Article

Design and Implementation of High-Performance ECC Processor with Unified Point Addition on Twisted Edwards Curve

Md. Mainul Islam ¹ , Md. Selim Hossain ² , Moh. Khalid Hasan ¹ , Md. Shahjalal ¹
and Yeong Min Jang ^{1,*}

¹ Department of Electronics Engineering, Kookmin University, Seoul 02707, Korea; mainul.islam@ieee.org (M.M.I.); khalidrahman45@ieee.org (M.K.H.); mdshahjalal26@ieee.org (M.S.)

² Department of Electrical and Electronic Engineering, Khulna University of Engineering & Technology (KUET), Khulna 9203, Bangladesh; selim@eee.kuet.ac.bd

* Correspondence: yjang@kookmin.ac.kr; Tel.: +82-02-910-5068

Received: 23 August 2020; Accepted: 4 September 2020; Published: 10 September 2020

Abstract: With the swift evolution of wireless technologies, the demand for the Internet of Things (IoT) security is rising immensely. Elliptic curve cryptography (ECC) provides an attractive solution to fulfill this demand. In recent years, Edwards curves have gained widespread acceptance in digital signatures and ECC due to their faster group operations and higher resistance against side-channel attacks (SCAs) than that of the Weierstrass form of elliptic curves. In this paper, we propose a high-speed, low-area, simple power analysis (SPA)-resistant field-programmable gate array (FPGA) implementation of ECC processor with unified point addition on a twisted Edwards curve, namely Edwards25519. Efficient hardware architectures for modular multiplication, modular inversion, unified point addition, and elliptic curve point multiplication (ECPM) are proposed. To reduce the computational complexity of ECPM, the ECPM scheme is designed in projective coordinates instead of affine coordinates. The proposed ECC processor performs 256-bit point multiplication over a prime field in 198,715 clock cycles and takes 1.9 ms with a throughput of 134.5 kbps, occupying only 6543 slices on Xilinx Virtex-7 FPGA platform. It supports high-speed public-key generation using fewer hardware resources without compromising the security level, which is a challenging requirement for IoT security.

Keywords: elliptic curve cryptography (ECC); elliptic curve point multiplication (ECPM); twisted Edwards curve; unified point addition; simple power analysis (SPA) attacks

1. Introduction

The Internet of Things (IoT) refers a global network, where billions of devices are connected through the Internet and share data with each other. Since most of these devices have constrained resources, data are usually stored in the cloud, where people can continuously upload and download data from anywhere via the Internet [1]. Security concerns arise as data owners have no control over the data management in the cloud-computing environment. The importance of data security and the limited resources of IoT devices motivate us to install lightweight cryptographic schemes that can satisfy the security, low-energy, and low-memory requirements of the existing IoT applications.

Elliptic curve cryptography (ECC), a public-key cryptography (PKC), has become a promising approach to the IoT security, smart card security, and digital signatures as it provides high levels of security with smaller key sizes. Compared with traditional Rivest–Shamir–Adleman (RSA) algorithm, ECC provides an equal level of security but with a shorter key length [2–4]. ECC can be implemented with low hardware resource usage and low energy consumption without degrading its security

level. Owing to low hardware use, it is well suited for the security of low-power, low-memory, and resource-constrained IoT devices. ECC implemented in a small chip can provide high-speed data encryption and decryption facilities. In addition, it prevents unauthorized devices from gaining access to wireless sensor networks (WSNs) by providing a key agreement protocol for the wireless sensor nodes connected to the IoT infrastructures in the networks [5–8]. An elliptic curve cryptosystem would be one of the best candidates to meet the privacy and security challenges emerged in radio-frequency identification (RFID) technologies [9–11]. Presently, ECC-based untraceable RFID authentication protocols are used in smart healthcare environments to enhance medical data security [12–14]. Elliptic curve-based digital signature schemes such as elliptic curve digital signature algorithm (ECDSA) [2] and Edwards curve digital signature algorithm (EdDSA) [15,16] are adopted in wireless body area networks (WBANs) to fulfill the security requirements for real-time health data (e.g., blood pressure, heart rate, and pulse) management [17–19]. Modern security protocols such as transport layer security (TLS) and datagram transport layer security (DTLS) deploy these signature schemes for the energy efficient mutual authentication of the servers and clients in IoT platforms [20–22].

An ECC hierarchy is equipped with four consecutive levels as shown in Figure 1. The first level contains finite field arithmetic, such as addition, subtraction, multiplication, squaring, and inversion, which can be performed in both the Galois binary field $GF(2^n)$ and Galois prime field $GF(p)$. The second level incorporates elliptic curve group operations, such as point addition (PA) and point doubling (PD). In the third level, elliptic curve point multiplication (ECPM) is accomplished by combining the elliptic curve group operations in a sequential manner. The top level includes ECC protocols such as ECDSA and EdDSA. The central and the most time-consuming operation in an elliptic curve-based cryptographic system is ECPM. The principle of ECPM can be specified as $Q = kP$, where P is a base point on an elliptic curve, k is a nonzero positive integer, and Q is another point on the curve [23]. Q and k are considered to be public key and private key, respectively, and P is regarded as the public-key generator. The retrieval of k knowing the points P and Q is known as elliptic curve discrete logarithm problem (ECDLP) [2] that measures the security strength of the ECPM operation and finds out the weaknesses of the system. The easiest technique to accomplish ECPM is the binary/double-and-add (DAA) algorithm [2] that requires fewer hardware resources compared with other available methods. Therefore, ECC schemes adopting the DAA-based ECPM are suited for IoT applications because of their lower hardware resource requirements and lower power consumption. The major disadvantage of the DAA method is that the DAA-based ECPM is vulnerable to simple power analysis (SPA) attacks [24,25] unless it uses unified point operations.

Figure 1. Hierarchy of elliptic curve cryptography.

Edwards curves, a family of elliptic curves, are gaining enormous attention among security researchers because of their simplicity and high resistance against SCAs [26]. ECPM on Edwards curves is faster and more secure than that on the Weierstrass form of elliptic curves [27,28]. Edwards curves

have the advantage of providing strongly unified addition formulas [28], which cover both PA and PD. Separate hardware architectures for PA and PD are not required to perform ECPM. Moreover, unified PA prevents probable SPA attacks by making the secret key indistinguishable from power tracing. When ECPM adopts the same module for PA and PD, the binary bit pattern of the secret key cannot be retrieved by SPA. The twisted Edwards curves are a generalization of Edwards curves [29], which are mainly used in the digital signature scheme EdDSA. One of the most compatible twisted Edwards curves in digital signature systems is Edwards25519, which is the Edwards form of the elliptic curve Curve25519 [23,30]. In modern times, Edwards25519 curve is used for a high-speed, high-security digital signature scheme called Ed25519 [15,16]. ECPM using unified twisted Edwards curve not only provides high resistance against SPA but also it reduces the area of ECC processors.

ECC can be accomplished with both hardware and software approaches. Although the software implementation is simple and cost-effective, it cannot provide high-speed computation as the hardware implementation can. Indeed, the hardware implementation of ECC with limited resources is a highly challenging task because low hardware use leads to a lower computational speed. In this point of view, Edwards curves are more effective than classical elliptic curves as they can be implemented on a smaller area with higher processing speed. Most of the hardware implementations of ECC reported in the literature are based on the Weierstrass form of elliptic curves. Few hardware implementations based on twisted Edwards curves over $GF(p)$ have been reported. Baldwin et al. [31] first documented hardware implementation of a reconfigurable 192-bit ECC processor adopting twisted Edwards curve over $GF(p)$. They provide a comparison between the FPGA implementation of an elliptic curve-based point multiplication and that of a twisted Edwards curve for different number of arithmetic logic units (ALUs) operated in parallel, which shows the Edwards curve as more efficient. Additionally, the twisted Edwards curve point operations are compared with the unified version of these operations. Although the unified version shows little bit worse performance, it provides a higher resistance against SPA. Liu et al. [21] present a computable endomorphism on twisted Edwards curves to boost the speed of ECDSA verification process. They provide area-efficient hardware architecture for signature verification with its FPGA implementation. Application specific integrated circuit (ASIC) implementation of the architecture is also provided for low-cost applications. The implementation results show that the design reduces approximately 50% of the number of PD operations required. Parallel architectures for ECPM on extended twisted Edwards are proposed by Abdulrahman et al. [32]. The authors present a new radix-8 ECPM algorithm to cope with SCAs and speed up computations. However, no hardware implementation of these architectures is reported.

In this paper, a lightweight FPGA-based hardware implementation of ECC over $GF(p)$ is proposed for IoT appliances. The major contributions of this paper are summarized as follows:

- An efficient radix-4 interleaved modular multiplier is proposed to perform 256-bit modular multiplication over a prime field.
- A novel hardware architecture for strongly unified PA on the Edwards25519 curve is proposed.
- An efficient ECPM scheme is proposed to perform high-speed point multiplication on the Edwards25519 curve. The same module is used for PA and PD to prevent probable SPA attacks. The area required by the scheme is significantly lower than other available designs for ECPM.
- ECPM is performed in projective coordinates to avoid the most expensive (in terms of computational complexity) modular division operation. In addition, a projective-to-affine (P2A) converter is proposed to transform the projective output into its affine form. This type of transformation reduces the computation time additionally required for the modular division operation performed in affine coordinate-based PA.
- An ECC processor is designed by combining the ECPM scheme and the P2A converter in such a manner as to reduce the number of modular inversion operations required. The area-delay product of the proposed ECC processor is considerably small that ensures a better performance of our processor.

The rest of this paper is organized as follows:

Section 2 presents the mathematical background of the twisted Edwards curve and unified PA formula. Section 3 presents the proposed hardware architectures for field operations (modular multiplication and modular inversion), unified PA, ECPM, and ECC processor. Section 4 presents the implementation results of the proposed designs. Section 5 shows a performance comparison of our proposed ECC processor with other related processors. Finally, Section 6 concludes this research study.

2. Mathematical Background

This section presents the twisted Edwards curve with its affine and projective representations as well as the unified PA formula for the curve.

2.1. Twisted Edwards Curve

The affine representation of a twisted Edwards curve over a prime field $\mathbb{F}_p$ with not characteristic 2 is given by the equation [23,29]:

$$t_{a,d} : ax^2 + y^2 = 1 + dx^2y^2, \quad (1)$$

where $a, b \in \mathbb{F}_p \setminus \{0, 1\}$ with $a \neq d$. When $a = 1$, the curve is called untwisted Edwards curve or, formally, Edwards curve. In the case of $a = -1$, the curve will be

$$t_d : -x^2 + y^2 = 1 + dx^2y^2. \quad (2)$$

when $a = -1$, $d = -121665/121666$, and $p = 2^{255} - 19$, the curve is called Edwards25519 that is the Edwards form of the elliptic curve Curve25519 [23].

In a projective or Jacobian coordinate system, each point (x, y) on $t_{a,d}$ is represented by a triplet form (X, Y, Z) . The affine point $P(x, y)$ corresponds to the projective point $P(X = x, Y = y, Z = 1)$. The projective point $P(X, Y, Z)$ corresponds to the affine point $P(x = X/Z, y = Y/Z)$ with $Z \neq 0$.

The projective representation of the curve $t_{a,d}$ is given by the equation [23,29]:

$$T_{a,d} : (aX^2 + Y^2)Z^2 = Z^4 + dX^2Y^2. \quad (3)$$

The projective form of the curve t_d is given by the equation:

$$T_d : (-X^2 + Y^2)Z^2 = Z^4 + dX^2Y^2. \quad (4)$$

2.2. Unified Point-Addition Formula

PA on the curve T_d in projective coordinates is given by the equation:

$$P_1(X_1, Y_1, Z_1) + P_2(X_2, Y_2, Z_2) = P_3(X_3, Y_3, Z_3). \quad (5)$$

where P_1 and P_2 are two points on the curve and P_3 is the resultant point.

The unified PA formula [29] for T_d can be given as follows:

$$\begin{aligned} A &= X_1X_2, B = Y_1Y_2, C = Z_1Z_2, D = X_1Y_2, \\ E &= X_2Y_1, F = AB, G = A + B, \\ H &= C^2, I = D + E, J = dF, K = CG, \\ L &= IC, M = H + J, N = H - J, \\ X_3 &= LN, Y_3 = MK, Z_3 = MN. \end{aligned} \quad (6)$$

The above formula is applicable for both PA and PD. PD can be performed considering that the points P_1 and P_2 are identical.

3. Proposed Hardware Architectures

This Section presents the proposed hardware architectures for ECC operations and the final ECC processor.

3.1. Modular Multiplication

Modular multiplication is the most important arithmetic operation of an ECC processor. The speed and occupied area of the processor entirely depend on it. Although a radix-2 multiplier consumes less hardware resources compared to higher radix (e.g., radix-4 and radix-8) multipliers [33], it is not compatible for high-speed multiplication because of its high latency. To reduce the latency, an efficient radix-4 interleaved modular multiplication algorithm is proposed as demonstrated in Algorithm 1. It requires $n/2 + 1$ clock cycles (CCs) to multiply two n -bit integers A and B over the prime field $GF(p)$, where p is an n -bit prime number. Figure 2 illustrates the proposed modular multiplier based on this algorithm.

Algorithm 1 Proposed Radix-4 Interleaved Modular Multiplication

Input : $A = \sum_{i=0}^{n-1} a_i 2^i, B = \sum_{i=0}^{n-1} b_i 2^i; a_i, b_i \in \{0, 1\}$

Output : $C = (A \cdot B) \bmod p$

```

1:  $C \leftarrow 0$ ;
2:  $T \leftarrow B \parallel "01"$ ;
3: while  $T(n-1 \text{ downto } 0) \neq 0$  do
4:    $D \leftarrow 4C$ ;
5:   if  $T(n+1 \text{ downto } n) = "01"$  then
6:      $E \leftarrow D + A$ ;
7:   else if  $T(n+1 \text{ downto } n) = "10"$  then
8:      $E \leftarrow D + 2A$ ;
9:   else if  $T(n+1 \text{ downto } n) = "11"$  then
10:     $E \leftarrow D + 3A$ ;
11:   else
12:     $E \leftarrow D$ ;
13:   end if;
14:    $C \leftarrow E \bmod p$ ;
15:    $T \leftarrow T(n-1 \text{ downto } 0) \parallel "00"; \backslash \backslash \text{left shift operation}$ 
16: end while;
17: return  $C$ ;

```

Modular multiplication is obtained by performing iterative addition of its interim partial products reducing to modulo p . A shift-left register “Reg T” is used to perform left to right bitwise multiplication and for a synthesizable loop operation. $T[(n+1) : 2]$ is precomputed as the multiplier B and $T[1 : 0]$ is precomputed as “01”. These two extra bits are added at the rightmost position of the register T to determine the appropriate end of the loop in the case of $b_0 = 0$. At the beginning of each iteration, accumulator C is quadrupled and computed as D . For the bitwise multiplication, A , $2A$, and $3A$ are separately added to D . MUX1 is used to select one of the four outputs $D, D + A, D + 2A$, and $D + 3A$ as E based on the three bits $T[(n+1) : n]$. If T_{n+1} and T_n both are zero, D remains unchanged and E becomes D . At the end of each iteration, E is reduced to modulo p and T is shifted to the left by 2 bits. The modulo operation $(E \bmod p)$ is performed by subtracting the prime numbers p to $(j-1)p$ from E , where E is always less than jp ; ($j = 3, 4, 5, \dots$). In this module, $(E \bmod p)$ is obtained by subtracting the prime numbers p to $6p$ from E as E is always less than $7p$. These subtractions are executed using the 2’s complement method. MUX2 selects one of the seven outputs $E, E - p, E - 2p, E - 3p, E - 4p, E - 5p$, and $E - 6p$ as C for the next iteration based on the comparisons $E \geq p, E \geq 2p, E \geq 3p, E \geq 4p, E \geq 5p$, and $E \geq 6p$. These comparisons are obtained by checking the three bits $E[(n+1) : (n-1)]$. After $n/2$

number of iterations, B , as well as $T[(n-1):0]$, is shifted to zero value and the execution is stopped. The final content of the register “Reg C” is the modular multiplication of A and B .

Figure 2. Proposed modular multiplier.

A total of $n/2 + 1$ CCs are required to perform the modular multiplication operation, where $n/2$ CCs correspond to $n/2$ number of iterations and one extra CC is required for the initialization. To perform modular squaring, the inputs A and B are taken as identical.

3.2. Modular Inversion

Modular inversion is the costliest (in terms of the hardware resource requirements) arithmetic operation in finite fields. In affine representations, PA and PD require modular inversion operation to perform modular division. In this study, although our ECC processor is designed in projective coordinates, modular inversion is required for P2A conversion. Algorithm 2 [2] demonstrates the binary modular inversion for the P2A conversion module proposed in this paper. The hardware architecture of this module is depicted in Figure 3.

Figure 3. Proposed hardware architecture for modular inversion.

Algorithm 2 Binary Modular Inversion [2]**Input :** $B = \sum_{i=0}^{n-1} b_i 2^i; b_i \in \{0, 1\}$ **Output :** $C = B^{-1} \bmod p$

```

1:  $C \leftarrow 0, q \leftarrow B, r \leftarrow p, s \leftarrow 1, t \leftarrow 0;$ 
2: while  $q \neq 1$  do
3:   while  $q(0) = 0$  do
4:      $q \leftarrow q/2;$ 
5:     if  $s(0) = 0$  then
6:        $s \leftarrow s/2;$ 
7:     else
8:        $s \leftarrow (s + p)/2;$ 
9:     end if;
10:  end while;
11:  while  $r(0) = 0$  do
12:     $r \leftarrow r/2;$ 
13:    if  $t(0) = 0$  then
14:       $t \leftarrow t/2;$ 
15:    else
16:       $t \leftarrow (t + p)/2;$ 
17:    end if;
18:  end while;
19:  if  $q > r$  then
20:     $q \leftarrow q - r;$ 
21:    if  $s > t$  then
22:       $s \leftarrow s - t;$ 
23:    else
24:       $s \leftarrow s + p - t;$ 
25:    end if;
26:  else
27:     $r \leftarrow r - q;$ 
28:    if  $t > s$  then
29:       $t \leftarrow t - s;$ 
30:    else
31:       $t \leftarrow t + p - s;$ 
32:    end if;
33:  end if;
34: end while;
35: return  $s \bmod p;$ 

```

The contents of the registers “Reg Q”, “Reg R”, “Reg S”, and “Reg T” are updated in every iteration. Five multiplexers such as MUX1, MUX2, MUX3, MUX4, and MUX5 are used to select corresponding outputs, satisfying different conditions by their select lines. In the case of q being even, MUX1 selects $q/2$ and MUX3 selects $s/2$ if s is even or $(s + p)/2$ if s is odd. In the case of q being odd and greater than r , MUX1 selects $q - r$ and MUX3 selects $s - t$ if $s > t$ or $s + p - t$ if $s < t$. The comparisons $q > r$ and $s > t$ are obtained by checking the sign bits of the subtractions $q - r$ and $s - t$, respectively. If q is odd and less than r , q and r both remain unchanged. Similarly, MUX2 selects one of the three outputs $r, r/2$, and $r - q$ based on the conditions $r(0) = 0$ and $r > q$. MUX4 selects one of the five outputs $t, t/2, (t + p)/2, t - s$, and $t + p - s$ based on the conditions $r(0) = 0, t(0) = 0, r > q$, and $t > s$. MUX5 is used to select the final result as $(s \bmod p)$ if $q = 1$. In this regard, q is subtracted by 2 to check whether $q < 2$ at the end of each iteration. When the sign bit of the subtraction $q - 2$ is 1, $(s \bmod p)$ is stored in the register “Reg C”, which is the modular inversion of B .

In this architecture, on average $n + n/4$ CCs are required to perform the modular inversion operation, where n number of iterations are to reduce the n -bit variable q to 1 in a regular manner and additional $n/4$ number of iterations are for such uncertain case as q being odd. The clock cycles required for the modular inversion operation may vary from our estimation depending on the binary bit pattern of B .

3.3. Unified Point Addition

Unified PA is required to perform both PA and PD by the same module so as to prevent possible SPA attacks in ECPM. The proposed hardware architecture for the unified PA formula described in (6) is depicted in Figure 4. The architecture includes 12 multiplications, 1 squaring, 3 additions, and 1 subtraction, which can be denoted as (12M+1S+4A). The proposed design consists of four consecutive levels, in which the arithmetic modules are connected in a sequential manner. The modules are arranged in horizontally parallel among the levels to achieve the shortest data path. The whole architecture is efficiently balanced to reduce the area required. Start signals are used to start the arithmetic operations and Done signals are used to confirm the end of the operations. The Done signals of the modules at each level are considered to be the Start signals of the modules at its subsequent level. AND blocks are used to synchronize the horizontal modules in time (e.g., if the Done signals d_1, d_2, d_3, d_4 , and d_5 all be 1, the Start signal s_1 will be 1; otherwise, it will be 0). The modular multiplier and the squarer require $n/2 + 1$ CCs to perform modular multiplication and squaring. Modular addition and subtraction are completed in only one CC. The level that contains any multiplication or squaring operation requires $n/2 + 1$ CCs and the level that contains no multiplication or squaring requires one CC to jump to the next level. In this design, a total of $2n + 5$ CCs are required to complete the unified PA operation.

Figure 4. Proposed hardware architecture for unified PA.

3.4. Elliptic Curve Point Multiplication

ECPM is the ultimate operation of an ECC processor. It multiplies a point on an elliptic curve with a scalar. The execution time of ECC schemes is dominated by ECPM. Let $P(X, Y, Z)$ be a point on the curve T_d , k be a scalar that is considered to be secret key. A public key $Q(X, Y, Z)$ is generated from the known base point P and the secret key k by performing ECPM as follows:

$$Q = kP, \quad (7)$$

where Q is also a point on the curve. It can be obtained by adding P to itself $k - 1$ times such as

$$Q = P + \underbrace{P + \dots + P}_{k-1 \text{ times}}. \quad (8)$$

If k is expressible as a power of 2, Q can be obtained by doubling P on itself $\log_2 k$ times such as

$$Q = \underbrace{\dots 2(2(P))}_{\log_2 k \text{ times}}. \quad (9)$$

In the binary / DAA method, ECPM is performed by a combination of PD and PA following the binary bit pattern of the secret key as shown in Algorithm 3. In this algorithm, separate modules are required to perform PA and PD. The power consumption of the two separate modules are different. Monitoring these two power levels by SPA, the bit pattern of k can be retrieved as shown in Figure 5. Moreover, k can be assumed by timing analysis; hence, ECPM based on this algorithm is vulnerable to SPA attacks. To cope with SPA, Algorithm 3 is modified to Algorithm 4, where PD is replaced by unified PA. According to this algorithm, power is only consumed for PA with a fixed power consumption, which is independent of the bit pattern of k as shown in Figure 6. Since the power consumption is the same across all the iterations, this algorithm is free from SPA. Figure 7 illustrates the proposed hardware architecture for ECPM based on Algorithm 4. Two point-addition blocks PA1, PA2 and three multiplexers MUX1, MUX2, MUX3 are used in this processor. Initially, Q_1 is precomputed as P . PA1 adds the point Q_1 to itself and the output Q_2 goes to the input of PA2. Identical inputs are inserted in PA1 to perform PD by means of PA. One of the two inputs of PA2 is the output of PA1 and the other one is P or 0. If $k_i = 1$, PA2 adds the point P to the point Q_2 and the output Q_3 goes to the input of the PA1 via the register Rg. On the contrary, if $k_i = 0$, PA2 remains idle and the output of PA1 directly goes to its input via Rg. MUX1 is used to select the i th bit of k by $\log_2 l$ number of select lines, where l is the bit length of k . Based on k_i , MUX2 selects P or 0 as one of the two inputs of PA2; MUX3 selects Q_2 or Q_3 as the input Q_1 for the subsequent iteration.

Algorithm 3 DAA ECPM without Unified PA [2]

Input : $P(X, Y, Z), k = \sum_{i=0}^{l-1} k_i 2^i; k_i \in \{0, 1\}, k_{l-1} = 1$

Output : $Q(X, Y, Z)$

```

1:  $Q \leftarrow P;$ 
2: for  $i$  from  $l - 2$  to 0 do
3:    $Q \leftarrow 2Q;$     \ \ PD
4:   if  $k_i = 1$  then
5:      $Q \leftarrow Q + P;$   \ \ PA
6:   end if;
7: end for;
8: return  $Q;$ 
```

Algorithm 4 Proposed Unified PA-based ECPM

Input : $P(X, Y, Z), k = \sum_{i=0}^{l-1} k_i 2^i; k_i \in \{0, 1\}, k_{l-1} = 1$

Output : $Q(X, Y, Z)$

```

1:  $Q_1 \leftarrow P$ ;
2: for  $i$  from  $l-2$  to  $0$  do
3:    $Q_2 \leftarrow Q_1 + Q_1$ ;   $\backslash \backslash$  PA
4:   if  $k_i = 1$  then
5:      $Q_3 \leftarrow Q_2 + P$ ;   $\backslash \backslash$  PA
6:      $Q_1 \leftarrow Q_3$ ;
7:   else
8:      $Q_1 \leftarrow Q_2$ ;
9:   end if
10: end for
11: return  $Q_1$ ;

```

Figure 5. Simplistic representation of SPA in conventional DAA ECPM.

Figure 6. Simplistic representation of SPA in proposed unified PA-based ECPM.

Figure 7. Proposed hardware architecture for ECPM.

For the l -bit k , the register stores kP as the final result after $l - 1$ number of iterations. The average CCs required to perform the ECPM can be calculated as

$$\begin{aligned} \text{ECPM}_{\text{CC}} &= (l - 1) \times (\text{PA1}_{\text{CC}} + \text{Rg}_{\text{CC}}) + l/2 \times \text{PA2}_{\text{CC}} \\ &= (l - 1) \times (2n + 5 + 1) + (l/2) \times (2n + 5) \\ &= 3nl - 2n + 8.5l - 6. \end{aligned} \quad (10)$$

For $l = n$,

$$\text{ECPM}_{\text{CC}} = 3n^2 + 6.5n - 6. \quad (11)$$

PA1 and Rg remain active in every iteration, whereas PA2 goes idle in the case of $k_i = 0$. In every iteration, a total $2n + 6$ CCs are spent by PA1 and Rg. Additional $2n + 5$ CCs are spent by PA2 if $k_i = 1$. On average, $l(n + 2.5)$ CCs are spent by PA2 across the ECPM. For the n -bit k , the latency of the ECPM is approximately $3n^2 + 6.5n - 6$ CCs. This latency may vary depending on the bit pattern of the key; it increases with the number of 1 and decreases with the number 0 present in the bit pattern. In this study, an average case is considered. This means that the key has equal number of 1 and 0 in its bit pattern, although this is not always the case.

3.5. Proposed ECC Processor

A time-area-efficient ECC processor is designed for public-key generation using the proposed projective coordinate-based ECPM along with a P2A converter as shown in Figure 8. This processor will generate a public key from a private key and a base point on T_d . Initially, the affine base point $P(x, y)$ is transformed into its projective form such as $P(X, Y, Z)$ by an affine-to-projective (A2P) converter. The public key $Q(X, Y, Z)$ is obtained by performing ECPM of the projective point $P(X, Y, Z)$ with the secret key k . Finally, $Q(X, Y, Z)$ is transformed into its affine form such as $Q(x, y)$ by the P2A converter. For the P2A conversion, Z is inverted by the proposed modular inversion module and separately multiplied by X and Y . The latency required by the processor to process the ECPM operation along with the coordinate conversions is $3n^2 + 8.25n - 5$ CCs, which is the total sum of the latency of ECPM, modular inversion, and modular multiplication.

Figure 8. Proposed ECC processor for public-key generation.

4. Implementation Results

The proposed ECC processor was programmed in VHDL and implemented using the Xilinx ISE 14.7 Design Suite software. Xilinx ISim simulator was used to simulate the ECC operations. The simulation results were verified by the Maple 18 software. Synthesizing, mapping, placing, and routing of the proposed ECC modules were performed on Xilinx Virtex-7 and Virtex-6 FPGA platforms, separately. The details of these FPGA platforms and settings are as follows:

- Platform 1: Virtex-7 (XC7VX690T)
- Platform 2: Virtex-6 (XC6VHX380T)
- Design Goal: Balanced
- Design Strategy: Xilinx Default
- Optimization Goal: Speed
- Optimization Effort: Normal

The implementation results of the proposed ECC modules are summarized in Table 1. On Platform 1, all the modules run at a maximum frequency of 104.39 MHz. The proposed ECC processor occupies 6543 slices (25,898 LUTs) and generates a public key from a given 256-bit private key in 1.9 ms with a throughput of 134.5 kbps. On Platform 2, the modules operate at a maximum frequency of 93.23 MHz. The numbers of slices and LUTs used by the processor are 6579 and 25,968, respectively, the delay of the public-key generation is 2.13 ms, and the throughput is 120.1 kbps.

Table 1. Implementation results of the proposed ECC modules on different FPGA platforms over $\mathbb{F}_p$ -256.

Operation	Platform	CCs	Area (Slices)	Area (LUTs)	Maximum Frequency (MHz)	Time (μ s)	Throughput (Mbps)
Modular multiplication	Virtex-7	129	416	1451	104.39	1.24	207.2
	Virtex-6	129	420	1460	93.23	1.38	185
Modular inversion	Virtex-7	320	1197	4155	110.65	2.89	83.5
	Virtex-6	320	1209	4156	97.94	3.27	74.6
Unified PA	Virtex-7	517	4159	15,594	104.39	4.95	51.7
	Virtex-6	517	4292	15,593	93.23	5.55	46.2
ECPM (projective)	Virtex-7	198,266	5457	21,194	104.39	1899	134.8×10^{-3}
	Virtex-6	198,266	5541	21,224	93.23	2126	120.4×10^{-3}
Public-key generation	Virtex-7	198,715	6543	25,898	104.39	1903	134.5×10^{-3}
	Virtex-6	198,715	6579	25,968	93.23	2131	120.1×10^{-3}

The performance of the ECC modules on the Virtex-6 FPGA platform is a little bit worse compared to the Virtex-7 FPGA platform in terms of speed. However, the area use of the different modules on these platforms are almost the same. It must be noted that no digital signal processing (DSP) slice is used to implement our processor. Although DSP slices increase processing speed, they increase processor's cost as well.

5. Performance Comparison

Several hardware implementations of ECC have been reported in [34–53], where some authors aimed to minimize the area use while others tried to reduce the computation time. Achieving a higher processing speed with low-area use is technically challenging. We tried to maintain a balance

between area and time as they are two important performance criteria of a cryptographic processor. A performance comparison of our proposed ECC processor with other related designs is presented in Table 2.

Table 2. Performance comparison of the proposed ECC processor with other related designs over $\mathbb{F}_p$ -256.

Design	Platform	Area (Slices)	CCs	Frequency (MHz)	Time (ms)	Throughput (kbps)	Area $\times$ Time
Ours (a)	Virtex-7	6.5k	198.7	104.39	1.9	134.49 ^a	12.35
Ours (b)	Virtex-6	6.6k	198.7	93.23	2.13	120.12 ^a	14.05
[34]	Virtex-7	24.2k, 2.8k DSPs	215.9	72.9	2.96	1816.2	71.63
[35]	Kintex-7	11.3k	397.3	121.5	3.27	78.28	36.95
[36]	Virtex-6	65.6k	153.2	327	0.47	546.42 ^a	30.83
[37]	Virtex-5	8.7k	361.6	160	2.26	113.27 ^a	19.66
[38]	Virtex-5	10.2k	442.2	66.7	6.63	38.61 ^a	67.63
[39]	Virtex-4	12k	459.9	36.5	12.6	20.32 ^a	151.20
[40]	Virtex-4	9.4k, 14 DSPs	610	20.44	29.84	8.58 ^a	280.50
[41]	Virtex-4	35.7k	207.1	70	2.96	86.53 ^a	105.67
[42]	Virtex-4	13.2k	200	40	5	51	66.00
[43]	Virtex-4	20.6k	191.6	49	3.91	65.47	80.55
[44]	Virtex-4	20.1k	331.1	43	7.7	33.25 ^a	154.77
[45]	Virtex-4	20.8k, 32 DSPs	414	60	6.9	37.10 ^a	143.52
[46]	Virtex-4	7k, 8 DSPs	993.7	182	5.46	46.88 ^a	38.22
[47]	Spartan-3	27.6k	708	40	17.7	14.46 ^a	488.52
[48]	Virtex-II Pro	12k	337.7	36	9.38	27.29 ^a	112.56
[49]	Virtex-II Pro	8.3k	163.2	37	4.41	58.04 ^a	36.60
[50]	Virtex-II Pro	15.8k, 256 DSPs	151.4	39.5	3.86	66.74	60.98
[51]	Virtex-II Pro	41.6k	252.1	94.7	2.66	96.17 ^a	110.66
[52]	Virtex-E	16.4k	156.8	39.7	3.95	64.82 ^a	64.78
[53]	Virtex-E	14.2k	118.3	34.7	3.41	75.09 ^a	48.42

^a Estimated by the authors of this paper as Throughput = (Maximum frequency $\div$ CCs) $\times$ 256.

The residue number system (RNS)-based ECC design reported in [34] provides a higher throughput (1816.2 kbps) by performing ECPM on 21 keys in parallel. Conventional DAA method is adopted for ECPM, where PA and PD are executed by separate modules carrying high risk of SPA attacks. On Virtex-7 FPGA, the design consumes 96,867 LUTs (approx. 24,216 slices) with 2799 additional DSP slices. Although the throughput of this design is higher than that of our design, it costs 3.7 times more hardware resources. The novelty of this design is that it processes 21 keys simultaneously, which prevents template-based attacks by increasing the computation complexity. In [35], the authors propose a high-performance ECC processor with its ASIC and FPGA implementations. A novel hardware architecture for combined PA-PD operation in Jacobian

coordinates is proposed to achieve high-speed ECPM with low hardware use. On Kintex-7 FPGA, the processor separately designed in affine and Jacobian coordinates performs ECPM in 4.7 ms and 3.27 ms, occupying 9.3k and 11.3k slices, respectively. Our processor implemented on 7-series FPGA is 1.72 times faster and costs 1.73 times less slices as compared with this processor designed in Jacobian coordinates. The throughput of our design is 1.76 times higher. A high-speed ECC processor is proposed in [36] providing redundant signed digit (RSD)-based carry free modular arithmetic. The processor performs high-speed ECPM with a higher throughput. However, it occupies 10 times more slices on Virtex-6 FPGA than our processor. Although RSD representation offers fast computation, it consumes a vast amount of hardware resources, which makes processor bulky and hence not suited for low-power IoT devices. The high-speed RSD-based modular multiplier proposed in this paper performs single multiplication in only 0.34 μ s, consuming 22k LUTs. In comparison with this multiplier, our proposed modular multiplier performs single multiplication in 1.45 μ s and consumes only 1.3k LUTs with almost 4 times better efficiency in terms of area-time (AT) product. The RSD-based ECC processors reported in [37,38] present comprehensive pipelining technique for Karatsuba–Ofman multiplication to achieve high throughput. Our processor has smaller AT product compared with these processors.

Liu et al. [39] propose a hardware-software approach for flexible dual-field ECC processor with its ASIC and FPGA implementations. The traditional DAA method for ECPM is replaced by the double-and-add-always (DAAA) method to protect the processor from SPA attacks. Although the DAAA method for ECPM provides high resistance against SPA, it increases the computational complexity and hence reduces the frequency and throughput. In addition, it consumes more power than the conventional DAA method as PA and PD are performed in every iteration. Our processor is protected against SPA attacks by implementing the cost-effective DAA algorithm with unified PA. When compared to our processor, the main advantage of this processor is that it is flexible and reconfigurable over different field orders. In addition, it can perform ECPM over both $GF(2^n)$ and $GF(p)$, whereas our processor performs ECPM over $GF(p)$ only.

Hu et al. [40] propose an SPA-resistant ECC design over $GF(p)$, providing its ASIC and FPGA implementations. The design uses 9370 slices with 14 additional DSP slices on Virtex-4 FPGA. Despite employing additional DSP slices, the speed of this design is considerably low. It takes 29.84 ms with a frequency of 20.44 MHz to perform single ECPM over a 256-bit prime field. The advantage of this design that makes it well suited for embedded applications is its reconfigurable computing capability. A low latency ECPM design is proposed in [41] exploiting parallel multiplication over $GF(p)$. Protection against timing and SPA attacks is provided by using the DAAA method for ECPM. The latency of this design is $3n^2 + 37n + 4n$ CCs, whereas the latency of our design is $3n^2 + 8.25n - 5$ CCs. Therefore, the computational complexity of ECPM in this design is higher than that in our design. The radix-4 parallel interleaved modular multiplier proposed in this paper performs multiplication in 0.79 μ s, consuming 6.3k LUTs. Four multiplier units are operated in parallel to speed up the multiplication process. The main feature of this design is its capability to perform ECPM over $GF(p)$ with any arbitrary value of p less than or equal to 256 bits in size.

The design reported in [42] exploits the Montgomery ladder algorithm for SPA-resistant ECPM. Although the Montgomery ladder algorithm offers lower latency ECPM and higher resistance against SPA than the general DAA method [23], it deals with around 50% additional PA operations that results in a higher power consumption. Hence, the DAA method is more efficient than the Montgomery ladder technique in terms of energy consumption. The advantage of this design is that it supports any prime number $p \leq 256$ -bit. In [43], the authors present a high-performance hardware design for ECPM adopting non-adjacent form (NAF) method. Although NAF method has the advantage of reducing the latency of ECPM, the computational complexity and its vulnerability to SCAs are high in this method. Moreover, additional point subtraction operation is required for NAF scalar multiplication. Like the designs reported in [40,41], this design is programmable for any prime $p \leq 256$ -bit. Parallel crypto design is proposed in [44] using the DAAA method to perform SCA-resistant ECPM over different

field orders. The design is represented in affine coordinates, where PA and PD require modular division operations. Modular division is the most time-consuming arithmetic operation in finite fields. Therefore, this design is not convenient for high-speed computation. However, it provides high resistance against timing and SPA attacks by parallel computation of PA and PD.

Ananyi et al. [45] propose a flexible hardware ECC processor that supports five National Institute of Standard and Technology (NIST) recommended prime curves. They provide a comparison between the binary and NAF ECPM over all five NIST prime fields such as p_{192} , p_{224} , p_{256} , p_{384} , and p_{521} , where the NAF ECPM is found to be more time-efficient. Their processor consumes 20,793 slices (31,946 LUTs) with 32 additional DSP blocks on Virtex-4 FPGA and performs the binary ECPM in 6.9 ms and the NAF ECPM in 6.1 ms over p_{256} . The modular inverter designed in this paper operates at a frequency of 58.6 MHz costing 10,921 slices with 32 DSP blocks, whereas our modular inverter implemented on Virtex-7 FPGA runs at 110.65 MHz consuming 1197 slices without any DSP block.

A scalable ECC processor developed by Loi et al. [46] can perform ECPM on five NIST suggested prime curves such as P-192, P-224, P-256, P-384, and P-521 without hardware reconfiguring. On Virtex-4 FPGA, this processor performs ECPM in 5.46 ms, occupying 7020 slices along with 8 additional DSP slices. Despite using DSP slices, the computational speeds of the processors reported in [45,46] are low. The main significance of these processors is that they are flexible over the five NIST prime fields and hence they can be programmed to perform ECPM for variable prime numbers ranging from 192 to 521 bits in size without being architecturally reconfigured. The processors reported in [47–53], are implemented on some backdated FPGA platforms, which are now obsolete.

Performance comparison in terms of AT product is shown in Figure 9. The AT product of our design is lower than that of the other designs tabulated in Table 2. Figure 10 shows performance comparison in terms of throughput per slice. The per slice throughput of our design is higher than that of the other designs except [34]. The RNS-based design reported in [34] provides a higher throughput by performing ECPM on 21 keys concurrently. Our processor's low value of AT product and high value of throughput ensure a better performance in IoT platforms. However, a fair comparison is not possible because the compared processors are implemented on different FPGA platforms. Our proposed ECC processor is implemented only on the Virtex-7 and Virtex-6 FPGAs because the number of input/output blocks (IOBs) is limited in earlier FPGAs. Furthermore, the earlier FPGAs such as Virtex-II-Pro, Virtex-4, and Virtex-5 are not compatible with low-power devices because of their high power consumption.

Figure 9. Performance comparison in terms of AT product.

Figure 10. Performance comparison in terms of throughput per slice.

6. Conclusions

In this paper, a high-performance ECC processor has been proposed exploiting unified PA on Edwards25519 curve to perform SPA-resistant point multiplication. An efficient ECPM module has been designed in projective coordinates, which supports 256-bit point multiplication over a prime field. Unified PA is adopted for the ECPM module to provide strong protection against SPA attacks and reduce the area required by an additional PD module. To perform high-speed modular multiplication, an efficient radix-4 interleaved modular multiplier has been proposed. The proposed ECC processor performs fast point multiplication with a considerably lower area use, providing high resistance against SPA. Because of its less hardware resource requirements and high computation speed, it is well suited for resource-constrained IoT devices. Since it provides a faster ECPM that is a rising demand of elliptic curve-based digital signature schemes, it could be manipulated in Bitcoin-like cryptocurrencies for high-speed digital signature generation and verification, which would reduce latency in transaction confirmation. Based on the overall performance analyses, it can be concluded that the proposed ECC processor could be a good choice for the IoT security as well as the emerging technology “Blockchain”.

Author Contributions: All the authors contributed to this paper. Specifically, M.M.I. and M.S.H. proposed the idea and programmed the design; M.M.I. analyzed and verified the implementation results and wrote the paper; M.K.H. and M.S. reviewed and edited the paper; and Y.M.J. supervised the work and provided funding support. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the Ministry of Science and ICT (MSIT), Korea, under the Information Technology Research Center (ITRC) support program (IITP-2018-0-01396) supervised by the Institute for Information & communication Technology Promotion (IITP).

Acknowledgments: As the first author of this paper, I would like to thank my beloved mother and brother who always supported me in every success and failure in my life.

Conflicts of Interest: This manuscript has not been published elsewhere and is not under consideration by another journal. We have approved the manuscript and agree with submission in *Sensors*. There are no conflict of interest to declare.

References

1. Ding, S.; Li, C.; Li, H. A novel efficient pairing-free CP-ABE based on elliptic curve cryptography for IoT. *IEEE Access* **2018**, *6*, 27336–27345. [[CrossRef](#)]
2. Hankerson, D.; Menezes, A.; Vanstone, S. *Guide to Elliptic Curve Cryptography*; Springer: New York, NY, USA, 2004.

3. ElGamal, T. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Inf. Theory* **1985**, *31*, 469–472. [CrossRef]
4. Rivest, R.L.; Shamir, A.; Adleman, L. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* **1978**, *21*, 120–126. [CrossRef]
5. Diffie, W.; Hellman, M. New directions in cryptography. *IEEE Trans. Inf. Theory* **1976**, *22*, 644–654. [CrossRef]
6. Liu, Z.; Huang, X.; Hu, Z.; Khan, M.K.; Seo, H.; Zhou, L. On emerging family of elliptic curves to secure internet of things: ECC comes of age. *IEEE Trans. Dependable Secur. Comput.* **2017**, *14*, 237–248. [CrossRef]
7. Challa, S.; Wazid, M.; Das, A.K.; Kumar, N.; Reddy, A.G.; Yoon, E.-J.; Yoo, K.-Y. Secure signature-based authenticated key establishment scheme for future IoT applications. *IEEE Access* **2017**, *5*, 3028–3043. [CrossRef]
8. Lara-Nino, C.A.; Diaz-Perez, A.; Morales-Sandoval, M. Elliptic curve lightweight cryptography: A survey. *IEEE Access* **2018**, *6*, 72514–72550. [CrossRef]
9. Lee, Y.; Sakiyama, K.; Batina, L. Elliptic-curve-based security processor for RFID. *IEEE Trans. Comput.* **2008**, *57*, 1514–1527. [CrossRef]
10. Liao, Y.; Hsiao, C. A secure ECC-based RFID authentication scheme integrated with ID-verifier transfer protocol. *Ad Hoc Netw.* **2014**, *18*, 133–146. [CrossRef]
11. Chou, J. An efficient mutual authentication RFID scheme based on elliptic curve cryptography. *J. Supercomput.* **2014**, *70*, 75–94. [CrossRef]
12. Zhang, Z.; Qi, Q. An efficient RFID authentication protocol to enhance patient medication safety using elliptic curve cryptography. *J. Med. Syst.* **2014**, *38*, 5. [CrossRef] [PubMed]
13. Zhao, Z. A secure RFID authentication protocol for healthcare environments using elliptic curve cryptosystem. *J. Med. Syst.* **2014**, *38*, 5. [CrossRef] [PubMed]
14. He, D.; Zeadally, S. An analysis of RFID authentication schemes for internet of things in healthcare environment using elliptic curve cryptography. *IEEE Internet Things J.* **2015**, *2*, 72–83. [CrossRef]
15. Bernstein, D.J.; Duif, N.; Lange, T.; Schwabe, P.; Yang, B.Y. High-speed high-security signatures. *J. Cryptogr. Eng.* **2012**, *2*, 77–89. [CrossRef]
16. Liusvaara, I.; Josefsson, S. Edwards Curve Digital Signature Algorithm (EdDSA). In Internet-Draft: Draft-irtf-cfrg-eddsa-05, Internet Engineering Task Force, 2017. Available online: <https://tools.ietf.org/html/rfc8032> (accessed on 1 January 2017).
17. Liu, J.; Zhang, Z.; Chen, X.; Kwak, K. Certificateless remote anonymous authentication schemes for wireless body area networks. *IEEE Trans. Parallel Distrib. Syst.* **2014**, *25*, 332–342. [CrossRef]
18. He, D.; Zeadally, S.; Kumar, N.; Lee, J.-H. Anonymous authentication for wireless body area networks with provable security. *IEEE Syst. J.* **2017**, *11*, 2590–2601. [CrossRef]
19. Saeed, M.E.S.; Liu, Q.-Y.; Tian, G.; Gao, B.; Li, F. Remote authentication schemes for wireless body area networks based on the Internet of Things. *IEEE Internet Things J.* **2018**, *5*, 4926–4944. [CrossRef]
20. Keoh, S.L.; Kumar, S.S.; Tschofenig, H. Securing the Internet of Things: A standardization perspective. *IEEE Internet Things J.* **2014**, *1*, 265–275. [CrossRef]
21. Liu, Z.; Grossschadl, J.; Hu, Z.; Jarvinen, K.; Wang, H.; Verbauwhede, I. Elliptic curve cryptography with efficiently computable endomorphisms and its hardware implementations for the Internet of Things. *IEEE Trans. Comput.* **2017**, *66*, 773–785. [CrossRef]
22. Banerjee, U.; Wright, A.; Juvekar, C.; Waller, M.; Arvind, A.; Chandrakasan, A.P. An Energy-Efficient Reconfigurable DTLS Cryptographic Engine for Securing Internet-of-Things Applications. *IEEE J. Comput.* **2019**, *54*, 2339–2352. [CrossRef]
23. Islam, M.M.; Hossain, M.S.; Hasan, M.K.; Shahjalal, M.; Jang, Y.M. FPGA implementation of high-speed area-efficient processor for elliptic curve point multiplication over prime field. *IEEE Access* **2019**, *7*, 178811–178826. [CrossRef]
24. Brier, E.; Joye, M. Weierstraß elliptic curves and side-channel attacks. In *Public Key Cryptography (LNCS)*; Springer: Heidelberg, Germany, 2002; Volume 2274, pp. 335–345.
25. Joye, M. Elliptic curves and side-channel analysis. *ST J. Syst. Res.* **2003**, *4*, 17–21.
26. Edwards, H.M. A normal form for elliptic curves. *Bull. Am. Math. Soc.* **2007**, *44*, 393–422. [CrossRef]
27. Bernstein, D.J.; Lange, T. Faster addition and doubling on elliptic curves. In *Proceedings of the Advances in Cryptology (LNCS)*; Springer: Heidelberg, Germany, 2007; Volume 4833, pp. 29–50.

28. Hisil, H.; Wong, K.K.H.; Carter, G.; Dawson, E. Twisted Edwards curves revisited. In *Proceedings of the Advances in Cryptology (LNCS)*; Springer: Heidelberg, Germany, 2008; Volume 5350, pp. 326–343.
29. Bernstein, D.J.; Birkner, P.; Lange, T.; Peters, C. Twisted Edwards curves. In *Proceedings of the Advances in Cryptology (LNCS)*; Springer: Heidelberg, Germany, 2008; Volume 5023, pp. 389–405.
30. Bernstein, D.J. Curve25519: New Diffie-Hellman speed records. In *Proceedings of the Public Key Cryptography (LNCS)*; Springer: Heidelberg, Germany, 2006; Volume 3958, pp. 207–228.
31. Baldwin, B.; Moloney, R.; Byrne, A.; McGuire, G.; Marnane, W.P. A hardware analysis of twisted Edwards curves for an elliptic curve cryptosystem. In *Proceedings of the Reconfigurable Computing: Architectures Tools and Applications (LNCS)*; Springer: Heidelberg, Germany, 2009; Volume 5453, pp. 355–361.
32. Abdulrahman, E.A.H.; Masoleh, A.R. New regular radix-8 scheme for elliptic curve scalar multiplication without pre-computation. *IEEE Trans. Comput.* **2015**, *64*, 438–451. [\[CrossRef\]](#)
33. Islam, M.M.; Hossain, M.S.; Shahjalal, M.; Hasan, M.K.; Jang, Y.M. Area-time efficient hardware implementation of modular multiplication for elliptic curve cryptography. *IEEE Access* **2020**, *8*, 73898–73906. [\[CrossRef\]](#)
34. Asif, S.; Hossain, M.S.; Kong, Y. High-throughput multi-key elliptic curve cryptosystem based on residue number system. *IET Comput. Digit. Tech.* **2017**, *11*, 165–172. [\[CrossRef\]](#)
35. Hossain, M.S.; Kong, Y.; Saeedi, E.; Vayalil, N. High-performance elliptic curve cryptography processor over NIST prime fields. *IET Comput. Digit. Tech.* **2016**, *11*, 33–42. [\[CrossRef\]](#)
36. Shah, Y.A.; Javeed, K.; Azmat, S.; Wang, X. Redundant signed digit based high-speed elliptic curve cryptographic processor. *J. Circuits Syst. Comput.* **2018**, *28*, 1950081. [\[CrossRef\]](#)
37. Marzouqi, H.; Al-Qutayri, M.; Salah, K.; Schinianakis, D.; Stouraitis, T. A high-speed FPGA implementation of an RSD-based ECC processor. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2016**, *24*, 151–164. [\[CrossRef\]](#)
38. Marzouqi, H.; Al-Qutayri, M.; Salah, K. An FPGA implementation of NIST 256 prime field ECC processor. In *Proceedings of the IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*, Abu Dhabi, UAE, 8–11 December 2013; pp. 493–496.
39. Liu, Z.; Liu, D.; Zou, X. An efficient and flexible hardware implementation of the dual-field elliptic curve cryptographic processor. *IEEE Trans. Ind. Electron.* **2017**, *64*, 2353–2362. [\[CrossRef\]](#)
40. Hu, X.; Zheng, X.; Zhang, S.; Cai, S.; Xiong, X. A low hardware consumption elliptic curve cryptographic architecture over GF(p) in embedded application. *Electronics* **2018**, *7*, 104. [\[CrossRef\]](#)
41. Javeed, K.; Wang, X. Low latency flexible FPGA implementation of point multiplication on elliptic curves over GF(p). *Int. J. Circuit Theory Appl.* **2016**, *45*, 214–228. [\[CrossRef\]](#)
42. Javeed, K.; Wang, X. FPGA based high-speed SPA-resistant elliptic curve scalar multiplier architecture. *Int. J. Reconfigurable Comput.* **2016**, *2016*, 1–10. [\[CrossRef\]](#)
43. Javeed, K.; Wang, X.; Scott, M. High performance hardware support for elliptic curve cryptography over general prime field. *Microprocess. Microsyst.* **2017**, *51*, 331–342. [\[CrossRef\]](#)
44. Ghosh, S.; Alam, M.; Chowdhury, D.R.; Gupta, I.S. Parallel crypto-devices for GF(p) elliptic curve multiplication resistant against side-channel attacks. *Comput. Electr. Eng.* **2009**, *35*, 329–338. [\[CrossRef\]](#)
45. Ananyi, K.; Alrimeih, H.; Rakhmatov, D. Flexible hardware processor for elliptic curve cryptography over NIST prime fields. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2009**, *17*, 1099–1112. [\[CrossRef\]](#)
46. Loi, K.C.C.; Ko, S.B. Scalable elliptic curve cryptosystem FPGA processor for NIST prime curves. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2015**, *23*, 2753–27. [\[CrossRef\]](#)
47. Sakiyama, K.; Mentas, N.; Batina, L.; Preneel, B.; Verbauwhede, I. Reconfigurable modular arithmetic logic unit for high-performance public-key cryptosystems. In *Proceedings of the Reconfigurable Computing: Architectures and Applications (LNCS)*; Springer: Heidelberg, Germany, 2006; Volume 3985, pp. 347–357.
48. Ghosh, S.; Mukhopadhyay, D.; Roychowdhury, D. Petrel: Power and timing attack resistant elliptic curve scalar multiplier based on programmable GF(p) arithmetic unit. *IEEE Trans. Circuits Syst. I-Regul. Pap.* **2011**, *58*, 1798–1812. [\[CrossRef\]](#)
49. Lee, J.-W.; Chung, S.C.; Chang, H.C.; Lee, C.Y. Efficient power-analysis-resistant dual-field elliptic curve cryptographic processor using heterogeneous dual-processing-element architecture. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2014**, *22*, 49–61. [\[CrossRef\]](#)
50. Mcivor, C.J.; Mcloone, M.; Mccanny, J.V. Hardware elliptic curve cryptographic processor over GF(p). *IEEE Trans. Circuits Syst. I-Fundam. Theor. Appl.* **2006**, *53*, 1946–1957. [\[CrossRef\]](#)

51. Lai, J.Y.; Huang, C.-T. High-throughput cost-effective dual-field processors and the design framework for elliptic curve cryptography. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2008**, *16*, 1567–1580.
52. Schinianakis, D.M.; Fournaris, A.P.L.; Michail, H.E.; Kakarountas, A.P.; Stouraitis, T. An RNS implementation of an F_p elliptic curve point multiplier. *IEEE Tran. Circuits Syst. I-Regul. Pap.* **2009**, *56*, 1202–1213. [[CrossRef](#)]
53. Esmaeildoust, M.; Schinianakis, D.; Javashi, H.; Stouraitis, T.; Navi, K. Efficient RNS implementation of elliptic curve point multiplication over $GF(p)$. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2013**, *21*, 1545–1549. [[CrossRef](#)]

© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).