Chinese Journal of Electronics
Vol.32, No.6, Nov. 2023

A Time-Area-Efficient and Compact ECSM Processor over GF(p)

HE Shiyang[1], LI Hui[1], LI Qingwen[1], and LI Fenghua[1,2]

(1. *Xidian University, Xi'an 710126, China*)

(2. *Chinese Academy of Sciences, Beijing 100093, China*)

Abstract — The elliptic curve scalar multiplication (ECSM) is the core of elliptic curve cryptography (ECC), which directly determines the performance of ECC. In this paper, a novel time-area-efficient and compact design of a 256-bit ECSM processor over GF(p) for the resource-constrained device is proposed, where p can be selected flexibly according to the application scenario. A compact and efficient 256-bit modular adder/subtractor and an improved 256-bit Montgomery multiplier are designed. We select Jacobian coordinates for point doubling and mixed Jacobian-affine coordinates for point addition. We have improved the binary expansion algorithm to reduce 75% of the point addition operations. The clock consumption of each module in this architecture is constant, which can effectively resist side-channel attacks. Reuse technology is adopted in this paper to make the overall architecture more compact and efficient. The design architecture is implemented on Xilinx Kintex-7 (XC7K325T-2FFG900I), consuming 1439 slices, 2 DSPs, and 2 BRAMs. It takes about 7.9 ms at the frequency of 222.2 MHz and 1763k clock cycles to complete once 256-bit ECSM operation over GF(p).

Key words — Elliptic curve encryption, Finite field, Montgomery multiplication, Field programmable gate array, Side-channel attacks.

I. Introduction

With the continuous development of global informatization, network security has become a crucial issue in the current Internet world. Maintaining the regular order of network society and ensuring the confidentiality and integrity of information transmission and storage in the network is a significant problem that has been studied in the field of information security. Asymmetric cryptography or public-key cryptography (PKC) [1] is an essential branch of modern cryptography and the cornerstone of all information security systems. As one of the most widely used PKC algorithms, elliptic curve cryptography (ECC) [2] requires minimal resources with the advantage of high security. At the same security level, the key length required by ECC is shorter than that of RSA [3]. Hence, ECC is widely used in systems and devices with limited hardware resources, such as wireless communication devices, smart IC cards, Internet of things (IoT), smart home appliances, wearable devices, etc. The performance of ECC systems depends on elliptic curve scalar multiplication (ECSM) operations, which are critical to ECC. Therefore, it is an extensive research field to achieve high computing performance of the ECSM processor and optimize its area-time (AT) value.

The implementation of ECSM can be completed over GF(2^m) or GF(p). As the arithmetic logic unit (ALU) of the ECSM processor based on GF(2^m) is composed of XOR operation, and the operation speed is faster than that based on GF(p), most of the proposed works are completed over GF(2^m), such as works [4]–[6]. The ALU of the ECSM processor over GF(p) is composed of modular operation, so it can also calculate RSA, which is composed of modular operation as well. Hence, the ECSM processor over GF(p) is more suitable for embedded systems and IoT security devices, such as near-field communication (NFC). In addition, ECC on GF(p) is more secure under the same key size. Therefore, this paper implements an ECSM processor based on GF(p).

Some ECSM processors implemented over GF(p) have been proposed with good performance [7]–[10]. However, most of them are based on curves with a spe-

Manuscript Received Aug. 9, 2022; Accepted Dec. 20, 2022. This work was supported by the National Key R&D Program of China (2022YFB3103400), the National Natural Science Foundation of China (61732022), the Province Key R&D Program of Shaanxi (2019 ZDLGY12-09), the Mobile Internet Security Innovation Team of Shaanxi Province (2018TD-007), and the China 111 Project (B16037).

 DOI:10.23919/cje.2022.00.267

cial prime structure, that is, ECSM can only be calculated over a specific prime field. Although ECSM based on this method has a faster computing speed, it tends to be not very flexible in the size and parameters of the key. The computing speed of processors supporting the implementation over any prime field may often be slow, but in practical application, some security technologies, such as supersingular isogeny key exchange (SIKE) algorithm, do not support the implementation over a specific field. Therefore this paper aims to design a general ECSM framework that can support ECSM calculation over any GF(p) and ensure a low value of AT.

The software implementation of ECSM is low cost and flexible, but it cannot meet the speed of password operation required by the network security equipment. ECSM hardware implementation can be achieved via the field-programmable gate array (FPGA) [11]–[13] and the application-specific integrated circuit (ASIC) [14]. While the implementation of ASIC is highly effective and requires fewer material resources, it lacks flexibility and universality. Once the ASIC designs determined, it can no longer be changed again. FPGA has a powerful programmable capability compared to ASIC, therefore it is preferable to use FPGA to implement ECSM [15].

Side-channel attack (SCA) is one of the main threats affecting the security of encryption hardware [16]. The core idea of SCA is to obtain ciphertext from various leaked information generated during hardware operations [17]. Some works [18], [19] use binary expansion algorithm in the implementation of ECSM. In this algorithm, whether the point addition (PA) operation is performed depends on the binary string of the key, so this method has low immunity to SCAs, such as the power analysis attack.

In this paper, we present a time-area-efficient and compact design of a 256-bit ECSM processor over GF(p), and we implement it on Xilinx Kintex-7 (XC7K325T-2FFG900I) FPGA with the aim to reduce resource consumption and make it possible to apply to the platform with limited resources. The main contributions of this paper are as follows.

1) A compact and efficient 256-bit modular adder and subtractor based on DSP units are proposed to complete the modular addition and subtraction calculation in ECSM, in which the modulus p can be selected flexibly according to the application scenario.

2) An improved 256-bit Montgomery multiplier based on two DSP units and some peripheral circuits is proposed. It can efficiently complete the multiplication calculation in ECSM.

3) This scheme makes full use of reuse technology, modular adder/subtractor, and Montgomery multiplier reuse two DSP units. In the process of ECSM, the utilization rate of two DSP units can almost reach 100%, further improving the resource utilization rate.

4) Jacobian coordinates and mixed Jacobian-affine coordinates are respectively selected for point doubling (PD) and point addition (PA) with the lowest computational complexity to replace the PA and PD in affine coordinates.

5) An improved binary expansion method is proposed to calculate the point addition of elliptic curve. Using pre-calculation and table lookup methods, 75% point addition calculation can be avoided.

6) In this design, the clock cycle consumption of all modules is constant, which can effectively resist SCAs.

The remainder of this paper is organized as follows. Section II will provide the basic theory of elliptic curve cryptosystem and a brief description of Montgomery multiplication. In Section III, we will first introduce our newly designed process element (PE) operation unit architecture, including the improved Montgomery multiplier and modular adder/subtractor. Then, the implementation of PD and PA modules will be introduced, respectively. Additionally, the overall architecture will be summarized. In Section IV, the result and the comparison are given. Finally, the conclusions are drawn in Section V.

II. Preliminaries

The overall structure of ECC is illustrated in Fig.1, following a top-down structure. The top layer includes ECC protocols, such as elliptic curve digital signature algorithm (ECDSA) [20] and elliptic curve digital Hellman key exchange (ECDH). The second layer is elliptic curve scalar multiplication implemented by calling the PA and PD operation within the group operations layer. At the same time, the PA and PD consist of modular addition, subtraction, multiplication, and inversion in the field operations layer.

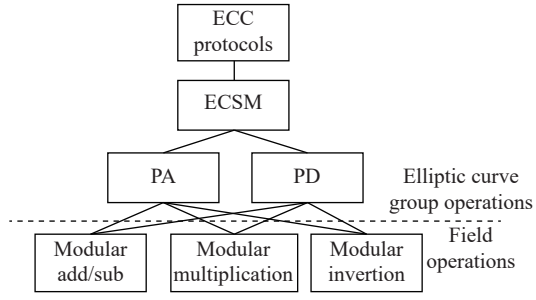

Fig. 1. This is the structure of ECC algorithm.

1. Elliptic curve

Elliptic curve refers to a curve determined by a cubic equation defined on the field. In the affine coordinates, the simplified Weierstrass equation is used to rep-

resent the elliptic curve over the prime field GF(p), which is shown in equation (1).

$$y^2 = x^3 + ax + b \tag{1}$$

where

$$x, y, a, b \in \mathrm{GF}(p) \tag{2}$$

and

$$4a^3 + 27b^2 \neq 0 \tag{3}$$

In addition to using affine coordinates, elliptic curves based on GF(p) can also be represented by using Jacobian or projective coordinates. In affine coordinates, PA and PD operations need a lot of modular division operation, the cost of which is enormous. Hence, Jacobian coordinates can be selected to avoid a large number of modular division operations.

In Jacobian coordinates, elliptic curves over GF(p) can be expressed as

$$y^2 z = x^3 + axz^4 + bz^6 \tag{4}$$

When $Z \neq 0$, the Jacobian point (X, Y, Z) of elliptic curve can be transformed with the affine point (x, y) as

$$X = xz^2, \;\; Y = yz^3, \;\; Z = 1 \tag{5}$$

and

$$x = X/z^2, \;\; y = Y/z^3 \tag{6}$$

2. Group operation in Jacobian coordinates

The definition of double point $2P(x_1, y_1, z_1)$ of Jacobian coordinate point $P(x_0, y_0, z_0)$ is shown as equation (7).

$$\begin{aligned} x_1 &= {T_1}^2 - 8x_0{T_2}^2 \\ y_1 &= T_1(4x_0{T_2}^2 - x_1) - 8{T_2}^2 \\ z_1 &= 2y_0z_0 \end{aligned} \tag{7}$$

where

$$T_1 = 3{x_0}^2 + a{z_0}^4, \;\; T_2 = {y_0}^2 \tag{8}$$

The point addition algorithm in this paper will use the mixed Jacobian-affine coordinates, that is to say, the Jacobian point $P(x_0, y_0, z_0)$ is added to the affine point $Q(x_1, y_1)$, and the result $P + Q = (x_2, y_2, z_2)$ is defined as shown in equation (9).

$$\begin{aligned} x_2 &= {T_2}^2 - (x_1{z_0}^2 + x_0){T_1}^2 \\ y_2 &= T_2(x_1{T_1}^2 - X_3) - y_1{T_1}^3 \\ z_2 &= T_1z_0 \end{aligned} \tag{9}$$

where

$$T_1 = x_1{z_0}^2 - x_0, \;\; T_2 = y_1{z_0}^2 - y_0 \tag{10}$$

3. Montgomery multiplication

Compared with modular addition and subtraction, modular multiplication in finite field operations is very complex and is also the bottleneck of Jacobian coordinates ECSM. The commonly used modular multiplication algorithms include Mersenne prime modular multiplication and Montgomery multiplication (MM). The former is a kind of modular multiplication algorithm that is mainly intended for Mersenne prime numbers. By using the properties of this kind of number, the algorithm converts one-time multiplication and one-time division of modular multiplication into partial modular addition and subtraction. However, the disadvantages of the algorithm are also apparent. The algorithm can only be used in the specific prime field and can not be applied to modular multiplication over other prime fields with variable bit width. In modular multiplication, MM converts the modular operation into shift operation, which effectively reduces the computational complexity of modular multiplication and can be used for different finite fields, which is also easy to implement in software and hardware. Considering operation efficiency and flexibility, MM is selected as the algorithm of the modular multiplier in this design. The definition of Montgomery modular multiplication is shown in formula (11).

$$r = \mathrm{MM}(a, b) = a \cdot b \cdot R^{-1} \bmod p \tag{11}$$

where $R = 2^n$, and n is the bit length of p.

It can be seen that when a and b perform MM, the calculation result is not $a \cdot b \bmod p$. Therefore, we introduce the concept of Montgomery field. For the numbers a and b over the prime field are respectively represented by $\widehat{a} = a \cdot R$ and $\widehat{b} = b \cdot R$ in the Montgomery domain, then the result of MM($\widehat{a}$,$\widehat{b}$) is

$$\widehat{r} = \mathrm{MM}(\widehat{a}, \widehat{b}) = a \cdot b \cdot R \bmod p \tag{12}$$

where $\widehat{r}$ is the representation of r in Montgomery Field.

The transformation between prime field and Montgomery field can also be calculated by MM, as is shown in formulas (13) and (14) respectively.

$$\widehat{a} = \mathrm{MM}(\widehat{a}, R^2) = a \cdot R \bmod p \tag{13}$$

$$a = \mathrm{MM}(\widehat{a}, 1) = a \bmod p \tag{14}$$

4. DSP48E1

DSP element in 7 Series FPGA, the DSP48E1 slice, has great flexibility and utilization, which can improve the efficiency of application while reducing the overall

energy consumption and improving the maximum frequency. The high performance of DSP48E1 allows designers to use the time-multiplexing method to realize multiple slow operations in a single DSP48E1 slice.

The DSP48E1 slice supports many independent functions, such as multiplication, accumulation multiplication (MACC), addition multiplication, barrel displacement, wide-bus multiplexing, and scanning detection. The architecture can also form broad math functions, DSP filters, and complex algorithms by cascading multiple DSP48E1 slices, general FPGA logic is not required.

The basic functionality of the DSP48E1 slice is illustrated in Fig.2 [21]. It should be mentioned that DSP also has many salient functionalities, including power saving pre adder, 25 × 18 two's complement multiplier and pattern detector.

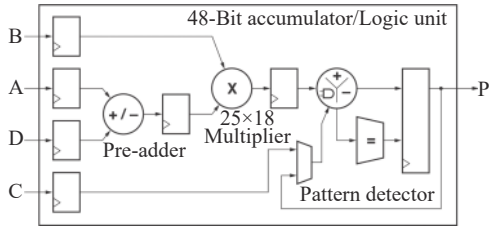

Fig. 2. This is the basic DSP48E1 slice functionality.

III. Hardware Implementation

In this section, the details of modular adder/subtractor and improved 256-bit Montgomery multiplier are explained, and the calculation paths of PA and PD are introduced. We also propose an improved binary expansion method to avoid 75% PA calculations. The overall hardware architecture is described as a conclusion.

1. Modular adder and subtractor

The direct implementation of modular addition or subtraction with 256-bit width can bring fast computing speed, but the resource consumption is huge. We divide each 256-bit width operand into 16 operands with 16-bit width. Operands a, b and modulus p are expressed as

$$a=\sum_{j=0}^{15}a_j2^{16j},\quad b=\sum_{j=0}^{15}b_j2^{16j},\quad p=\sum_{j=0}^{15}p_j2^{16j} \tag{15}$$

The pseudo-code of modular addition and subtraction algorithm is shown in **Algorithm 1** and **Algorithm 2**. s_0 and s_1 respectively represent carry-bit and borrow-bit in the modular addition algorithm, which is the opposite situation in the modular subtraction algorithm. The algorithm consumes 16 cycles to complete a 256-bit width modular addition or subtraction.

Algorithm 1 Modular addition

Require:

$a=\sum_{j=0}^{15}a_j2^{16j}$, $b=\sum_{j=0}^{15}b_j2^{16j}$, $p=\sum_{j=0}^{15}p_j2^{16j}$.

Ensure:

$a+b \bmod p$.

$c=0, d=0,\ s_0=0, s_1=0;$
for $j=0;\ j<16;\ j++$ **do**
    $(s_0,c_j)=a_j+b_j+s_0;$
    $(s_1,d_j)=c_j-p_j-s_1;$
end for
if $d<0$ **then**
    return c
else
    return d
end if

Algorithm 2 Modular subtraction.

Require:

$a=\sum_{j=0}^{15}a_j2^{16j}$, $b=\sum_{j=0}^{15}b_j2^{16j}$, $p=\sum_{j=0}^{15}p_j2^{16j}$.

Ensure:

$a-b \bmod p$.

$c=0, d=0,\ s_0=0, s_1=0;$
for $j=0;\ j<16;\ j++$ **do**
    $(s_0,c_j)=a_j-b_j-s_0;$
    $(s_1,d_j)=c_j+p_j+s_1;$
end for
if $c<0$ **then**
    return d
else
    return c
end if

The hardware architecture of modular adder and subtractor proposed in this paper is shown in Fig.3. In order to make the design architecture more compact, two integrated DSP units are used for addition and subtraction calculation of 16-bit width. DSP1 is used to complete the addition/subtraction and carry/borrow of operands a_j and b_j, while DSP2 is used to complete the addition/subtraction and carry/borrow of DSP1 calculation result and module p_j. Some shift registers are used to form the pipeline and store the intermediate value. It consumes 18 clock cycles to perform one 256-bit width addition or subtraction. By using DSP units, we can not only choose modulus p more flexibly but also reduce the usage of logic resources.

2. Montgomery multiplier

Montgomery multiplication is the most time-consuming operation in the implementation of ECSM [22]. The classical Montgomery algorithm shown in **Algorithm 3** requires large bit width multiplication, which will consume many hardware resources. We propose an efficient improved Montgomery multiplication based on

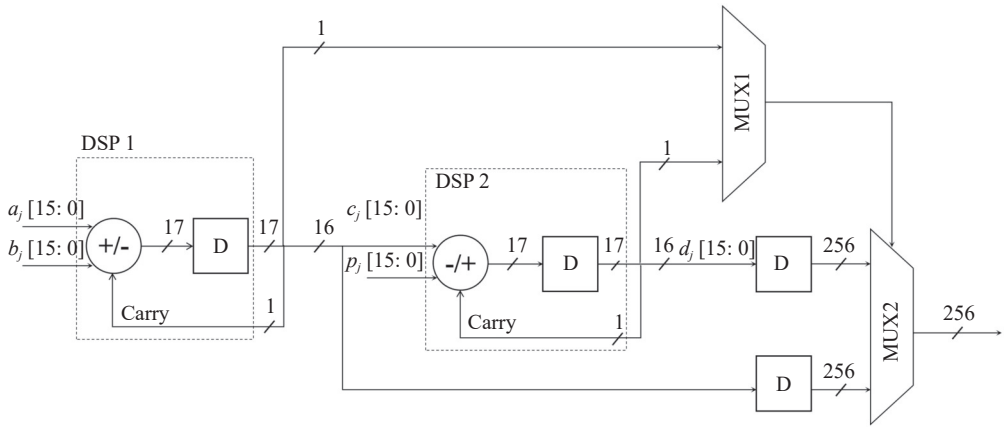

Fig. 3. This is the architecture of modular adder/subtractor.

the iterative algorithm, as is shown in **Algorithm 4**. In this method, we also divide each 256-bit width operand into 16 operands with 16-bit width, and the integrated DSP units are used to complete the calculation of 16-bit width. The hardware architecture of the Montgomery multiplier is more compact at the cost of adding some iterative cycles.

Algorithm 3 Classical montgomary multiplication

Require:

$a,\ b,\ p,\ \omega$.

Ensure:

$r = a \cdot b \cdot R^{-1} \bmod p$.

1: $c = 0, r = 0$;
2: $c = a \cdot b$;
3: $r = (c + (c \cdot \omega \bmod R) \cdot p)/R$;
4: **if** $r \geq p$ **then**
5: $r{=}r{-}p$;
6: **end if**
7: **return** r

Algorithm 4 Improved montgomary multiplication

Require:

$a{=}\sum_{m=0}^{15} a_m 2^{16m}, b{=}\sum_{m=0}^{15} b_m 2^{16m}, p{=}\sum_{m=0}^{15} p_m 2^{16m}, \omega$.

Ensure:

$r = a \cdot b \cdot R^{-1} \bmod p$.

1: $r = 0, c = 0, d = 0$;
2: **for** $i = 0;\ i \leq m-1;\ i++$ **do**
3: $u_i = (r_0 + b_i \cdot a_0) \cdot \omega \bmod 2^{16}$;
4: **for** $j = 0;\ j \leq m-1;\ j++$ **do**
5: $c_{j+1}c_j = b_i a_j + c_j$;
6: $d_{j+1}d_j = u_i p_j + d_j$;
7: **end for**
8: **for** $j = 0;\ j \leq m-1;\ j++$ **do**
9: $(\text{carry}, r_j) = r_j + c_j + d_j + \text{carry}$;
10: **end for**
11: $r = r/2^{16}$;
12: **end for**
13: **if** $r \geq p$ **then**
14: **for** $i = 0;\ i \leq m-1;\ i++$ **do**
15: $(\text{carry}, r_i) = r_i - p_i - \text{carry}$;
16: **end for**
17: **end if**
18: **return** r

Two DSP units constitute the hardware architecture core of the improved Montgomery multiplier, as is shown in Fig.4. These two DSP units can complete the addition and multiplication calculation of 16-bit width in the Montgomery multiplier. Some shift registers are used to construct the pipeline structure and store the calculated intermediate value. A few peripheral circuits are responsible for process control.

The calculation process of the improved Montgomery multiplier is divided into four stages.

1) Calculate u_i. DSP1 calculates $r_0 + b_i \cdot a_0$ and DSP2 calculates $(r_0 + b_i \cdot a_0) \cdot \omega$, which is completed after two clock cycles.

2) Calculate c and d. DSP1 and DSP2 calculate c and d respectively and complete the calculation after 16 clock cycles.

3) Calculate $c + d + r$, DSP1 calculate $c + d$, DSP2 calculate $c + d + r$. The calculation is completed after 16 clock cycles.

4) Cycle step 1) to step 3) 16 times, and finally calculate r mod p and output the result.

The improved Montgomery multiplier consumes about 528 clock cycles to complete a 256-bit width multiplication calculation. The hardware architecture mainly comprises DSP units, registers, and peripheral circuits. Although the clock cycle consumption is relatively large, the circuit structure is simple and compact.

The modular adder/subtractor module and the improved Montgomery multiplier module reuse two DSP units and some peripheral control circuits together. In the process of ECSM calculation, the utilization rate of

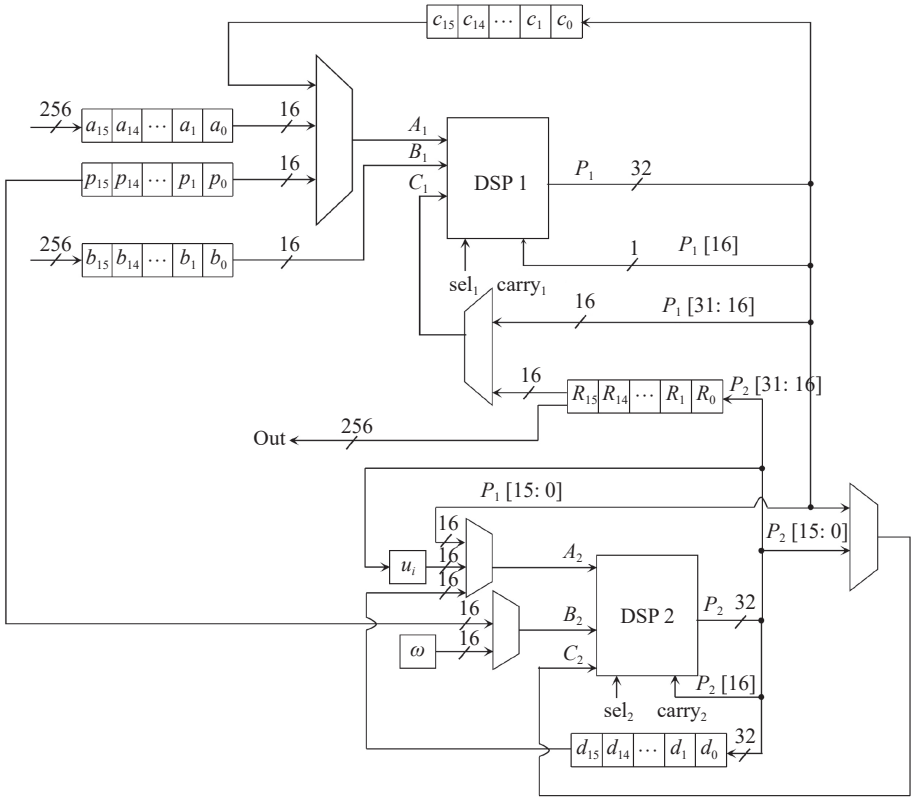

Fig. 4. This is the architecture of improved Montgomery multiplier.

the reused two DSP units can almost reach 100%, which significantly improves the resource utilization and the compactness of the scheme.

3. PA and PD

To avoid modular division, which is not user-friendly for hardware, this paper selects Jacobian coordinates for PD and Jacobian-affine mixed coordinates for PA with the lowest computational complexity to replace the PA and PD in affine coordinates. As a result, the points for PD and one of the points for PA are in the Jacobian coordinate system. At the same time, another point involved in PA points happens to be the input affine point, so there is no need to convert Jacobian points into affine points by modulus inverse. Thus, point multiplication can continuously carry out PD on Jacobian coordinates and PA on mixed Jacobian-affine coordinates.

Based on the definition of PD in Jacobian coordinates, as is shown in formula (9), this paper designs a serial circuit that performs the calculation steps of PD one by one to improve the utilization of hardware resources and reduce the complexity. The operation path based on Jacobian PD is shown in Fig.5.

In Fig.5, R_1–R_6 represent the intermediate variable register, which is used to store the value of the initial input of the PD and the value of the intermediate variable generated in the calculation process. Each addition, subtraction, or multiplication operation will change the value of the corresponding register, so it is necessary to plan the application of each register reasonably.

According to the hardware architecture of PD, 10 modular multiplications, 13 modular additions/subtractions, and 6 intermediate variable registers are required.

Similar to the PD, this paper designs a circuit for serial execution according to the definition of PA in mixed Jacobian-affine coordinates. The hardware architecture of PA in mixed Jacobian-affine coordinates is shown in Fig.6.

According to the path to calculate PA, 11 times modular multiplication, 7 times modular addition or subtraction, and 7 intermediate variable registers are required.

4. Improved binary expansion algorithm

Elliptic curve scalar multiplication is the core part of ECC, which is implemented by calling PD and PA. ECSM is the most expensive operation in ECC, so the efficiency of ECSM determines the performance of ECC.

The most common algorithm of calculating ECSM is binary expansion algorithm shown in **Algorithm 5**. First, the 256-bit number k is converted into binary form and expressed as $k = \sum_{j=0}^{255} k_j 2^j$. Then scan the binary form of k bit by bit from high to low until the end of the last bit of k, perform once PD in each cycle, and perform PA if $k_j = 1$ in this cycle.

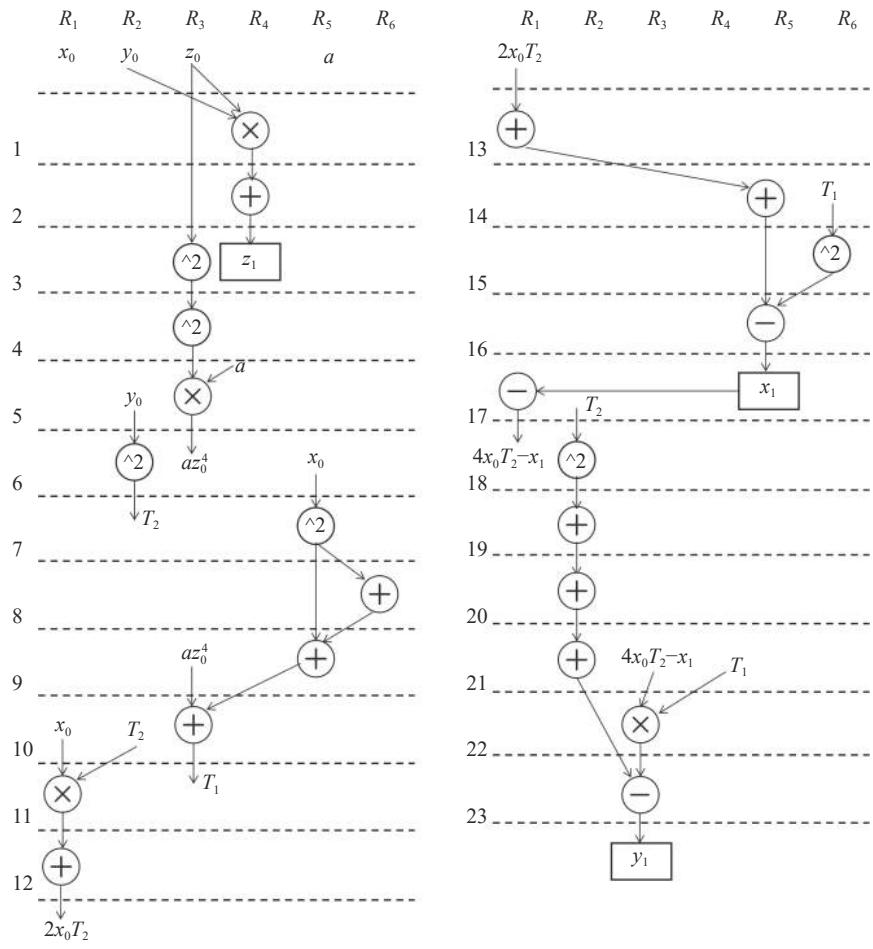

Fig. 5. This is the hardware architecture of PD in Jacobian coordinates.

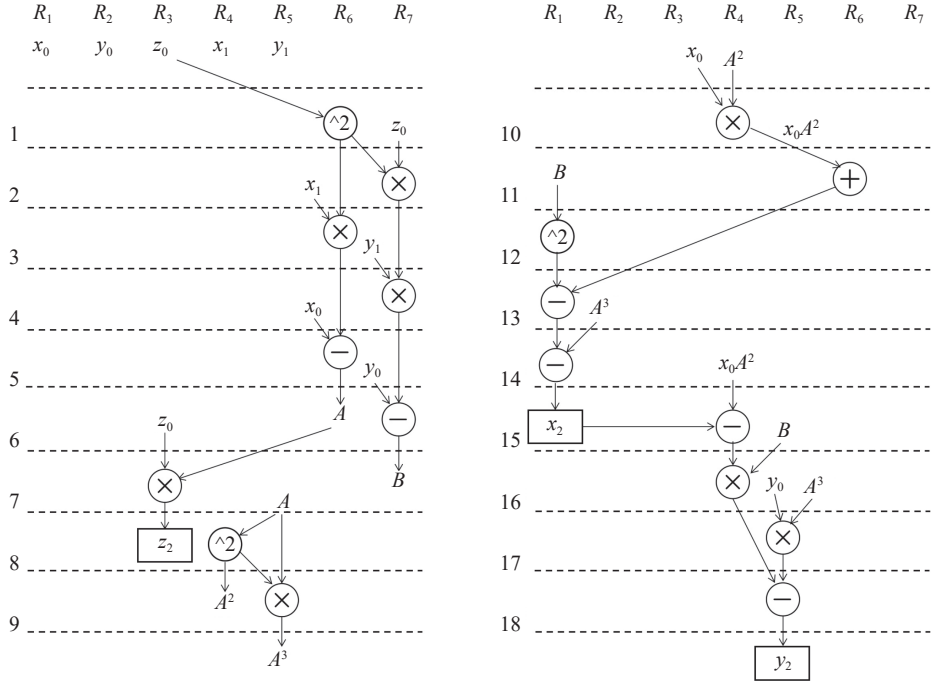

Fig. 6. This is the hardware architecture of PA in mixed Jacobian-affine coordinates.

Algorithm 5 Binary expansion algorithm

Require:

$k = \sum_{j=0}^{255} k_j 2^j$, ECC point P.

Ensure:

kP.

1: $Q = 0$;
2: **for** $j = 255$; $j \geq 0$; j- - **do**
3: $Q = 2Q$;
4: **if** $k_j = 1$; **then**
5: $Q = Q + P$;
6: **end if**
7: **end for**
8: **return** Q

Based on the binary expansion algorithm, this paper proposes a scheme to reduce the number of PA operations in the binary expansion algorithm by pre-calculation, making the calculation of ECSM more efficient. The result of $\{P, 2P, \cdots, 15P\}$ needs to be pre-calculated and stored in the ROM.

The specific algorithm is shown in the **Algorithm 6**. Initially, it still converts k into binary form. The difference is that the improved algorithm divides every four bits of the binary form of k into one data block, so the integer 256-bit number k will be divided into 64 data blocks, which are recorded as $k = \sum_{j=0}^{63} K_j 2^{4j}, K_j \in [0, 15]$. Next, scan each K_j from high to low until the scanning to K_0 ends the cycle. In each cycle, the PD is executed four times, and the point corresponding to the current K_j is added.

Algorithm 6 Improved binary expansion algorithm

Require:

$k = \sum_{j=0}^{63} K_j 2^{4j}, \{0, P, 2P, \cdots, 15P\}$.

Ensure:

kP.

1: $Q = 0$;
2: **for** $j = 63$; $j \geq 0$; j- - **do**
3: $Q = 2^4 Q$;
4: $Q = Q + K_j P$;
5: **end for**
6: **return** Q

Fig.7 shows the hardware design scheme of the improved binary expansion algorithm based on Algorithm 6. The input for each cycle corresponds to the output of PA module in the previous round stored in Reg Q. The final result is the output of the last cycle of PA.

In the improved binary expansion algorithm, there are mainly two modules PD and PA to consume power. The power of PA is higher than that of PD and the operation time is relatively longer. If the power or time analysis is carried out when the hardware is running, the binary form of key k can be easily divulged. So the simple binary expansion algorithm can not resist SCAs.

However, the improved binary expansion method proposed in this paper, no matter how much the value

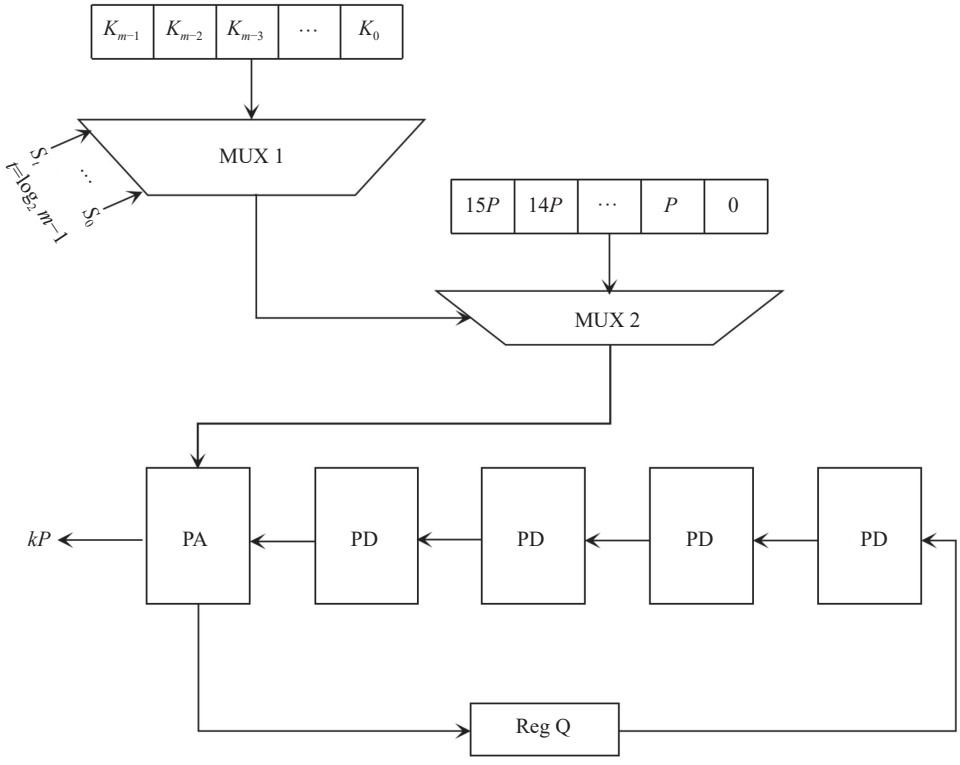

Fig. 7. This is the architecture of poposed ECSM processor.

of the k is, needs four times PD and once PA operation. The improved binary expansion algorithm will not divulge the key k under power and time analysis, and can well resist SCAs. Moreover, because the improved algorithm mainly performs PD, the operation time also has certain superiority with the same key length.

The improved binary expansion algorithm avoids part of PA by pre-calculating. And it can be seen that the time of PD operations used in calculating 256-bit ECSM by this algorithm is 252, and the time of PD operations used is 63.

In comparison to other SCAs-resistant algorithms, it reduces the number of PA operations by about 75%. More precisely, other ECSM algorithms must perform PA and PD operations within each loop to avoid revealing k in order to achieve SCAs resistance. Therefore, when processing 256-bit ECSM, other processors that can withstand SCAs must execute 255 times PD and 255 times PA operations. In contrast, our algorithm reduces the number of PA operations by approximately 75% and PD operations by multiple times.

5. Overall hardware architecture of ECSM

The overall hardware architecture of ECSM is shown in Fig.8. The details of each module are introduced in the previous article. Table 1 lists the resource consumption of each module.

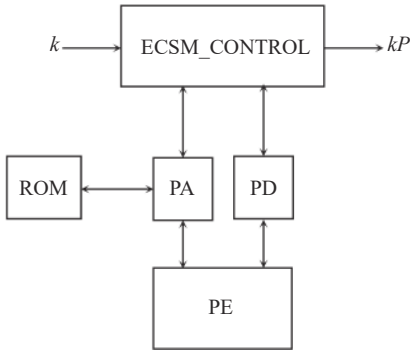

Fig. 8. This is the the overall hardware architecture of ECSM.

Table 1. The resource consumption of each module

Module	Slice	DSP
ECSM_CONTROL	48	0
PA&PD	262	0
PE	1129	2

In Fig.8, the top layer is ECSM_CONTROL module, which is mainly responsible for the overall control and scheduling of other modules to complete the scalar multiplication calculation in ECC. Firstly, the scalar k involved in ECSM is input to the module, then it operates by calling the PA and PD module in the next layer, and finally outputs the result of ECSM as kP.

PA and PD modules are respectively responsible for calculating point addition and doubling. PA module looks up the table by calling the ROM module and performs the underlying calculation by calling the process element (PE) module to complete PA calculation. The ROM module stores the pre-calculated results, that is, the 15 values of points $\{P, 2P, \ldots, 15P\}$. PD module only needs to call PE module to complete PD calculation. PA and PD modules save resource consumption and improve utilization by time-sharing multiplexing the PE module.

The PE module comprises the modular adder/subtractor and the improved Montgomery multiplier described above. They reused two DSP units to complete 256-bit modular addition, subtraction, and Montgomery multiplication. In the calculation process, the utilization rate of the two DSP units can almost reach 100%, which further improves the compactness of the hardware architecture and resource utilization.

IV. Result and Comparison

The proposed design has been described by Verilog HDL and implemented using the Xilinx Vivado 2019.1 software, simulated by the Xilinx Vivado simulator. The designs are synthesized, mapped, placed, and routed on the Xilinx Kintex-7 (XC7K325T-2FFG900I). We aim to present a time-area-efficient and compact design of a 256-bit ECSM processor over GF(p), reduce resource consumption, and make it possible to apply to the platform with limited resources.

This work costs 1439 slices, 2 DSPs, and 2 BRAMs. It takes about 7.9 ms at the frequency of 222.2 MHz and 1763k clock cycles with a throughput of 32.3 kbps to complete a 256-bit ECSM operation over GF(p), where p can be flexibly selected according to the application scenario. PE module is the bottom operation unit of ECSM, which is responsible for calculating modular addition/subtraction and Montgomery modular multiplication. The PE module consumes 532 slices and 2 DSPs. It takes about 18 clock cycles to complete a 256-bit modular addition/subtraction and 528 clock cycles to complete a 256-bit Montgomery modular multiplication. It requires 13 modular additions/subtractions and 10 Montgomery modular multiplications to complete a 256-bit PD operation in the PD module on Jacobian coordinates. And it requires 7 modular additions/subtractions and 10 Montgomery modular multiplications as well to complete a 256-bit PA operation in the PA module on mixed Jacobian-affine coordinates. The clock cycles consumed by the PD module and PA module to complete a PD operation and a PA operation are 5514 and 5934, respectively. In the ECSM_CONTROL module, 63 PA operations and 252 PD operations are re-

quired to complete a 256-bit ECSM. Due to the application of improved ECSM architecture and reuse technology, the area-time value of this scheme has obvious advantages over other schemes. Moreover, since the clock cycles consumed by each module are constant, the design architecture can effectively resist SCAs.

Table 2 lists the performance and resource consumption comparison between the elliptic curve scalar multiplication design proposed in this paper and other ECSM processors designed over the 256-bit prime field on FPGA. Although the throughput of this article is low compared to other works, this paper has great advantages in terms of the AT value and resource consumption.

Table 2. Comparison between our ECSM design and similar work over GF(p)

Reference	Publish year	Platform	Freq. (MHz)	Slice	DSP	Speed (ms)	Throughput (kbps)	AT[a]	AT[b]	Support arbitrary p
Ours	2023	Kintex-7	222.2	1.4k	2	7.90	32.30	11.06	15.8	yes
Islam *et al.* [23]	2019	Virtex-7	177.7	8.9k	0	1.48	173.20	13.17	0	no
Kudithi *et al.* [24]	2020	Kintex-7	122.8	7.4k	0	2.44	104.90	18.06	0	no
Amiet *et al.* [19]	2016	Virtex-7	225.0	1.7k	20	1.49	171.81	2.53	29.8	yes
Wu *et al.* [25]	2019	Virtex-4	162.0	21.9k	16	1.01	68.52	22.12	16.2	yes
Hossain *et al.* [18]	2016	Kintex-7	121.5	11.3k	0	3.27	78.28	36.95	0	no
Asif *et al.* [26]	2017	Virtex-7	72.9	24.2k	0	2.96	1816.20	71.63	0	yes

Note: [a] Slice cost multiplied by scalar multiplication delay; [b] DSP cost multiplied by scalar multiplication delay.

Islam *et al.* [23] proposed an efficient ECSM design on special curves Edwards25519, which can achieve fast scalar multiplication with high security. The design uses Montgomery ladder algorithm to resist SCAs and utilize operation parallelization to decrease the latency and the number of arithmetic modules. However, the modulus p in Edwards25519 is fixed, so the processor cannot flexibly select p according to requirements. In addition, the area-time product of their design is slightly bigger than ours.

The work in reference [24] shows a hardware structure of ECSM on Jacobian coordinates, in which PA and PD modules share resources. It is implemented both in FPGA and ASIC. On Virtex-7 FPGA platform, the ECSM processor needs 2.44 ms to complete a 256-bit prime ECSM, which is three times as fast as the processor proposed in this paper. However, the processor uses 7.4k slices, five times as many as ours. In addition, the processor does not provide any protection against SCAs, as their ECSM module applies the binary expansion algorithm.

Amiet *et al.* [19] designed a hardware architecture that supports arbitrary prime fields up to 1024 bits and different standards of ECSM. The processor needs 1.49 ms to realize 256-bit multiplication over the prime field with 20 DSPs and 6816 LUTs, which has excellent performance. Their design is about a quarter of ours in terms of AT value. However, they use 10 times as many DSPs as ours.

Wu *et al.* [25] proposed a scalable hardware implementation multiplier, and a fast unified architecture for ECSM over five NIST primes. At the same time, the Montgomery ladder with projective coordinate is used in the design to reduce the computational complexity and provide the ability to resist SCAs. It takes 1.57 ms to calculate a 256-bit ECSM, taking 21638 slices and 32 DSPs, which has relatively low performance compared with ours in terms of DSP usage and AT value.

Hossain *et al.* [18] presented an efficient hardware implementation of a new PA and PD combined architecture for ECSM processor. The design supports 224-bit and 256-bit prime numbers recommended by NIST. It takes 4.7 ms to complete a 256-bit ECSM in affine coordinates on Xilinx Kintex-7 FPGA, slightly faster than our design. However, their design area is nearly 7.5 times that of ours.

Asif *et al.* [26] proposed a multi-key ECSM based on the residue number system, which employes deep pipelining to allow simultaneous encryption of 21 keys. Their implementation on Virtex-7 FPGA took the advantages of RNS-based implementation, whereas it expends 17.3 times more slices and is exactly 7 times less efficient than our ECSM processor.

V. Conclusions

This paper presents a time-area-efficient and compact design of a 256-bit ECSM processor over GF(p), where p can be selected flexibly according to the application scenario. Improved Montgomery modular multiplier and modular adder/subtracter are proposed. They reuse 2 DSP units to make their hardware architecture more compact and efficient. In ECSM processor, we use Jacobian coordinates for the PD operation and mixed Jacobian-affine coordinates for the PA operation to further improve the calculation efficiency. An improved binary expansion method is proposed to calculate the scalar multiplication of elliptic curve. We avoid 75% of the point addition calculations using pre-calculation and table lookup method. The design architecture is imple-

mented on Xilinx Kintex-7 (XC7K325T-2FFG900I) and it consums 1439 slices, 2 DSPs, and 2 BRAMs, which makes it suitable for resource-constrained devices. It takes about 7.9 ms at the frequency of 222.2 MHz and 1763k clock cycles to complete a 256-bit ECSM operation over GF(p).

References

[1] T. Elgamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*, vol.31, no.4, pp.469–472, 1985.

[2] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of Computation*, vol.48, no.177, pp.203–209, 1987.

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

[4] J. Y. Lai and C. T. Huang, "A highly efficient cipher processor for dual-field elliptic curve cryptography," *IEEE Transactions on Circuits and Systems II:Express Briefs*, vol.56, no.5, pp.394–398, 2009.

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

[6] C. Rebeiro and D. Mukhopadhyay, "High speed compact elliptic curve cryptoprocessor for FPGA platforms," in *Proceedings of the 9th International Conference on Cryptology in India*, Kharagpur, India, pp.376–388, 2008.

[7] N. Guillermin, "A high speed coprocessor for elliptic curve scalar multiplications over Fp," in *Proceedings of the 12th International Workshop on Cryptographic Hardware and Embedded Systems*, Santa Barbara, CA, USA, pp.48–64, 2010.

[8] J. Y. Lai, Y. S. Wang, and C. T. Huang, "High-performance architecture for elliptic curve cryptography over prime fields on FPGAs," *Interdisciplinary Information Sciences*, vol.18, no.2, pp.167–173, 2012.

[9] G. Chen, G. Q. Bai, and H. Y. Chen, "A high-performance elliptic curve cryptographic processor for general curves over GF(p) based on a systolic arithmetic unit," *IEEE Transactions on Circuits and Systems II:Express Briefs*, vol.54, no.5, pp.412–416, 2007.

[10] J. F. Fan, K. Sakiyama, and I. Verbauwhede, "Elliptic curve cryptography on embedded multicore systems," *Design Automation for Embedded Systems*, vol.12, no.3, pp.231–242, 2008.

[11] 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 Transactions on Circuits and Systems I:Regular Papers*, vol.58, no.8, pp.1798–1812, 2011.

[12] H. Marzouqi, M. Al-Qutayri, and K. Salah, "An FPGA implementation of NIST 256 prime field ECC processor," in *2013 IEEE 20th International Conference on Electronics, Circuits, and Systems (ICECS)*, Abu Dhabi, United Arab Emirates, pp.493–496, 2013.

[13] C. J. McIvor, M. Mcloone, and J. V. Mccanny, "Hardware elliptic curve cryptographic processor over $rmGF(p)$," *IEEE Transactions on Circuits and Systems I:Regular Papers*, vol.53, no.9, pp.1946–1957, 2006.

[14] M. Machhout, Z. Guitouni, K. Torki, *et al.*, "Coupled FPGA/ASIC implementation of elliptic curve cryptoprocessor," *International Journal of Network Security & its Applications*, vol.2, no.2, pp.100–112, 2010.

[15] T. Y. Li, F. Zhang, W. Guo, *et al.*, "A survey: FPGA-based dynamic scheduling of hardware tasks," *Chinese Journal of Electronics*, vol.30, no.6, pp.991–1007, 2021.

[16] K. K. Wu, H. Y. Li, D. J. Zhu, *et al.*, "Efficient solution to secure ECC against side-channel attacks," *Chinese Journal of Electronics*, vol.20, no.3, pp.471–475, 2011.

[17] P. C. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," in *Proceedings of the 16th Annual International Cryptology Conference*, Santa Barbara, CA, USA, pp.104–113, 1996.

[18] S. Hossain, Y. N. Kong, E. Saeedi, *et al.*, "High-performance elliptic curve cryptography processor over NIST prime fields," *IET Computers & Digital Techniques*, vol.11, no.1, pp.33–42, 2017.

[19] D. Amiet, A. Curiger, and P. Zbinden, "Flexible FPGA-based architectures for curve point multiplication over GF(p)," in *Proceedings of the 2016 Euromicro Conference on Digital System Design (DSD)*, Limassol, Cyprus, pp.107–114, 2016.

[20] M. D. Zhu, X. Qin, L. Wang, *et al.*, "A time-to-digital-converter utilizing bits-counters to decode carry-chains and DSP48E1 slices in a field-programmable-gate-array," *Journal of Instrumentation*, vol.16, no.2, 2021.

[21] W. Yu, K. P. Wang, B. Li, *et al.*, "Montgomery algorithm over a prime field," *Chinese Journal of Electronics*, vol.28, no.1, pp.39–44, 2019.

[22] G. Locke and P. Gallagher, FIPS PUB 186-3 Digital signature standard (DSS), Federal Information Processing Standards Publication, 2009, article no.186.

[23] M. Islam, S. Hossain, M. K. Hasan, *et al.*, "FPGA implementation of high-speed area-efficient processor for elliptic curve point multiplication over prime field," *IEEE Access*, vol.7, pp.178811–178826, 2019.

[24] T. Kudithi and R. Sakthivel, "An efficient hardware implementation of the elliptic curve cryptographic processor over prime field," *International Journal of Circuit Theory and Applications*, vol.48, no.8, pp.1256–1273, 2020.

[25] T. Wu and R. M. Wang, "Fast unified elliptic curve point multiplication for NIST prime curves on FPGAS," *Journal of Cryptographic Engineering*, vol.9, no.4, pp.401–410, 2019.

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

HE Shiyang received the M.S. degree in telecommunications engineering from Xidian University, China, in 2016. He is currently working toward the Ph.D. degree at the School of Cyber Engineering, Xidian University, Xi'an, China. His research interests include cryptographic algorithm, hardware speedup and field-programmable gate array architectures and applications. (Email: syhe@xidian.edu.cn)

LI Hui (corresponding author) received the B.S. degree from Fudan University in 1990, M.S. and Ph.D. degrees from Xidian University in 1993 and 1998. Since June 2005, he has been a Professor in the School of Cyber Engineering, Xidian University, Xi'an, China. His research interests are in the areas of cryptography, wireless network security, information theory, hardware security, and network coding. He is a Chair of ACM SIGSAC China. He served as the Technique Committee Chair or Co-chair of several conferences. He has published more

than 170 international academic research papers on information security and privacy preservation.
(Email: lihui@mail.xidian.edu.cn)

LI Qingwen received the B.S. degree in information security from Xidian University, Xi'an, China, in 2022. She is currently working toward the M.S. degree at the School of Cyber Security, Xidian University, Xi'an, China. Her research interests include cryptographic algorithm. (Email: liqwww1017@163.com)

LI Fenghua received the B.S. degree in computer software, M.S. and Ph.D. degrees in computer systems architecture from Xidian University, Xi'an, China, in 1987, 1990, and 2009, respectively. He is currently a Professor and a Doctoral Supervisor with the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China. He is also a Doctoral Supervisor with the Xidian University. His research interests include network security, system security, privacy computing, and cryptographic processors. (Email: lfh@iie.ac.cn)