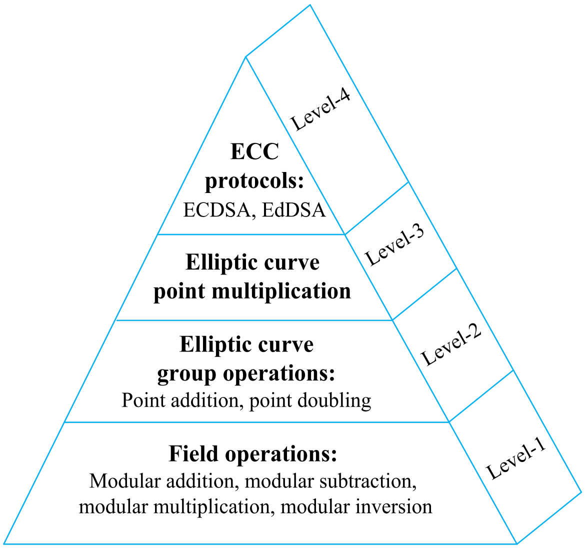

Level-4
ECC protocols:
ECDSA, EdDSA
Level-3
Elliptic curve point multiplication
Level-2
Elliptic curve group operations:
Point addition, point doubling
Level-1
Field operations:
Modular addition, modular subtraction, modular multiplication, modular inversion

FIGURE 1. Elliptic curve cryptography hierarchy.

nodes are adopting ECC primitives to provide a high level of security with optimal hardware resources.

Edwards curves, a family of elliptic curves, introduced by Edward [9], have gained much interest among cryptography researchers because of their fast group operations [10] and high immunity to side-channel attacks (SCAs) [11]. Edwards curves can offer strongly unified addition [12] formulas that can be used for both point addition and point doubling, ensuring side-channel security. An Edwards curve based cryptographic processor can be developed with low area utilization and low power consumption, providing high computational speed and high level of security. Edwards25519 is a twisted Edwards curve [12]–[15], which is the Edwards form of the elliptic curve "curve25519" [16]. It is mainly used for high-speed key generation as well as in Edwards curve digital signature algorithm (EdDSA) [13], [17].

A typical hierarchy of ECC consists of four successive levels as shown in Figure 1. The first level contains finite field arithmetic such as modular addition, subtraction, multiplication, and inversion. The second level comprises elliptic curve group operations such as point addition and point doubling, which accommodate a number of modular arithmetic. The third level relates to elliptic curve point/scalar multiplication (ECPM/ECSM) that integrates the elliptic curve group operations in a sequential manner. The top-level includes ECC protocols such as elliptic curve digital signature algorithm (ECDSA) and EdDSA.

The most dominant and time-consuming operation in ECC is ECPM, which is defined as $Q = k \cdot P$, where P is a base point on an elliptic curve, k is a scaler, and Q is another point on the elliptic curve. The main goal of ECPM is to generate the public key Q by multiplying the private key k with the base point P on the curve. It is mathematically difficult to find the value of the secret key by reversing the ECPM as $k = Q \cdot (P^{-1})$. Solving k from the points P and Q is regarded as the elliptic curve discrete logarithm problem (ECDLP) [1] that measures the weakness of ECC schemes. The security of an elliptic curve cryptosystem lies in the hardness of the ECDLP and the secrecy of the private key. The algorithms for ECPM are the binary method or double-and-add method, non-adjacent form (NAF) method, the Montgomery ladder algorithm, sliding window method, and fixed-base comb method [1], [18], [19]. In the binary method, since point addition and point doubling are performed serially following the binary bit pattern of the key, the method is vulnerable to SCAs (e.g., timing and power analysis attacks). In the Montgomery ladder technique, point addition and point doubling are performed in parallel, which cannot be distinguished by the bit pattern of the key; hence, this method precludes SCAs. ECPM can be performed representing the points in both affine and Jacobian or projective coordinates [1]. ECPM in Jacobian coordinates is faster than that in affine coordinates because it requires no modular division or inversion operation to perform point addition and point doubling. In affine coordinates, point addition and point doubling require inversion operation, which is the costliest arithmetic operation in finite fields. The cost of a modular inversion is the same as that of 80 modular multiplications.

An elliptic curve cryptosystem can be implemented with either a hardware or software approach. In this research, our focus is only on the hardware approach because the hardware implementation offers considerably faster operations compared with the software implementation. However, the hardware implementation of ECC with low hardware consumption and low time complexity is a very challenging task. Area and time are two contradictory parameters of an ECC processor, one of which has to be compromised to achieve high efficiency in terms of the other one. For better performance, the area-time (AT) product of the processor should be as small as possible. This research aims to develop an ECC processor well-suited for high-speed, low-power cryptographic devices by reducing the computation time and area required for ECPM.

A. RELATED WORKS AND MOTIVATION

Many hardware implementations of ECC processors over both the Galois binary field GF(2^n) and Galois prime field GF(p) have been documented in the literature, where some authors aimed to reduce computation time for fast data encryption and others aimed to reduce required hardware resources for small-device applications. Several field-programmable gate array (FPGA) implementations of ECC processors are proposed in the papers [20]–[43]. Ors *et al.* [20] proposed hardware implementation of a bit-length-efficient elliptic curve processor over GF(p) for PKC, providing a novel architecture for Montgomery modular multiplication. In [21], Sakiyama *et al.* proposed a reconfigurable hardware architecture for PKC that is suited for both the RSA and ECC. A novel hardware architecture for 256-bit ECC over GF(p) was developed by McIvor *et al.* [22], providing unified modular inversion and multiplication. In [23]–[25], the authors proposed residue number system (RNS) based hardware implementations of ECPM to achieve high-speed point multiplication. The design reported in [26] introduces an RNS-based multi-key elliptic curve cryptosystem that performs ECPM on 21 keys simultaneously between its pipeline registers, providing high throughput. A high-throughput cost-effective dual-field ECC processor was proposed in [27] that supports arbitrary elliptic curves with different field orders. Fan *et al.* [28] developed an embedded 192-bit multicore ECC processor over a prime field, in which multiple modular operations are performed in parallel to speed up ECPM. In [29], [30], the authors proposed FPGA-based flexible ECC processors over National Institute of Standards and Technology (NIST) prime fields that support all five NIST recommended curves without reconfiguring the hardware. Liu *et al.* [31] proposed a flexible dual-field ECC processor, providing both application-specific integrated circuit (ASIC) and FPGA implementations. They adopted a hardware-software approach for their ECC processor.

In [32], Ghosh *et al.* proposed parallel crypto devices for ECPM over GF(p), providing both FPGA and ASIC implementations, which are resistant to different SCAs. The design reported in [33] introduces a compact FPGA-based architecture for ECC over a 256-bit prime field using carry-chain logic. A power and timing resistant ECSM over programmable GF(p) was proposed by Ghosh *et al.* [34] that is resistant to differential power analysis (DPA) attacks. In [35]–[37], the authors proposed FPGA implementations of 256-bit ECC processors over NIST prime fields based on redundant signed digit (RSD) representations for carry free arithmetic to achieve high-speed point multiplication. The area-delay products of their designs are very low compared with that of the other available designs for taking the advantages of RSD-based modular arithmetic. A power-analysis-resistant ECC processor was developed by Lee *et al.* [38] using heterogeneous dual processing elements that can perform over both GF(2^n) and GF(p). In [39]–[41], the authors proposed FPGA-based ECPM over GF(p) by introducing a new parallel modular multiplier to take the full advantage of parallelism in group operations, which can provide high-speed point multiplication with low computational complexity. A high-performance ECC processor over NIST prime fields was developed by Hossain *et al.* [42], providing both ASIC and FPGA implementations that can perform fast scalar multiplication with low hardware utilization. They proposed both the affine and projective representations of their processor along with a projective to affine converter. Hu *et al.* [43] proposed a low hardware consumption ECC architecture over GF(p) in embedded applications, which is safe from simple power analysis (SPA) attacks.

B. OUR CONTRIBUTIONS

In this paper, FPGA implementation of a time-area-efficient 256-bit ECC processor over GF(p) is presented. The number of clock cycles, as well as computation time for point multiplication, is aimed to reduce as far as possible. The minimization of the hardware resources required for the group and modular operations is emphasized to reduce the occupied area of the processor. The major contributions of this paper can be summarized as follows:

- An efficient design for ECPM on a twisted Edwards curve named Edwards25519 is proposed to achieve faster point multiplication with higher security.
- The Montgomery ladder algorithm is adopted for the ECPM design to provide significant protection against probable SCAs (e.g., timing and power analysis attacks).
- The design is represented in projective coordinates instead of in affine coordinates to avoid modular inversion operation, which is computationally expensive.
- Novel hardware architectures are proposed for the twisted Edwards curve group operations (point addition and point doubling) minimizing the latency and the number of arithmetic modules as far as possible by manipulating parallelization technique.
- An optimized hardware architecture is proposed for radix-2 interleaved modular multiplication to perform faster group operations with low area utilization.
- Furthermore, the area-delay product of the proposed ECPM design is very low and the throughput of the design is high compared with that of the other similar works, which ensure better performance of the ECC processor.

The remaining of this paper is organized as follows:

The acronyms used in this paper are enlisted in Table 1. The mathematical background of the twisted Edwards curve is described in Section II. The hardware architectures for the ECC operations are proposed in Section III. The implementation and simulation results of the proposed ECC designs are presented in Section IV. A performance comparison of our ECPM design with other available designs is shown in Section V. Finally, in Section VI, this research work is summarized and concluded.

II. MATHEMATICAL BACKGROUND

This section presents the twisted Edwards curve and group law of this curve. The point addition and point doubling formulas for the Edwards25519 curve in projective coordinates are also presented in this section.

A. TWISTED EDWARDS CURVE

A twisted Edwards curve over a prime field $\mathbb{F}_p$ with not characteristic 2 is defined by the equation

$$e_{a,d} : ax^2 + y^2 = 1 + dx^2y^2 \tag{1}$$

where $a, d \in \mathbb{F}_p \setminus \{0, 1\}$ with $a \neq d$. In the case of $a = 1$, the curve is called Edwards curve, which is untwisted. Therefore, the twisted Edwards curve is a generalization of Edwards curve. When $a = -1$ and $d = -121665/121666$, the curve is called Edwards25519, which is the Edwards form of the elliptic curve "curve25519" over $\mathbb{F}_p$, where $p = 2^{255} - 19$. In the case of $a = -1$, the curve $e_{a,d}$ will be

$$e_d : -x^2 + y^2 = 1 + dx^2y^2 \tag{2}$$

TABLE 1. List of acronyms and corresponding meanings.

Acronym	Meaning
ASIC	Application Specific Integrated Circuit
DPA	Differential Power Analysis
DSP	Digital Signal Processing
ECC	Elliptic Curve Cryptography
ECDLP	Elliptic Curve Discrete Logarithm Problem
ECDSA	Elliptic Curve Digital Signature Algorithm
EdDSA	Edwards curve Digital Signature Algorithm
ECPM	Elliptic Curve Point Multiplication
ECSM	Elliptic Curve Scalar Multiplication
FPGA	Field Programmable Gate Array
GF	Galois Field
LUT	Look Up Table
NAF	Non-Adjacent Form
NIST	National Institute of Standards and Technology
PA	Point Addition
PD	Point Doubling
PKC	Public Key Cryptography
RNS	Residue Number System
RSD	Redundant Signed Digit
SCAs	Side-Channel Attacks
SPA	Simple Power Analysis

B. ARITHMETIC ON TWISTED EDWARDS CURVE

The addition of the affine points $A\,(x_1, y_1)$ and $B\,(x_2, y_2)$ on the curve $e_{a,d}$ is given by the formula [14]

$$A\,(x_1, y_1) + B\,(x_2, y_2) = R\,(x_3, y_3)$$

where

$$\begin{aligned} x_3 &= \frac{x_1y_2 + y_1x_2}{1 + dx_1x_2y_1y_2}, \\ y_3 &= \frac{y_1y_2 - ax_1x_2}{1 - dx_1x_2y_1y_2} \end{aligned} \tag{3}$$

The doubling of the affine point $A\,(x_1, y_1)$ on the curve $e_{a,d}$ is given by the formula [14]

$$2A\,(x_1, y_1) = R\,(x_2, y_2)$$

where

$$\begin{aligned} x_2 &= \frac{2x_1y_1}{y_1^2 + ax_1^2}, \\ y_2 &= \frac{y_1^2 - ax_1^2}{2 - y_1^2 - ax_1^2} \end{aligned} \tag{4}$$

C. GROUP OPERATIONS IN PROJECTIVE COORDINATES

Projective or Jacobian coordinates are used to avoid the most expensive modular inversion operation, which is essentially used in affine coordinate systems. In a projective coordinate system, each point (x, y) on the curve $e_{a,d}$ is represented in a triplet form (X, Y, Z) that corresponds to the affine point $(x = X/Z, y = Y/Z)$ with $Z \neq 0$.

The affine point (x, y) can be transformed to the projective point (X, Y, Z) as

$$X = x, \quad Y = y, \; Z = 1 \tag{5}$$

The projective point (X, Y, Z) can be transformed to the affine point (x, y) as

$$(x = X/Z, y = Y/Z) \tag{6}$$

The projective form [14] of the curve $e_{a,d}$ is given by the equation

$$E_{a,d} : (aX^2 + Y^2)Z^2 = Z^4 + dX^2Y^2 \tag{7}$$

The projective form of the curve e_d is given by the equation

$$E_d : (-X^2 + Y^2)Z^2 = Z^4 + dX^2Y^2 \tag{8}$$

The point addition can be performed on the curve E_d as

$$A(X_1, Y_1, Z_1) + B(X_2, Y_2, Z_2) = R(X_3, Y_3, Z_3)$$

where

$$\begin{aligned} X_3 &= Z_1Z_2(Z_1^2Z_2^2 - dX_1X_2Y_1Y_2)(X_1Y_2 + Y_1X_2), \\ Y_3 &= Z_1Z_2(Z_1^2Z_2^2 + dX_1X_2Y_1Y_2)(X_1X_2 + Y_1Y_2), \\ Z_3 &= (Z_1^2Z_2^2 + dX_1X_2Y_1Y_2)(Z_1^2Z_2^2 - dX_1X_2Y_1Y_2) \end{aligned} \tag{9}$$

The point doubling can be performed on the curve E_d by

$$2A(X_1, Y_1, Z_1) = R(X_2, Y_2, Z_2)$$

where

$$\begin{aligned} X_2 &= (2X_1Y_1)(Y_1^2 - X_1^2 - 2Z_1^2), \\ Y_2 &= (X_1^2 - Y_1^2)(X_1^2 + Y_1^2), \\ Z_2 &= (Y_1^2 - X_1^2)(Y_1^2 - X_1^2 - 2Z_1^2) \end{aligned} \tag{10}$$

III. PROPOSED HARDWARE ARCHITECTURES

This section presents all algorithms and proposed hardware architectures for modular multiplication, point addition (PA), point doubling (PD), and ECPM.

A. MODULAR MULTIPLICATION

Modular multiplication is one of the most time-consuming arithmetic operations of an ECC processor over a prime field on which the efficiency of an ECPM scheme entirely depends. Although higher radix modular multipliers offer fewer clock cycles as well as less computation time to perform modular multiplication, these require more hardware resources that increase occupied area. To minimize the area required for ECPM, a radix-2 interleaved modular multiplier is adopted to implement the ECPM scheme, which requires $n + 1$ clock cycles to perform modular multiplication of two n-bit integers. The modular multiplication of the two n-bit integers A and B over the prime field GF(p) can be defined as

$$\begin{aligned} C &= A \cdot B \bmod p \\ &= \{b_0 \cdot A + b_1 \cdot (2^1A) + b_2 \cdot (2^2A) + \ldots + b_{n-1} \\ &\quad \cdot (2^{n-1}A)\} \bmod p \\ &= \{\sum_{i=0}^{n-1} b_i \cdot (2^iA)\} \bmod p \end{aligned} \tag{11}$$

Algorithm 1 Radix-2 Interleaved Modular Multiplication

```
Input: $A = \sum_{i=0}^{n-1} a_i 2^i, B = \sum_{i=0}^{n-1} b_i 2^i, p = \sum_{i=0}^{n-1} p_i 2^i;$
       $a_i, b_i, p_i \in \{0, 1\}$
Output: $C = (A \cdot B) \bmod p$;
$C \leftarrow 0$;
$T \leftarrow B\&\text{'1'}$;
while $T(n-1 \text{ downto } 0) \neq 0$ loop
    $C \leftarrow 2C$;
    if $T_n = 1$ then //$n^{\text{th}}$ bit of $T$
        $C \leftarrow C + A$;
    end if;
    $C \leftarrow C \bmod p$;
    T $\leftarrow$ T($n-1$downto0)&'0';// left-shift operation
end loop;
return $C$;
```

where

$$A = (\sum_{i=0}^{n-1} a_i 2^i)_{10} \text{ (multiplicand)},$$

$$B = (\sum_{i=0}^{n-1} b_i 2^i)_{10} \text{ (multiplier)},$$

$$p = (\sum_{i=0}^{n-1} p_i 2^i)_{10} \text{ (prime number)}; \quad a_i, b_i, p_i \in \{0, 1\}.$$

An efficient algorithm based on iterative addition of partial product is proposed for the modular multiplication as shown in Algorithm 1. Figure 2 depicts the proposed modular multiplier over GF(p) based on this algorithm. In this method, accumulator C is doubled at the beginning of each iteration to perform iterative addition of the successive partial products. A shift-left register is used to perform synthesizable loop operation for the left to right bitwise multiplication. To determine the appropriate end of the loop, a temporary variable T of $n + 1$ bits is used in which $T(n \text{ downto } 1)$ is precomputed as the multiplier B and the least significant bit (LSB) of T is precomputed as 1. One extra bit is added at the LSB to cope with the completion of the left-shift operation in the case of $b_0 = 0$. The multiplicand A is added to the accumulator in each iteration if the most significant bit (MSB) of T is 1. The content of the accumulator is reduced to modulo p after each addition. To perform this modular operation, C is subtracted by the prime numbers p and $2p$. As the content of the accumulator is always less than $3p$, subtractions by p and $2p$ are enough to confine the content below the value of p. The subtractions $C - p$ and $C - 2p$ are performed by adding the 2's compliment of the subtrahends p and $2p$ to the minuend C. The comparisons $C \geq p$ and $C \geq 2p$ are performed by checking the sign bits of the differences $C - p$ and $C - 2p$, respectively. At the end of each iteration, T is shifted to the left by one bit. After n number of iterations, $T(n-1 \text{ downto } 0)$ is shifted to zero value and the content of the accumulator is stored in register 'Reg C', which is the final modular product of the integers A and B. The module comprises two multiplexers, in which MUX1 is used to keep the content of the accumulator unchanged if $T_n = 0$ or add A to the accumulator if $T_n = 1$ and MUX2 is used for performing $C \bmod p$. In the proposed architecture, a total of $n + 1$ clock cycles (CC) are required to perform the modular multiplication, where n clock cycles are for n number of iterations and one extra clock cycle is to store the final result in the register. Modular squaring can be performed by taking the inputs of the proposed modular multiplier identical such as (A, A) instead of (A, B).

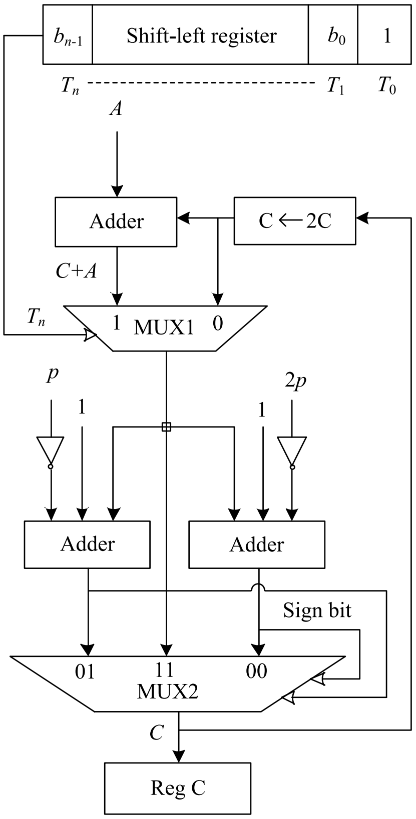
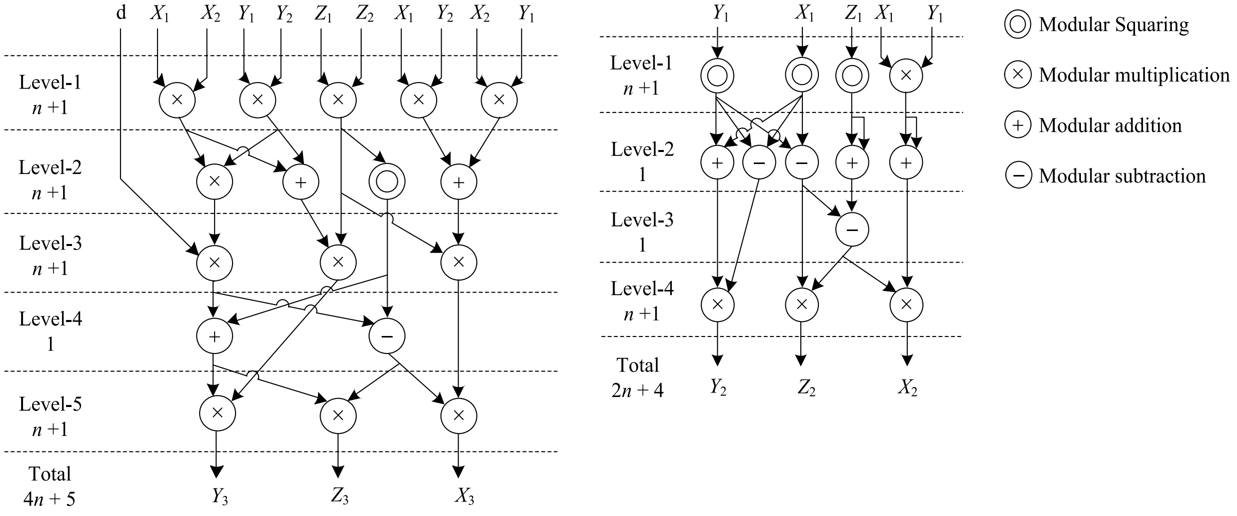


FIGURE 2. Proposed modular multiplier.

B. ELLIPTIC CURVE GROUP OPERATIONS

Elliptic curve group operations include several arithmetic modules such as modular adder, subtractor, multiplier, and squarer that belong to several successive levels for sequential data flow to perform ECPM. The PA and PD architectures are designed in projective coordinates. Figure 3(a) depicts the hardware design for PA based on (9), which has five consecutive levels that cost twelve **m**ultiplications, one **s**quaring, three **a**dditions, and one subtraction denoted as (12M+1S+4A). Figure 3(b) illustrates the hardware design for PD based on (10), which has four consecutive levels that cost four **m**ultiplications, three **s**quaring, three **a**dditions, and three subtractions denoted as (4M+3S+6A). The point addition and point doubling formulas for the projective



(a)

(b)

FIGURE 3. Proposed hardware architectures for (a) PA and (b) PD.

twisted Edwards curve reported in [14], [15] are modified to minimize the number of arithmetic modules as well as the hardware resource requirements. To obtain the shortest data path and the optimal latency, the architectures are efficiently balanced and the arithmetic operations are horizontally parallelized among the levels. Each multiplication and squaring requires $n + 1$ clock cycles, each addition and subtraction requires one clock cycle to be completed, where n is the number of bits under operation. The level that contains one or more than one squaring or multiplication takes $n+1$ clock cycles and the level that contains no multiplication or squaring takes only one clock cycle to jump to the next level. In this way, the latencies across the proposed PA and PD architectures are $4n+5$ and $2n+4$ clock cycles, respectively.

C. ELLIPTIC CURVE POINT MULTIPLICATION

ECPM is the pivotal operation of an ECC processor, which is computationally the most expensive. The underlying operation of ECPM can be defined as $Q = k \cdot P$, where P is a base point on the curve E_d, k is a scalar, which is the secret key, and Q is another point on the curve, which is the public key. Q can be obtained by adding P to itself $k - 1$ times or doubling P on itself $\log_2 k$ times if k is even. The point multiplication can be performed as a sequence of point addition and point doubling following the binary bit pattern of k. The simplest and easiest way to perform ECPM is the double-and-add method [1], as shown in Algorithm 2, in which point doubling is performed in every iteration, whereas point addition is performed only when $k_i = 1$. There are two timing and power consumption profiles in this method: one is only point doubling and the other is point addition following point doubling. Tracing the power consumption profiles by simple power analysis (SPA) [18], the binary bit pattern of the secret key can be easily retrieved as shown in Figure 4; therefore, this method is vulnerable to SCAs.

Algorithm 2 Double-and-Add Point Multiplication (Left to Right)

```
Input: $P$, $k = (\sum_{i=0}^{l-1} k_i 2^i)_{10}$; $k_i \in \{0, 1\}$, $k_{l-1} = 1$
Output: $Q$
$Q \leftarrow P$;
for $i$ from $l - 2$ downto 0 do
    $Q \leftarrow 2Q$;     // point doubling
    if $k_i = 1$ then
        $Q \leftarrow Q + P$;     // point addition
    end if;
end for;
return $Q$;
```

The Montgomery ladder algorithm [19] is used for the proposed ECPM scheme as shown in Algorithm 3, in which point addition and point doubling are performed simultaneously to make the secret key uncertain. Figure 5 justifies the SCAs resistance of the Montgomery algorithm based ECPM by showing its power tracing profile. The initial power consumption in each iteration is the total power consumed by both PA and PD modules because of their parallel operations. As the latency of point addition is higher than that of point doubling, point doubling is completed before point addition. After the completion of point doubling operation, power is only consumed by the PA module. There is no possible way to guess the bit pattern of the secret key by tracing the identical power pattern. Figure 6 illustrates the proposed hardware design for ECPM based on Algorithm 3, which utilizes a sequential combination of the PA and PD modules. The initial inputs of the PA module are computed as P and $2P$. The precomputation of the PD module depends on the $(l - 2)^{\text{th}}$ bit of k, where l is the bit length of k. MUX1 is used to

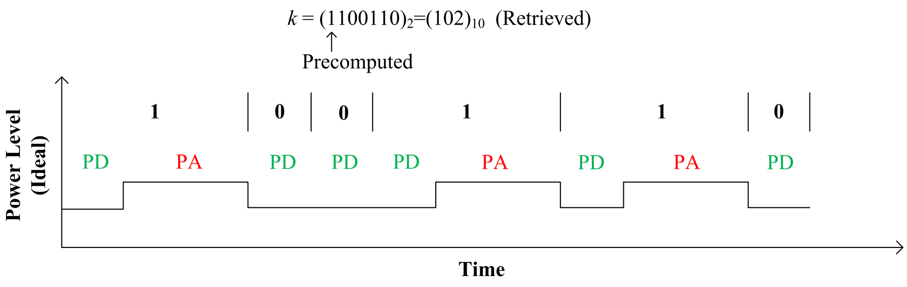


FIGURE 4. Power tracing profile of DAA algorithm based ECPM.

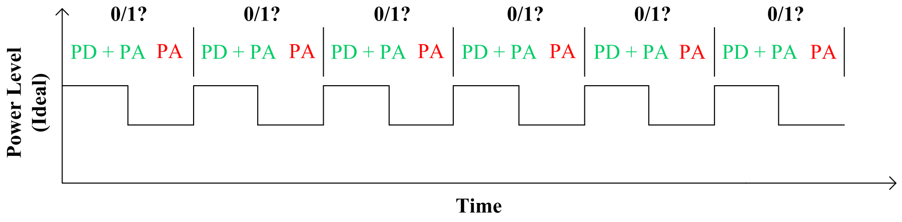


FIGURE 5. Power tracing profile of Montgomery algorithm based ECPM.

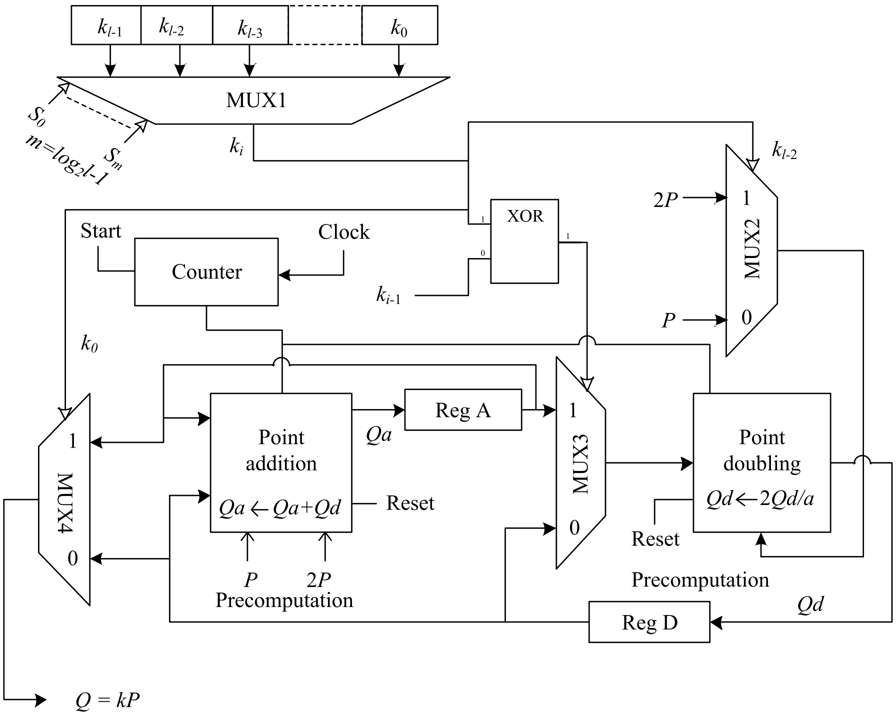


FIGURE 6. Proposed hardware architecture for ECPM.

select i^{th} bit of k among k_0 to k_{l-1} by $\log_2 l$ number of select lines. MUX2 selects the initial input of the PD module as P if $k_{l-2} = 0$ or $2P$ if $k_{l-2} = 1$. An XOR gate is used to perform $k_i \oplus k_{i-1}$ that determines when to change the input of the PD module, where k_i is the current operating bit of k and k_{i-1} is the upcoming bit of k in the left to right point multiplication. If the output of the XOR gate is low, no change of state occurs and the output of the PD module goes to its input via a feedback loop. Else if the output of the XOR gate is high, change of state occurs and the output of the PA module goes to the input of the PD module.

The input selection process of the PD module is operated by MUX3. In both cases, one of the two inputs of the PA module is its own output via a feedback loop and the other is the output of the PD module. 'Reg A' and 'Reg D' are used to store the intermediate outputs of the PA and PD modules, respectively. After $l - 1$ number of iterations, MUX4 selects the output of the PA module if $k_0 = 1$ or the output of the PD module if $k_0 = 0$ as the final result. Since the point addition and point doubling are performed simultaneously and the PA module requires more clock cycles than the PD module to complete its operation, the number of iterations, as well as

TABLE 2. Implementation results of the proposed ECC modules on different FPGA platforms over $\mathbb{F}_p$-256.

Operation	Platform	Clock cycles	Number of slices	Number of LUTs	Maximum frequency (MHz)	Time (μs)	Throughput (Mbps)
Modular multiplication	Virtex-7	257	427	1311	177.7	1.45	177.01
	Virtex-6	257	414	1310	161.1	1.60	160.47
Point addition	Virtex-7	1029	4459	15619	177.7	5.79	44.21
	Virtex-6	1029	4339	15489	161.1	6.39	40.08
Point doubling	Virtex-7	516	1801	6687	177.7	2.90	88.16
	Virtex-6	516	1990	6958	161.1	3.20	79.93
ECPM	Virtex-7	262650	8873	32781	177.7	1.48 ms	173.20 kbps
	Virtex-6	262650	9246	33238	161.1	1.63 ms	157.02 kbps

Algorithm 3 Montgomery Ladder Point Multiplication (Left to Right)

```
Input: $P, k = (\sum_{i=0}^{l-1} k_i 2^i)_{10}; k_i \in \{0, 1\}, k_{l-1} = 1$
Output: $Q$
$Q_1 \leftarrow P; Q_2 \leftarrow 2P;$
for $i$ from $l - 2$ downto 0 do
    if $k_i = 1$ then
        $Q_1 \leftarrow Q_1 + Q_2;$   // point addition
        $Q_2 \leftarrow 2Q_2;$   // point doubling
    else
        $Q_2 \leftarrow Q_1 + Q_2;$ //   point addition
        $Q_1 \leftarrow 2Q_1;$   // point doubling
    end if;
end for;
return $Q_1$;
```

the latency of the ECPM, depends on the PA module. Thus, the latency of the ECPM can be calculated as

$$\begin{aligned} ECPM_{cc} &= (l-1) \times PA_{cc} + (l-1) \times Rg_{cc} \\ &= (l-1) \times (4n+5) + (l-1) \times 1 \\ &= (l-1) \times (4n+6) \end{aligned} \tag{12}$$

For $l = n$,

$$\begin{aligned} ECPM_{cc} &= (n-1) \times (4n+6) \\ &= 4n^2 + 2n - 6 \end{aligned} \tag{13}$$

where

PA_{cc} = clock cycles required to perform point addition,

Rg_{cc} = clock cycle required to store PA output in 'Reg A'.

In the double-and-add method demonstrated in Algorithm 2, the average clock cycles required for the ECPM can be calculated as

$$\begin{aligned} ECPM_{cc} &= (l-1) \times PD_{cc} + (l/2) \times PA_{cc} + (l-1) \\ &= (l-1) \times (2n+4) + (l/2) \times (4n+5) + (l-1) \end{aligned} \tag{14}$$

For $l = n$,

$$\begin{aligned} ECPM_{cc} &= (n-1) \times (2n+4) + (n/2) \times (4n+5) + (n-1) \\ &= 4n^2 + 5.5n - 5 \end{aligned} \tag{15}$$

The same ECPM design can be implemented with fewer clock cycles by the Montgomery ladder method than the double-and-add method, which makes the ECC processor faster. We take the advantages of high-speed computation and high resistance against probable SCAs offered by the Montgomery ladder algorithm for our ECPM module.

IV. IMPLEMENTATION AND SIMULATION RESULTS

The proposed ECC processor is implemented using the Xilinx ISE 14.7 Design Suite software and simulated by the Xilinx ISim simulator. The simulation results are verified by the Maple software. All the designs are synthesized, mapped, placed, and routed on the Xilinx Virtex-7 (XC7VX690T) and Virtex-6 (XC6VHX380T) FPGA platforms, separately. The design goal is set to "Balanced" and the design strategies are set to the default values. The implementation results of the proposed ECC modules over a prime field of 256 bits are summarized in Table 2. On the Virtex-7 FPGA, the proposed modular multiplier, PA, PD, and ECPM modules run at a maximum frequency of 177.7 MHz. The latencies of the multiplier, PA, PD, and ECPM modules are 257, 1029, 516, and 262,650 clock cycles, respectively. To perform modular multiplication, the multiplier takes 1.45 μs with 177 Mbps throughput on the Virtex-7 FPGA, occupying 427 slices equivalent to 1311 look up tables (LUTs). On the same platform, the PA module takes 5.79 μs with a throughput of 44.21 Mbps to add two points on the curve E_d, consuming 4459 slices equivalent to 15,619 LUTs. The PD module occupies 1801 slices equivalent to 6687 LUTs and takes 2.90 μs with 88.16 Mbps throughput for doubling a point on the curve. The final ECPM scheme combines both the PA and PD modules, utilizing 8873 slices equivalent to 32,781 LUTs. It requires 1.48 ms with a throughput of 173.2 kbps to perform single point multiplication for a 256-bit key.

On the Virtex-6 FPGA, all the modules run at a maximum frequency of 161.1 MHz. The multiplier, PA, and PD modules take 1.60, 6.39, and 3.20 μs with 160.47, 40.08,

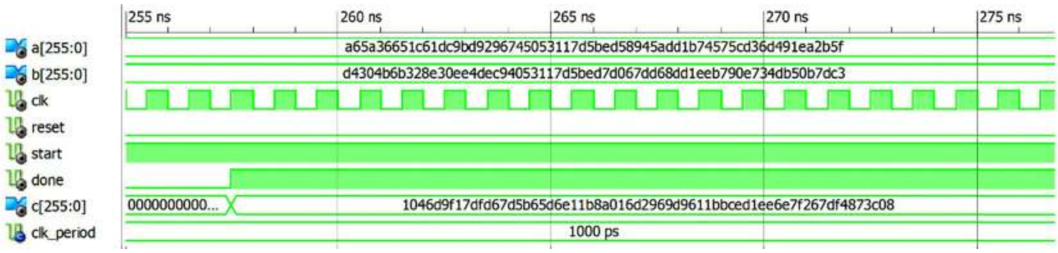

255 ns
260 ns
265 ns
270 ns
275 ns
a[255:0]
a65a36651c61dc9bd9296745053117d5bed58945add1b74575cd36d491ea2b5f
b[255:0]
d4304b6b328e30ee4dec94053117d5bed7d067dd68dd1eeb790e734db50b7dc3
clk
reset
start
done
c[255:0]
0000000000...
1046d9f17dfd67d5b65d6e11b8a016d2969d9611bbced1ee6e7f267df4873c08
clk_period
1000 ps

FIGURE 7. Simulation result of modular multiplication.

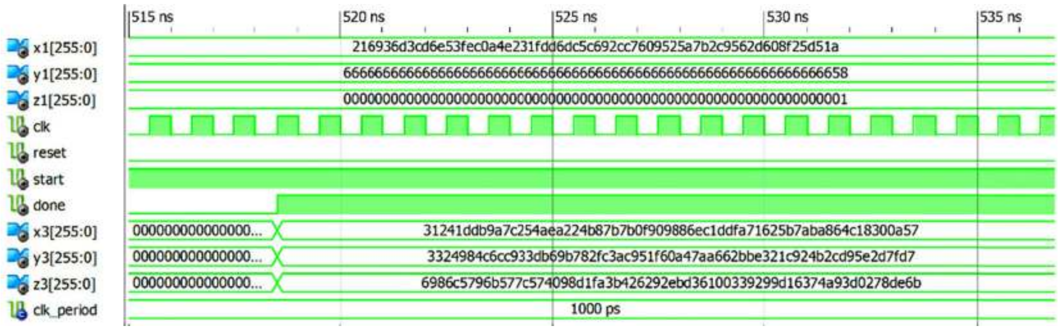


FIGURE 8. Simulation result of point doubling.

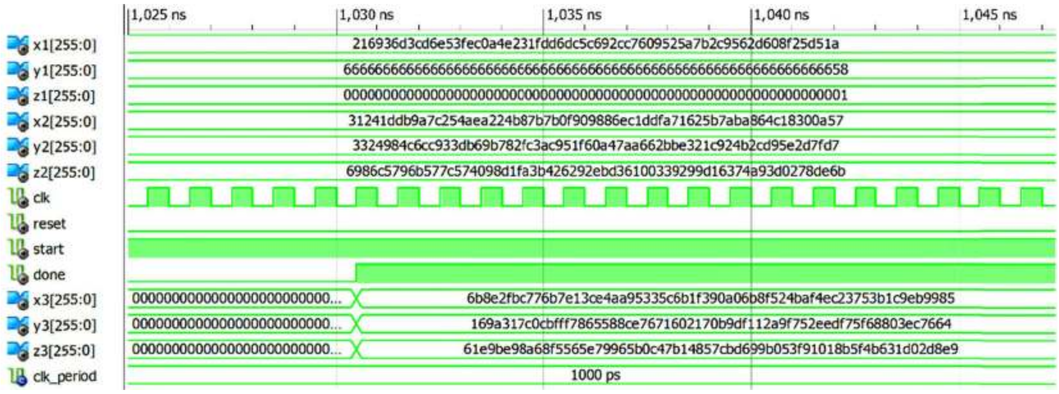

1,025 ns
1,030 ns
1,035 ns
1,040 ns
1,045 ns
x1[255:0]
216936d3cd6e53fec0a4e231fdd6dc5c692cc7609525a7b2c9562d608f25d51a
y1[255:0]
6666666666666666666666666666666666666666666666666666666666666658
z1[255:0]
0000000000000000000000000000000000000000000000000000000000000001
x2[255:0]
31241ddb9a7c254aea224b87b7b0f909886ec1ddfa71625b7aba864c18300a57
y2[255:0]
3324984c6cc933db69b782fc3ac951f60a47aa662bbe321c924b2cd95e2d7fd7
z2[255:0]
6986c5796b577c574098d1fa3b426292ebd36100339299d16374a93d0278de6b
clk
reset
start
done
x3[255:0]
000000000000000000000000...
6b8e2fbc776b7e13ce4aa95335c6b1f390a06b8f524baf4ec23753b1c9eb9985
y3[255:0]
000000000000000000000000...
169a317c0cbfff7865588ce7671602170b9df112a9f752eedf75f68803ec7664
z3[255:0]
000000000000000000000000...
61e9be98a68f5565e79965b0c47b14857cbd699b053f91018b5f4b631d02d8e9
clk_period
1000 ps

FIGURE 9. Simulation result of point addition.

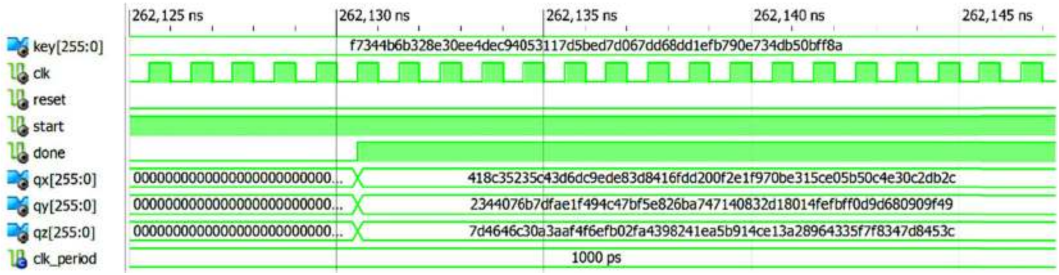

262,125 ns
262,130 ns
262,135 ns
262,140 ns
262,145 ns
key[255:0]
f7344b6b328e30ee4dec94053117d5bed7d067dd68dd1efb790e734db50bff8a
clk
reset
start
done
qx[255:0]
000000000000000000000000...
418c35235c43d6dc9ede83d8416fdd200f2e1f970be315ce05b50c4e30c2db2c
qy[255:0]
000000000000000000000000...
2344076b7dfae1f494c47bf5e826ba747140832d18014fefbff0d9d680909f49
qz[255:0]
000000000000000000000000...
7d4646c30a3aaf4f6efb02fa4398241ea5b914ce13a28964335f7f8347d8453c
clk_period
1000 ps

FIGURE 10. Simulation result of ECPM.

and 79.93 Mbps throughputs, occupying 414, 4339, and 1990 slices, respectively, to perform. To complete single point multiplication, the ECPM scheme spends 1.63 ms with a throughput of 157 kbps, utilizing 9246 slices on this platform. The same designs show slightly worse performance in terms of speed and area utilization on the Virtex-6 FPGA than on the Virtex-7 FPGA.

Figure 7 shows the simulation result of the modular multiplication. 'a' and 'b' are the inputs and 'c' is the output. The multiplication takes 257.5 clock cycles to be completed, where 1 clock cycle = 1ns and offset delay = 0.511ns. The simulation result of the point doubling operation on the curve E_d is shown in Figure 8, where 518.5 clock cycles are required to double the point (x1, y1, z1) that makes

TABLE 3. Performance comparison of the proposed ECPM module with other designs over $\mathbb{F}_p$-256.

Work	Publishing year	Platform	Reported area (slices)	Clock cycles (k)	Maximum frequency (MHz)	Time (ms)	Area × Time (AT)[a]	Throughput (kbps)
This work (a)	–	Virtex-7	8.9k	262.7	177.7	1.48	13.17	173.20[b]
This work (b)	–	Virtex-6	9.2k	262.7	161.1	1.63	15.00	157.00[b]
Shah *et al.* [37]	2018	Virtex-6	65.6k	153.2	327.0	0.47	30.83	546.42[b]
Hu *et al.* [43]	2018	Virtex-4	9.4k + 14 DSPs	610	20.44	29.84	280.50	8.58[b]
Hossain *et al.* [42]	2017	Kintex-7	11.3k	397.3	121.5	3.27	36.95	78.28
Asif *et al.* [26]	2017	Virtex-7	24.2k	215.9	72.9	2.96	71.63	1816.20
Liu *et al.* [31]	2017	Virtex-4	12k	459.9	36.5	12.60	151.20	20.32[b]
Javeed *et al.* [41]	2017	Virtex-4	20.6k	191.6	49.0	3.91	80.55	65.47
Javeed *et al.* [40]	2016	Virtex-4	13.2k	200	40.0	5.00	66.00	51.00
Javeed *et al.* [39]	2016	Virtex-4	35.7k	207.1	70.0	2.96	105.67	86.53[b]
Marzouqi *et al.* [36]	2016	Virtex-5	8.7k	361.6	160.0	2.26	19.66	113.27[b]
Loi *et al.* [30]	2015	Virtex-4	7k + 8 DSPs	993.7	182.0	5.46	38.22	46.88[b]
Lee *et al.* [38]	2014	Virtex-II Pro	8.3k	163.2	37.0	4.41	36.60	58.04[b]
Marzouqi *et al.* [35]	2013	Virtex-5	10.2k	442.2	66.7	6.63	67.63	38.61[b]
Ghosh *et al.* [34]	2011	Virtex-II-Pro	12k	337.7	36.0	9.38	112.56	27.29[b]
Ghosh *et al.* [32]	2009	Virtex-4	20.1k	331.1	43.0	7.70	154.77	33.25[b]
Ananyi *et al.* [29]	2009	Virtex-4	20.8k + 32 DSPs	414	60.0	6.10	126.08	37.10[b]
Schinianakis *et al.* [24]	2009	Virtex-E	16.4k	156.8	39.7	3.95	64.78	64.82[b]
McIvor *et al.* [22]	2006	Virtex-II Pro	15.8k + 256 DSPs	151.4	39.5	3.86	60.98	66.74

[a] Considered as unitless value for performance comparison, where A is area (kilo slices) and T is time (ms).

[b] Estimated by the authors of this paper as Throughput = (Maximum frequency ÷ Clock cycles) × 256 .

another point (x2, y2, z2) on the curve. Few extra clock cycles than the estimated value are spent because of some offset delays. Figure 9 shows the simulation result of the point addition operation, where (x1, y1, z1), (x2, y2, z2) are the additive points and (x3, y3, z3) is the resultant point. The addition takes 1030.5 clock cycles. The simulation result of the ECPM is shown in Figure 10. It spends 262.130k clock cycles, which is less than the estimated value. The reason behind this reduction in clock cycles required is that the ECPM deals with a number of modular multiplication operations in its interim states. If the multiplier of any interim multiplication is smaller than 256 bits in size, it requires less than 129 clock cycles to be completed. This results in a reduction in the latency of ECPM.

V. PERFORMANCE COMPARISION WITH SIMILAR WORKS

A performance comparison of our proposed ECPM design with other available designs for ECPM over $\mathbb{F}_p$-256 is presented in Table 3. Shah *et al.* [37] proposed an ECC processor, adopting an RSD representation for carry free arithmetic that provides high throughput for ECPM. This processor occupies 65.6K slices on Virtex-6 FPGA and takes 0.47 ms with a throughput of 546.42 kbps to perform point multiplication on the NIST recommended prime curve P-256. Although the processor is faster, it consumes more slices and is less efficient than our processor in terms of AT product. Their processor offers high-speed computation, costing more hardware resources, which is not suitable for resource-constrained devices. The processor reported in [43] is reconfigurable for different field orders and safe from SPA attacks. It takes 1066 clock cycles for point addition, 1325 clock cycles for point doubling, and 610k clock cycles for point multiplication, whereas our processor requires 1029, 516, and 262.7k clock cycles for the point addition, point doubling, and point multiplication, respectively. This processor consumes 9.4k slices with additional 14 DSP slices on Virtex-4 FPGA and requires 29.84 ms with 8.58 kbps throughput to perform ECPM. Our processor is faster and provides higher throughput than the processor. Hossain *et al.* [42] proposed a high-performance ECC processor, providing both ASIC and FPGA implementations. Their processor utilizes 11.3k slices on Kintex-7 FPGA and takes 3.97 ms with a throughput of 78.28 kbps to perform ECPM. The processor consumes 1.3 times more slices and is 2.2 times slower than our processor implemented on the same series FPGA. The throughput of our processor is also higher than that of their processor. Moreover, their processor does not provide any protection against SCAs

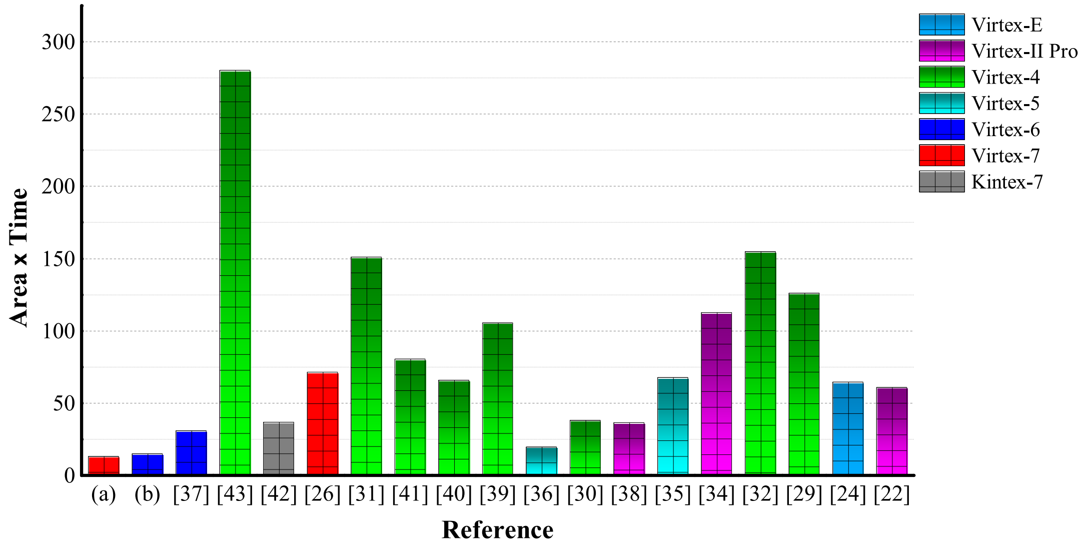


FIGURE 11. Performance comparison in terms of AT product.

as it is based on the double-and-add algorithm in which point addition and point doubling are performed sequentially. Asif *et al.* [26] proposed an RNS-based ECC processor that performs ECPM on 21 keys simultaneously. Although their ECPM module provides 10.5 times higher throughput, it expends 2.7 times more slices and is exactly two times slower than our ECPM module implemented on the same FPGA. The major contribution of their work is to perform ECPM on 21 keys simultaneously providing high throughput (1816.20 kbps) by taking the advantages of RNS-based implementation in which the arithmetic modules are divided into several groups and processed one by one for fast arithmetic operations.

Liu *et al.* [31] presented a duel-field ECC processor, adopting a modified radix-4 interleaved modular multiplication that consumes 12k slices on Virtex-4 FPGA and takes 12.60 ms to perform ECPM over either a prime field or a binary field of 256 bits. They provided both ASIC and FPGA implementations over duel-field showing power analysis attacks resistance. Compared with their processor, our processor is faster and provides higher throughput. In [39]–[41], Javeed *et al.* proposed ECC processors over prime fields that occupy 20.6k, 13.2k, and 35.7k slices on Virtex-4 FPGA and take 3.91, 5, and 2.96 ms, respectively, to perform ECPM. Our processor is more efficient in terms of AT product and provides higher throughput than these processors. A high-speed RSD-based ECC processor was proposed by Marzouqi *et al.* [36]. They adopted a Karatsuba multiplier for modular multiplication in which each of the two n-bit operands is split into two $n/2$-bit segments and multiplications are performed with the four $n/2$-bit segments recursively to compute the product. The processor consumes 8.7k slices on Virtex-5 FPGA and requires 2.26 ms with a throughput of 113.27 kbps. It offers significantly improved performance for using RSD-based modular arithmetic in which addition and subtraction can be performed without representing 2's complement. The processor reported in [35] is almost the same as the processor reported in [36], but it shows little bit worse performance on the same FPGA. A scalable ECC processor was developed by Loi *et al.* [30], which supports all five NIST recommended prime curves without reconfiguring the hardware. This processor occupies 7k slice with additional 8 DSP slices and 2 BRAM on Virtex-4 FPGA. It takes 5.46 ms in 993.7 clock cycles with a throughput of 46.88 kbps. Our processor is faster and better in terms of both AT product and throughput than the processor.

In [32], Ghosh *et al.* proposed an SCA-resistant ECC processor based on the double-and-add-always method, which consumes 20.1k slices on Virtex-4 FPGA and requires 7.70 ms with 33.25 kbps throughput to perform ECPM. In particular, they focused on side-channel security but did not pay enough attention to the processor speed. A flexible hardware ECC processor was proposed by Ananyi *et al.* [29] that can be used for all five NIST recommended prime curves without reconfiguring the hardware. They used both the binary and NAF algorithm [1] for scalar multiplication individually along with a regular multiplier with fast modular reduction replacing the conventional Montgomery modular multiplication. On Virtex-4 FPGA, their processor utilizes 20.8k slices with additional 32 DSP slices and takes 6.1 ms/6.90 ms to perform ECPM by the NAF/binary method. The processors reported in [22], [24], [34], [38] have higher AT products and lower throughputs than our processor. However, these processors are implemented on some backdated FPGAs, which are now obsolete. AT product can be a measure of the efficiency of an ECC processor as there is a trade-off between time and area. A lower value of AT product ensures better performance of the processor. Figure 11 shows the performance comparison of our ECPM design with the other designs tabulated in Table 3 in terms of AT product. The AT product of our design is comparatively low, which guarantees our design as more efficient for

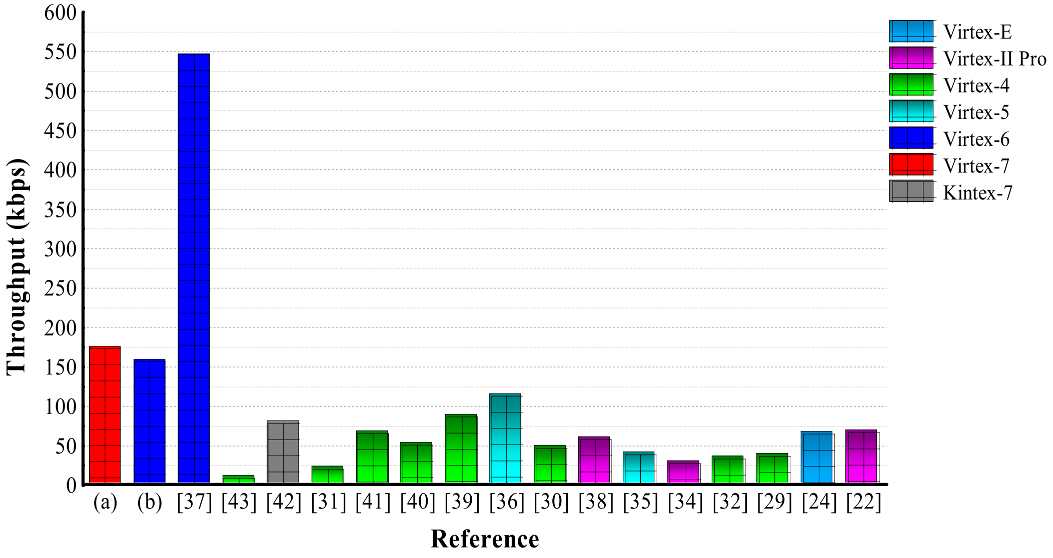


FIGURE 12. Performance comparison in terms of throughput.

IoT applications. Figure 12 shows the performance comparison in terms of throughput, which is another measure of the efficiency of an ECC processor. The processor reported in [26] provides the highest throughput among the processors enlisted in Table 3 as it operates 21 keys simultaneously. Owing to the processor's high relative value, its throughput is not shown in the same chart. The throughput of our processor is higher than that of the other designs, except the throughputs reported in [26] and [37]. It is worth noting that the updated FPGAs are manipulated to implement our design. The significant improvements in area and delay ensure better performance of our design. However, due to the implementations of the described processors on different FPGA platforms, a fair comparison is not possible. The earlier FPGAs such as Virtex-5, Virtex-4, Virtex-II-Pro, and Virtex-E are omitted to implement our proposed ECC processor because of their high power consumption and having less number of input/output blocks (IOBs).

VI. CONCLUSION

In this paper, a high-speed, area-efficient, SCA-resistant ECC processor is developed for fast point multiplication exploiting Edwards25519 curve with its projective representation. A radix-2 interleaved modular multiplier is adopted for modular multiplication that requires $n + 1$ clock cycles to multiply two n-bit integers. Novel hardware architectures for point addition and point doubling are proposed that require $4n + 5$ and $2n + 4$ clock cycles, respectively, to accomplish n-bit operations. The Montgomery scalar multiplication algorithm is used to perform ECPM as it offers fast computation with high resistance against SCAs. All the designs are implemented on the Xilinx Virtex-7 and Virtex-6 FPGA platforms individually over a prime field of 256 bits. The processor performs single point multiplication in 262,650 clock cycles and it takes 1.48 ms with a throughput of 173.2 kbps, consuming 8,873 slices on the Virtex-7 FPGA. It offers higher efficiency in terms of area-delay product and throughput without degrading the security level. Based on the overall performance analyses, it can be concluded that the proposed ECC processor can be a good choice for high-speed data encryption as well as for the privacy and security of resource-constrained IoT devices.

APPENDIX

RESULTS VERIFICATION

The simulation results are verified by the Maple software as follows:

Prime:

$$\begin{aligned} p :=\ & 2^{255} - 19;\\ & 57896044618658097711785492504343953926634 99\\ & 2332820282019728792003956564819949 \end{aligned} \tag{16}$$

Modular multiplication

$$\begin{aligned} a :=\ & \mathit{convert}(\text{“}A65A36651C61DC9BD9296745053117D\\ & 5BED58945ADD1B74575CD36D491EA2B5F\text{”},\\ & \mathit{decimal}, \mathit{hex});\\ & 75243324520898617289173384717141375907599 22\\ & 5675997126742903002171622814264159 \end{aligned} \tag{17}$$

$$\begin{aligned} b :=\ & \mathit{convert}(\text{“D4304B6B328E30EE4DEC94053117D5B}\\ & \text{ED7D067DD68DD1EEB790E734DB50B7DC3”},\\ & \mathit{decimal}, \mathit{hex});\\ & 95975653079776248084467029675907631863850 86\\ & 6141462094048851336995200614235587 \end{aligned} \tag{18}$$

$$\begin{aligned} c :=\ & \mathit{convert}(a \cdot b \bmod p, \mathit{hex});\\ & \text{1046D9F17DFD67D5B65D6E11B8A016D2969D96}\\ & \text{11BBCED1EE6E7F267DF4873C08} \end{aligned} \tag{19}$$

Curve

$$E_d : (-X^2 + Y^2) \cdot Z^2 = Z^4 + d \cdot X^2 \cdot Y^2; \tag{20}$$

$$LHS := (-X^2 + Y^2) \cdot Z^2; \tag{21}$$

$$RHS := Z^4 + d \cdot X^2 \cdot Y^2; \tag{22}$$

$$d := -\frac{121665}{121666} \bmod p; \tag{23}$$

37095705934669439343138083508754565189542113879843219016388785533085940283555

Projective point

$$\begin{aligned} &P_1(X_1, Y_1, Z_1) : \\ X_1 &:= convert(\text{“}216936D3CD6E53FEC0A4E231FDD \\ &\quad 6DC5C692CC7609525A7B2C9562D608F25D \\ &\quad 51A\text{”}, decimal, hex); \\ &\quad 1511222134953540077250115140958853151145 \\ &\quad 40126930418572060461132839498477622 02 \end{aligned} \tag{24}$$

$$\begin{aligned} Y_1 &:= convert(\text{“}66666666666666666666666666666666 \\ &\quad 666666666666666666666666666666 58\text{”}, \\ &\quad decimal, hex); \\ &\quad 4631683569492647816942839400347516314130 \\ &\quad 799386625622561578303360316525185596 0 \end{aligned} \tag{25}$$

$$Z_1 = 1; \tag{26}$$

$$\begin{aligned} LHS &:= (-X_1^2 + Y_1^2) \cdot Z_1^2 \bmod p; \\ &\quad 1086587272079549330507312378769575411096 \\ &\quad 184320537239777352469104049270346277 7 \end{aligned} \tag{27}$$

$$\begin{aligned} RHS &:= (Z_1^4 + d \cdot X_1^2 \cdot Y_1^2) \bmod p; \\ &\quad 1086587272079549330507312378769575411096 \\ &\quad 184320537239777352469104049270346277 7 \end{aligned} \tag{28}$$

The point $P_1(X_1, Y_1, Z_1)$ is on the curve.

Point doubling

$$\begin{aligned} &P(X_2, Y_2, Z_2) \\ &\quad := 2P(X_1, Y_1, Z_1); \end{aligned} \tag{29}$$

$$\begin{aligned} X_2 &:= (2X_1 \cdot Y_1) \cdot (Y_1^2 - X_1^2 - 2Z_1^2) \bmod p; \\ &\quad 222271421460536153836867114565920545 33 \\ &\quad 4817230652383280794910861657546885719 91 \end{aligned} \tag{30}$$

$$\begin{aligned} Y_2 &:= (X_1^2 - Y_1^2) \cdot (X_1^2 + Y_1^2) \bmod p; \\ &\quad 2313261289793576394737611881630293696 19 \\ &\quad 457538555924972125273302060347140013 67 \end{aligned} \tag{31}$$

$$\begin{aligned} Z_2 &:= (Y_1^2 - X_1^2) \cdot (Y_1^2 - X_1^2 - 2Z_1^2) \bmod p; \\ &\quad 4773096952541154348632334549159460820 09 \\ &\quad 688224541037287466376761790956438073 39 \end{aligned} \tag{32}$$

$$\begin{aligned} X_2hex &:= convert(X_2, hex); \\ &\quad \text{31241DDB9A7C254AEA224B87B7B0F909886} \\ &\quad \text{EC1DDFA71625B7ABA864C18300A57} \end{aligned} \tag{33}$$

$$\begin{aligned} Y_2hex &:= convert(Y_2, hex); \\ &\quad \text{3324984C6CC933DB69B782FC3AC951F60A47} \\ &\quad \text{AA662BBE321C924B2CD95E2D7FD7} \end{aligned} \tag{34}$$

$$\begin{aligned} Z_2hex &:= convert(Z_2, hex); \\ &\quad \text{6986C5796B577C574098D1FA3B426292EBD3} \\ &\quad \text{6100339299D16374A93D0278DE6B} \end{aligned} \tag{35}$$

$$\begin{aligned} LHS &:= (-X_2^2 + Y_2^2) \cdot Z_2^2 \bmod p; \\ &\quad 1425782756415315217761864667414493026 3 \\ &\quad 2589814719953111577158698112228023986 58 \end{aligned} \tag{36}$$

$$\begin{aligned} RHS &:= (Z_2^4 + d \cdot X_2^2 \cdot Y_2^2) \bmod p; \\ &\quad 1425782756415315217761864667414493026 32 \\ &\quad 589814719953111577158698112228023986 58 \end{aligned} \tag{37}$$

The point $P_2(X_2, Y_2, Z_2)$ is on the curve.

Point addition

$$\begin{aligned} &P_3(X_3, Y_3, Z_3) \\ &\quad := P_1(X_1, Y_1, Z_1) + P_2(X_2, Y_2, Z_2); \end{aligned} \tag{38}$$

$$\begin{aligned} X_3 &:= (Z_1 \cdot Z_2) \cdot ((Z_1 \cdot Z_2)^2 - d \cdot X_1 \cdot X_2 \cdot Y_1 \cdot Y_2) \\ &\quad \cdot (X_1 \cdot Y_2 + Y_1 \cdot X_2) \bmod p; \\ &\quad 4864869654473268926554821774677038067 07 \\ &\quad 0085201546435440947825496286376411789 3 \end{aligned} \tag{39}$$

$$\begin{aligned} Y_3 &:= (Z_1 \cdot Z_2) \cdot ((Z_1 \cdot Z_2)^2 + d \cdot X_1 \cdot X_2 \cdot Y_1 \cdot Y_2) \\ &\quad \cdot (X_1 \cdot X_2 + Y_1 \cdot Y_2) \bmod p; \\ &\quad 1022331864675484206284714053678363566 32 \\ &\quad 4158518363803613133033983003910972784 4 \end{aligned} \tag{40}$$

$$\begin{aligned} Z_3 &:= ((Z_1 \cdot Z_2)^2 - d \cdot X_1 \cdot X_2 \cdot Y_1 \cdot Y_2) \cdot ((Z_1 \cdot Z_2)^2 - d \\ &\quad \cdot X_1 \cdot X_2 \cdot Y_1 \cdot Y_2) \bmod p; \\ &\quad 4428733712592879126932677269979317467 01 \\ &\quad 6925175774597184916161948196847136586 5 \end{aligned} \tag{41}$$

$$\begin{aligned} X_3hex &:= convert(X_3, hex); \\ &\quad \text{6B8E2FBC776B7E13CE4AA95335C6B1F390A} \\ &\quad \text{06B8F524BAF4EC23753B1C9EB9985} \end{aligned} \tag{42}$$

$$\begin{aligned} Y_3hex &:= convert(Y_3, hex);\\ &\quad \mathrm{169A317C0CBFFF7865588CE7671602170B9D}\\ &\quad \mathrm{F112A9F752EEDF75F68803EC7664} \end{aligned} \tag{43}$$

$$\begin{aligned} Z_3hex &:= convert(Z_3, hex);\\ &\quad \mathrm{61E9BE98A68F5565E79965B0C47B14857CBD}\\ &\quad \mathrm{699B053F91018B5F4B631D02D8E9} \end{aligned} \tag{44}$$

$$\begin{aligned} LHS &:= (-X_3^2 + Y_3^2) \cdot Z_3^2 \bmod p;\\ &\quad 522885066891130133978983003361202017972\\ &\quad 032330850822522703595274992221939124 00 \end{aligned} \tag{45}$$

$$\begin{aligned} RHS &:= (Z_3^4 + d \cdot X_3^2 \cdot Y_3^2) \bmod p;\\ &\quad 522885066891130133978983003361202017972\\ &\quad 03233085082252270359527499222193912400 \end{aligned} \tag{46}$$

The point $P_3(X_3, Y_3, Z_3)$ is on the curve.

ECPM:

$$\begin{aligned} &Q(X, Y, Z)\\ &\quad := k \cdot P_1(X_1, Y_1, Z_1); \end{aligned} \tag{47}$$

$$\begin{aligned} k &:= convert(\text{“F7344B6B328E30EE4DEC94053117D}\\ &\quad \text{5BED7D067DD68DD1EFB790E734DB50BF}\\ &\quad \text{F8A”}, decimal, hex);\\ &\quad 111813670168449685214851708472567153439 7\\ &\quad 28431793055142389748562888767010766730 \end{aligned} \tag{48}$$

$$\begin{aligned} X &:= convert(\text{“418C35235C43D6DC9EDE83D8416F}\\ &\quad \text{DD200F2E1F970BE315CE05B50C4E30C2}\\ &\quad \text{DB2C”}, decimal, hex);\\ &\quad 2964806049285244060500256493619101320810\\ &\quad 80923798794928380964242545412157509 56 \end{aligned} \tag{49}$$

$$\begin{aligned} Y &:= convert(\text{“2344076B7DFAE1F494C47BF5E826B}\\ &\quad \text{A747140832D18014FEFBFF0D9D680909F49”},\\ &\quad decimal, hex);\\ &\quad 1595114651102521823626575422170809986924\\ &\quad 81177321423494314140537705759768983 77 \end{aligned} \tag{50}$$

$$\begin{aligned} Z &:= convert(\text{“7D4646C30A3AAF4F6EFB02FA43982}\\ &\quad \text{41EA5B914CE13A28964335F7F8347D8453C”},\\ &\quad decimal, hex);\\ &\quad 5666327374795392532911297730501156658928\\ &\quad 50046300621879338380003881391200842 84 \end{aligned} \tag{51}$$

$$\begin{aligned} LHS &:= (-X^2 + Y^2) \cdot Z^2 \bmod p;\\ &\quad 347956437956115307516924899793218900498 6\\ &\quad 461105966284820630299876654272411369 6 \end{aligned} \tag{52}$$

$$\begin{aligned} RHS &:= (Z^4 + d \cdot X^2 \cdot X^2) \bmod p;\\ &\quad 3479564379561153075169248997932189004986\\ &\quad 4611059662848206302998766542724113696 \end{aligned} \tag{53}$$

The point $Q(X, Y, Z)$ is also on the curve.

REFERENCES

[1] D. Hankerson, A. Menezes, and S. Vanstone, *Guide to Elliptic Curve Cryptography*. New York, NY, USA: Springer-Verlag, 2004.

[2] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. Inf. Theory*, vol. 31, no. 4, pp. 469–472, Jul. 1985.

[3] 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 Trans. Depend. Sec. Comput.*, vol. 14, no. 3, pp. 237–248, Jun. 2017.

[4] S. Challa, M. Wazid, A. K. Das, N. Kumar, A. G. Reddy, E.-J. Yoon, and K.-Y. Yoo, "Secure signature-based authenticated key establishment scheme for future IoT applications," *IEEE Access*, vol. 5, pp. 3028–3043, 2017.

[5] C. A. Lara-Nino, A. Diaz-Perez, and M. Morales-Sandoval, "Elliptic curve lightweight cryptography: A survey," *IEEE Access*, vol. 6, pp. 72514–72550, 2018.

[6] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, Feb. 1978.

[7] N. Koblitz, "Elliptic curve cryptosystems," *Math. Comput.*, vol. 48, no. 177, pp. 203–209, 1987.

[8] V. S. Miller, "Uses of elliptic curves in cryptography," in *Advances in Cryptology* (Lecture Notes in Computer Science), vol. 218. New York, NY, USA: Springer-Verlag, 1986, pp. 417–426.

[9] H. M. Edward, "A normal form for elliptic curves," *Bull. Amer. Math. Soc.*, vol. 44, no. 3, pp. 393–422, 2007.

[10] D. J. Bernstein and T. Lange, "Faster addition and doubling on elliptic curves," in *Advances in Cryptology* (Lecture Notes in Computer Science), vol. 4833. Heidelberg, Germany: Springer-Verlag, 2007, pp. 29–50.

[11] E. Brier and M. Joye, "Weierstraß elliptic curves and side-channel attacks," in *Public Key Cryptography* (Lecture Notes in Computer Science), vol. 2274. Heidelberg, Germany: Springer-Verlag, 2002, pp. 335–345.

[12] H. Hisil, K. K. H. Wong, G. Carter, and E. Dawson, "Twisted edwards curves revisited," in *Advances in Cryptology* (Lecture Notes in Computer Science), vol. 4833. Heidelberg, Germany: Springer-Verlag, 2008, pp. 29–50.

[13] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B. Y. Yang, "High-speed high-security signatures," *J. Cryptogr. Eng.*, vol. 2, no. 2, pp. 77–89, 2012.

[14] D. J. Bernstein, P. Birkner, T. Lange, and C. Peters, "Twisted edwards curves," in *Progress in Cryptology* (Lecture Notes in Computer Science), vol. 5023. Heidelberg, Germany: Springer-Verlag, 2008, pp. 389–405.

[15] B. Baldwin, R. Moloney, A. Byrne, G. McGuire, and W. P. Marnane, "A hardware analysis of twisted Edwards curves for an elliptic curve cryptosystem," in *Lect. Notes Reconfigurable Computing: Architectures Tools Appl. - ARC'09, Heidelberg, Germany: Springer-Verlag*, vol. 2009, vol. 5453, pp. 355–361.

[16] D. J. Bernstein, "Curve25519: New Diffie-Hellman speed records," in *Public Key Cryptography* (Lecture Notes in Computer Science), vol. 3958. Heidelberg, Germany: Springer-Verlag, 2006, pp. 207–228.

[17] I. Liusvaara and S. Josefsson, *Edwards Curve Digital Signature Algorithm (EdDSA)*, document Internet-Draft: Draft-irtf-cfrg-eddsa-05, Internet Engineering Task Force, 2016. [Online]. Available: https://tools.ietf.org/html/draft-irtf-cfrg-eddsa-05

[18] M. Joye, "Elliptic curves and side-channel analysis," *ST J. Syst. Res.*, vol. 4, no. 1, pp. 17–21, 2003.

[19] M. Joye and S. M. Yen, "The Montgomery powering ladder," in *Cryptographic Hardware and Embedded Systems* (Lecture Notes in Computer Science), vol. 2523. Heidelberg, Germany: Springer-Verlag, 2003, pp. 291–302.

[20] S. B. Ors, L. Batina, B. Preneel, and J. Vandewalle, "Hardware implementation of an elliptic curve processor over GF(p)," in *Proc. IEEE Int. Conf. Appl.-Specific Syst. Arch. Process. (ASAP)*, Jun. 2003, pp. 433–443.

[21] K. Sakiyama, N. Mentas, L. Batina, B. Preneel, and I. Verbauwhede, "Reconfigurable modular arithmetic logic unit for high-performance public-key cryptosystems," in *Reconfigurable Computing: Architectures and Applications* (Lecture Notes in Computer Science), vol. 3985. Heidelberg, Germany: Springer-Verlag, Mar. 2006, pp. 347–357.

[22] C. J. Mcivor, M. Mcloone, and J. V. Mccanny, "Hardware elliptic curve cryptographic processor over GF(p)," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 53, no. 9, pp. 1946–1957, Sep. 2006.

[23] D. Schinianakis, A. P. Kakarountas, and T. Stouraitis, "A new approach to elliptic curve cryptography: An RNS architecture," in *Proc. IEEE Medit. Electrotech. Conf. (MELECON)*, May 2006, pp. 1241–1245.

[24] D. Schinianakis, A. Fournaris, H. Michail, A. Kakarountas, and T. Stouraitis, "An RNS implementation of an F_p elliptic curve point multiplier," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 56, no. 6, pp. 1202–1213, Jun. 2009.

[25] M. Esmaeildoust, D. Schinianakis, H. Javashi, T. Stouraitis, and K. Navi, "Efficient RNS implementation of elliptic curve point multiplication over GF(p)," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 8, pp. 1545–1549, Apr. 2013.

[26] S. Asif, M. S. Hossain, and Y. Kong, "High-throughput multi-key elliptic curve cryptosystem based on residue number system," *IET Comput. Digit. Techn.*, vol. 11, no. 5, pp. 165–172, 2017.

[27] J.-Y. Lai and C.-T. Huang, "Elixir: High-throughput cost-effective dual-field processors and the design framework for elliptic curve cryptography," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 16, no. 11, pp. 1567–1580, Nov. 2008.

[28] J. Fan, K. Sakiyama, and I. Verbauwhede, "Elliptic curve cryptography on embedded multicore systems," *Des. Autom. Embed. Syst.*, vol. 12, no. 3, pp. 231–242, 2008.

[29] K. Ananyi, H. Alrimeih, and D. Rakhmatov, "Flexible hardware processor for elliptic curve cryptography over NIST prime fields," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 17, no. 8, pp. 1099–1112, Aug. 2009.

[30] K. C. C. Loi and S. B. Ko, "Scalable elliptic curve cryptosystem FPGA processor for NIST prime curves," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 11, pp. 2753–2756, Jan. 2015.

[31] Z. Liu, D. Liu, and X. Zou, "An efficient and flexible hardware implementation of the dual-field elliptic curve cryptographic processor," *IEEE Trans. Ind. Electron.*, vol. 64, no. 3, pp. 2353–2362, Mar. 2017.

[32] S. Ghosh, M. Alam, D. R. Chowdhury, and I. S. Gupta, "Parallel crypto-devices for GF(p) elliptic curve multiplication resistant against side-channel attacks," *Comput. Electr. Eng.*, vol. 35, no. 2, pp. 329–338, 2009.

[33] J. Vliegen, N. Mentens, J. Genoe, A. Braeken, S. Kubera, and A. Touhafi, "A compact FPGA-based architecture for elliptic curve cryptography over prime fields," in *Proc. IEEE Int. Conf. Appl.-Specific Syst. Archit. Process.*, Jul. 2010, pp. 313–316.

[34] S. Ghosh, D. Mukhopadhyay, and D. Roychowdhury, "Petrel: Power and timing attack resistant elliptic curve scalar multiplier based on programmable GF(p) arithmetic unit," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 58, no. 8, pp. 1798–1812, Jan. 2011.

[35] H. Marzouqi, M. Al-Qutayri, and K. Salah, "An FPGA implementation of NIST 256 prime field ECC processor," in *Proc. IEEE Int. Conf. Electron. Circuits Syst. (ICECS)*, Dec. 2013, pp. 493–496.

[36] H. Marzouqi, M. Al-Qutayri, K. Salah, D. Schinianakis, and T. Stouraitis, "A high-speed FPGA implementation of an RSD-based ECC processor," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 1, pp. 151–164, Jan. 2016.

[37] Y. A. Shah, K. Javeed, S. Azmat, and X. Wang, "Redundant signed digit based high-speed elliptic curve cryptographic processor," *J. Circuits Syst. Comput.*, vol. 28, no. 5, 2018, Art. no. 1950081.

[38] J.-W. Lee, S.-C. Chung, H.-C. Chang, and C.-Y. Lee, "Efficient power-analysis-resistant dual-field elliptic curve cryptographic processor using heterogeneous dual-processing-element architecture," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 1, pp. 49–61, Feb. 2013.

[39] K. Javeed and X. Wang, "FPGA based high-speed SPA-resistant elliptic curve scalar multiplier architecture," *Int. J. Reconfigurable Comput.*, vol. 2016, no. 5, pp. 1–10, 2016.

[40] K. Javeed and X. Wang, "Low latency flexible FPGA implementation of point multiplication on elliptic curves over GF(p)," *Int. J. Circuit Theory Appl.*, vol. 45, no. 2, pp. 214–228, 2016.

[41] K. Javeed, X. Wang, and M. Scott, "High performance hardware support for elliptic curve cryptography over general prime field," *Microprocess. Microsyst.*, vol. 51, pp. 331–342, Jun. 2017.

[42] M. S. Hossain, Y. Kong, E. Saeedi, and N. C. Vayalil, "High-performance elliptic curve cryptography processor over NIST prime fields," *IET Comput., Digit. Techn.*, vol. 11, no. 1, pp. 33–42, 2017.

[43] X. Hu, X. Zheng, S. Zhang, S. Cai, and X. Xiong, "A low hardware consumption elliptic curve cryptographic architecture over GF(p) in embedded application," *Electronics*, vol. 7, no. 7, p. 104, 2018.

MD. MAINUL ISLAM received the B.Sc. degree in electrical and electronic engineering from the Khulna University of Engineering and Technology, Bangladesh, in 2018. He is currently pursuing the M.Sc. degree in electronics engineering with the Wireless Network and Communication Laboratory, Kookmin University, South Korea. His research interests include wireless communications, cryptography, elliptic curve cryptography (ECC), VLSI design, FPGA technology, the IoT security, blockchain, 5G, and 6G.

MD. SELIM HOSSAIN received the B.Sc. degree in electrical and electronic engineering (EEE) from the Khulna University of Engineering and Technology (KUET), Khulna, Bangladesh, in 2008, and the Ph.D. degree in engineering from Macquarie University, Sydney, Australia, in 2017. He was an Intern at Lund University, Sweden, from January 2016 to February 2016. In 2008, he joined the Department of Electrical and Electronic Engineering (EEE), KUET, as a Lecturer, where he became an Assistant Professor, in 2012. In 2013, he received the International Macquarie University Research Excellence Scholarship (iMQRES) to work toward the Ph.D. degree. In 2019, he joined the Department of EEE, KUET, as an Associate Professor. He has authored or coauthored over 35 refereed journal and conference papers. His research interests include cryptosystems (Elliptic Curve Cryptosystem), processor architectures, high-speed and energy-efficient arithmetic circuits, FPGA, ASIC, and antennas. He is also a member of Institution of Engineers, Bangladesh (IEB).

MOH. KHALID HASAN received the B.Sc. degree in electrical and electronic engineering from the Khulna University of Engineering and Technology, Bangladesh, in May 2017, and the M.Sc. degree in electronics engineering, Kookmin University, South Korea, in August 2019. Since 2019, he has been a full-time Researcher with the Wireless Network and Communication Laboratory, Department of Electronics Engineering, Kookmin University.

His current research interests include wireless communications, 6G, wireless security, and artificial intelligence.

MD. SHAHJALAL received the B.Sc. degree in electrical and electronic engineering from the Khulna University of Engineering and Technology, Bangladesh, in 2017, and the M.Sc. degree in electronics engineering from Kookmin University, South Korea, in 2019, where he is currently pursuing the Ph.D. degree in electronics engineering. His research interests include optical wireless communications (OWC), wireless security, optical camera communication (OCC), non-orthogonal multiple access (NOMA), software-defined networking (SDN), deep neural networks, 5G, and 6G.

YEONG MIN JANG received the B.E. and M.E. degrees in electronics engineering from Kyungpook National University, South Korea, in 1985 and 1987, respectively, and the Ph.D. degree in computer science from the University of Massachusetts, USA, in 1999. He was with the Electronics and Telecommunications Research Institute (ETRI), from 1987 to 2000. Since 2002, he has been with the School of Electrical Engineering, Kookmin University, Seoul, South Korea, where he has been the Director of the Ubiquitous IT Convergence Center, from 2005 to 2010. He has been the Director of the LED Convergence Research Center, since 2010, and the Director of the Internet of Energy Research Center, since 2018. He is currently a Life Member of the Korean Institute of Communications and Information Sciences (KICS). He received the Young Science Award from the Korean Government, from 2003 to 2006. He has organized several conferences and workshops, such as the International Conference on Ubiquitous and Future Networks, from 2009 to 2017, the International Conference on ICT Convergence, from 2010 to 2016, the International Conference on Information Networking 2015, and the International Workshop on Optical Wireless LED Communication Networks, from 2013 to 2016. His research interests include 5G/6G mobile communications, the Internet of Energy, eHealth, multiscreen convergence, public safety, optical wireless communications, optical camera communication, and the Internet of Things (IoT). He had served as the Founding Chair of the KICS Technical Committee on Communication Networks, from 2007 to 2008. He had served as the Executive Director of KICS, from 2006 to 2014, the Vice President of KICS, from 2014 to 2016, and the Executive Vice President of KICS, in 2018. He serves as the Co-Editor-in-Chief of ICT Express, which is published by Elsevier. He has been the Steering Chair of the Multi-Screen Service Forum, since 2011, and the Society Safety System Forum, since 2015. He has served as the Chairman of the IEEE 802.15 Optical Camera Communications Study Group, in 2014. He is also the Chairman of the IEEE 802.15.7m Optical Wireless Communications Task Group and Chairman of IEEE 802.15 Vehicular Assistive Technology (VAT) Interest Group.