Hindawi
Mathematical Problems in Engineering
Volume 2021, Article ID 6633925, 8 pages
https://doi.org/10.1155/2021/6633925

Research Article

A High-Speed Elliptic Curve Cryptography Processor for Teleoperated Systems Security

Yong Xiao, Weibin Lin, Yun Zhao, Chao Cui, and Ziwen Cai

Electric Power Research Institute of CSG, Guangzhou, Guangdong, 510663, China

Correspondence should be addressed to Yong Xiao; xiaoyong@csg.cn

Received 15 October 2020; Revised 8 November 2020; Accepted 26 November 2020; Published 23 January 2021

Academic Editor: Zhan Li

Teleoperated robotic systems are those in which human operators control remote robots through a communication network. The deployment and integration of teleoperated robot's systems in the medical operation have been hampered by many issues, such as safety concerns. Elliptic curve cryptography (ECC), an asymmetric cryptographic algorithm, is widely applied to practical applications because its far significantly reduced key length has the same level of security as RSA. The efficiency of ECC on GF (p) is dictated by two critical factors, namely, modular multiplication (MM) and point multiplication (PM) scheduling. In this paper, the high-performance ECC architecture of SM2 is presented. MM is composed of multiplication and modular reduction (MR) in the prime field. A two-stage modular reduction (TSMR) algorithm in the SCA-256 prime field is introduced to achieve low latency, which avoids more iterative subtraction operations than traditional algorithms. To cut down the run time, a schedule is put forward when exploiting the parallelism of multiplication and MR inside PM. Synthesized with a 0.13 um CMOS standard cell library, the proposed processor consumes 341.98k gate areas, and each PM takes 0.092 ms.

1. Introduction

In teleoperated robotic systems, human operators, often geographically distant, interact with and control robots through a communication network. Teleoperated robotic systems have many applications such as bomb disposal, search and rescue, robotic surgery, and medical operation. Teleoperated robotic surgery is a particularly important application of medical operation. Expert surgery is able to be performed remotely and without direct human presence. It is expected to have a significant impact on the quality of medical services in isolated regions, battlefields, or disaster areas. With the development of teleoperated systems and robots, the deployment and integration of teleoperated robots in the medical operation have encountered many problems such as safety concerns [1], time delay [2], and bilateral control [3]. Security is one of the biggest issues that hamper the deployment and integration of teleoperated robots and there are some works on it [4].

Telerobotic surgery is expected to be employed in extreme conditions, where teleoperated robots may have to operate in harsh and low-power conditions, connecting to the Internet with potential loss. As depicted in Figure 1, the last communication link may even be a wireless link to a drone or a satellite, providing the connection to a trusted facility (possibly a large hospital with an established infrastructure) [5].

In such operating conditions, the security of the long-range control is significant, since if the teleoperated robotics are attacked by hackers, potential damage might be caused due to loss of proper control. Besides, verifying that these requirements are established and maintained during a teleoperated procedure is necessary [6].

In harsh conditions, low-power and time delay are significant. Hence, the security process, like digital signature/verification and encryption/decryption, should be implemented by hardware acceleration. Compared with software implementation, hardware implementation has many advantages, such as high efficiency, low power consumption, and safety. ECC is a kind of public key cryptography algorithm that can provide these security processes, proposed in 1986 by Miller [7] and Koblitz [8]. It

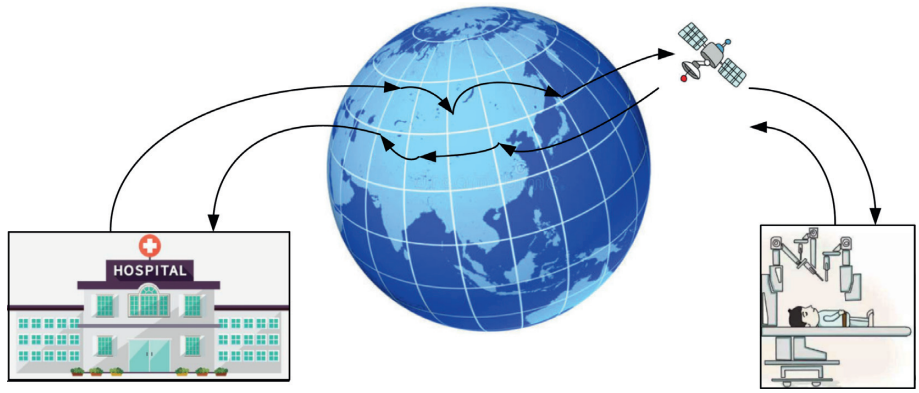

Figure 1: Visualization of a typical telerobotic surgery setup.

has been demonstrated to be used as an alternative to the classical RSA [9] thanks to its significantly reduced key lengths [10]. ECC when using 160–256 bits provides similar security compared with RSA or discrete logarithm schemes over finite fields (1024–4096 bits) [11]. SM2, as an ECC algorithm, was included in ISO/IEC14888-3/AMD1 in November 2017.

Considerable efforts have been made to implement the ECC with hardware as can be noticed in [12–22], during which MM operation is widely used for PM in ECC. In order to accelerate the MM, the proposed designs should be considered into three categories [23]: (1) the recommended prime modular multiplication algorithm, (2) Montgomery multiplication algorithm, and (3) the interleaved modular multiplication algorithm. Among those three categories, the first category is the fastest and it is limited by the specific prime field, such as NIST and SCA-256. The architecture in [12] equips Montgomery multiplier among 8-bit × 8-bit to 64-bit × 64-bit aiming to improve area efficiency and reduce delay at the cost of retarding speed. The designs in [9, 20] are based on the recommended prime modular multiplication algorithm. However, those MR algorithms only contain one stage, which will generate an intermediate result Z, such as $Z[0, 14p)$ in [9] and $Z[-4p, 5p)$ in [20]. Besides, an extra calculation is required to get the final result $Z[0, p)$. Notably, the architecture in [9] adopts a full-word 256-bit × 256-bit multiplier, and all the calculations are executed in the SCA-256 prime field. In MR operation of design [9], 13 subtractions are taken to transfer the intermediate value Z ($0 \le Z < 14p$) to the final value in the most needed situation, following with large latency.

Traditional software methods to implement cryptography algorithms will bring larger time delay and power consumption. However, hardware implementation can resolve these issues. Motived to provide highly efficient safety assurance for teleoperated systems, we realize ECC by hardware implementation. The main contributions of this paper include the following:

We propose a high-performance hardware processor, which adopts a half-word multiplier to improve performance while reducing hardware consumption. Compared with most of the other works, it has a better trade-off between performance and hardware overhead.

The TSMR algorithm in SCA-256 is proposed to implement low latency. The algorithm obtains the intermediate result Z $[0, 2p)$, which requires one subtraction to get the final result Z $[0, p)$. Compared with the traditional method [9] which obtains intermediate result Z $[0, 14p)$, our method avoids lots of subtractions to get the final result.

TSMR algorithm is implemented by a carry-save adder architecture to reduce latency and hardware overhead. Combined with Karatsuba-Ofman (KO) multiplication algorithm and pipeline design, MM requires an average of five clock cycles, even though one clock cycle for modular reduction and five clock cycles for multiplication are required.

The arrangement of this paper is as follows. In Section 2, the elliptic curve and PM are introduced. In Section 3, high-performance architecture is illustrated. Then, the proposed method is implemented and validated in Section 4. Finally, in Section 5, the conclusion of this work is provided.

2. Mathematical Background

2.1. Elliptic Curve. A nonsupersingular elliptic curve (EC) over GF (p) is defined as a set of points (x, y) that conform to equation (1), also known as the Weierstrass equation, and an infinity point additionally:

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

where a and b are parameters, identifying the EC which satisfied $4a^3 + 27b^2 \neq 0 \pmod p$.

2.2. Point Multiplication. PM describes a transformation that k identical EC points add up to one, denoted as a scalar times an EC point "kP," where $k = (k_{l-1} \cdots k_0)$, and l represents the binary length of k. In this work, the width NAF addition-subtraction method [24], given in Algorithm 1, is applied to point multiplication.

Input: width w, scalar k, EC point p
Output: EC point $Q = kP$
(1) Precomputation: $i \in \{1, 3, \ldots, 2^{w-1} - 1\}$, $P[i] = iP$
(2) Compute NAF$w(k) = \sum_{i=0}^{l-1} k_i 2^i$
(3) $Q = \infty$
(4) for i from $l - 1$ downto 0 do
  $Q = 2Q$
  if $k_i \neq 0$ then
  if $k_i > 0$ then $Q = Q + P[k_i]$
  else $Q = Q - P[-k_i]$
(5) Return Q

Algorithm 1: Width NAF addition-subtraction method.

PM operation is the elemental operation of ECC and is performed as a sequence of elliptic curve addition (ECADD) and elliptic curve doubling (ECDBL). Let EC point $P_i = (X_i, Y_i, Z_i)$; the ECADD is defined as $P_3 = P_1 + P_2$ and ECDBL is defined as $P_3 = 2P_1$. To avoid time-consuming modular inversion/division operation, ECADD reaches the fastest efficiency in mixed affine-Jacobian coordinates, while there is ECDBL in Jacobian coordinates [25].

ECADD in mixed affine-Jacobian coordinates and ECDBL in Jacobian coordinates are given in the two following equations:

$$\begin{cases} X_3 = \left(Y_2 Z_1^3 - Y_1\right)^2 - \left(X_2 Z_1^2 - X_1\right)^2 \left(X_1 + X_2 Z_1^2\right), \\ Y_3 = \left(Y_2 Z_1^3 - Y_1\right)\left(X_1 \left(X_2 Z_1^2 - X_1\right)^2 - X_3\right) - Y_1 \left(X_2 Z_1^2 - X_1\right)^3, \\ Z_3 = \left(X_2 Z_1^2 - Z_1\right) Z_1, \end{cases} \tag{2}$$

$$\begin{cases} X_3 = \left(3X_1^2 + aZ_1^4\right)^2 - 8X_1 Y_1^2, \\ Y_3 = \left(3X_1^2 + aZ_1^4\right)\left(4X_1 Y_1^2 - X_3\right) - 8Y_1^4, \\ Z_3 = 2Y_1 Z_1, \end{cases} \tag{3}$$

3. High-Performance Architecture of SM2

The PM architecture based on full-word multipliers is described below. TSMR and full-word multiplication constitute MM, while the binary modular inversion algorithm in [26] was applied to execute modular inversion (MI) operation.

3.1. Modular Reduction. SCA-256 has the characteristic that it can be denoted as $p = 2^{256} - 2^{224} - 2^{96} + 2^{64} - 1$. The traditional MR for SCA-256 [9] is given in Algorithm 2. After the fast reduction operation, the intermediate value can be represented as

$$Z = s_1 + s_2 + 2s_3 + 2s_4 + 2s_5 + s_6 + s_7 + s_8 + s_9 + 2s_{10} - s_{11} - s_{12} - s_{13} - s_{14}, \tag{4}$$

where $Z \in [0, 14p)$. It will cost at most 13 subtractions to get the final result $Z \in [0, p)$. Since the modular reduction would be computed in a single clock cycle, the repetitive subtractions have a significant influence on the latency and bring about a lot of hardware resources consumption.

A TSMR algorithm on SCA-256 is proposed in this paper to address this problem (Algorithm 3). The first state takes sixteen addition/subtraction operations to calculate Z_1, while the second one just costs two to calculate Z_2. The intermediate value after two state fast reduction operations is $Z_2 = s_1 + s_{16} - s_{17}$, where $Z_2 \in [0, 2p)$, and it only needs one subtraction at most to obtain the final value $Z \in [0, p)$.

In ECADD or ECDBL operation, modular addition (MA) or modular subtraction (MS) operations are always required by the following MM operation. One cycle can be reduced when MA/MS was carried out. The max delay of carry-save addition only cares about the final carry. Therefore, adding one value to the other twenty values will not have a huge impact on latency. As shown in Algorithm 3, operand a in previous MA/MS is added to $(c + a) \bmod p$. In Algorithm 5 proposed below, such an operation appears twice in ECADD (Step 9: T2T2-T4, Step 11: T1T2-T4) and in ECDBL (Step 6: T2T2-T1, Step 8: T1T2-T5), respectively. The clock cycles, $m/(w+1) * 2 + m * 2 = 256/(4+1) * 2 + 26 * 2 = 614$, are reduced.

3.2. Carry-Save Adder Architecture. In TSFR algorithm, there are five subtraction operations in Z_1 and one in Z_2. In order to reduce the area consumption and clock latency, a kind of new carry-save adder (CSA) architecture is presented for Algorithm 3, and the main advantage of CSA is that it can deal with subtraction operation. The subtraction operation becomes an addition operation by using the subtrahend's complement.

The first stage reduction result Z_1, $0 \leq Z_1 < 16p$, was designed as 261-bit data, and it contains 21 operands and 20 256-bit CSAs. Due to one extended sign bit for five subtrahends' complement, as shown in Figure 2, it is noted that the 20 most significant bits (MSBs) of CSA cannot be cumulated. The CSA of 261 or more bits is not met. As shown in Figure 2, the MSB of $Z_1[261]$ could not be got from the sum of sc14[260] to sc21[260]. However, the 256-th to 260-th bits of subtrahend's complement are set to 1, while the 257-th to 261-th bits of addend are set to 0. The sum of the 256-th to 260-th bits of the subtrahend can be precalculated, getting $5 * 5'b11111 = 7'b1011011$. Only the low 5 bits (5'b11011) are needed, and it can be placed in row 1 of 1-bit CSA. In this case, the proposed CSA is completed with the function of settling the subtraction operations.

The first stage reduction operation architecture can be divided into two parts: the left part is a 1-bit CSA and the right part is a 32-bit CSA, as shown in Figure 3. To simplify the analysis, 1-bit CSA (1 full adder) is presented by a thin rectangle on the left, while a 32-bit CSA composed of 32 1-bit CSAs is presented by a wider rectangle on the right. For example, the subtraction operation in row 15 of the 32-bit CSA, $s_{12} = (0, 0, 0, 0, 0, c_9, 0, c_8)$, is represented by $-s_{12} = \sim s_{12} + 1 = (\overline{0}, \overline{0}, \overline{0}, \overline{0}, \overline{0}, \overline{c_9}, \overline{0}, \overline{c_8}) + 1$, where $\overline{0} = 32'hFFFFFFFF$, $\overline{c_8} = \sim c_8$, and $1 = 32'h1$. The 32-bit CSA consists of 20 rows and 8 columns, which compute the result $Z_1[255:0]$. The 1-bit CSA is featured with 5 columns,

Input: Integer $c = (c_{15}, c_{14}, \ldots, c_0)$ in base 2^{32}; $c \in [0, p^2 - 1]$.
Output: c mod p
(1) $s1 = (c7, c6, c5, c4, c3, c2, c1, c0)$, $s2 = (c15, c14, c13, c12, c11, 0, c9, c8)$,
$s3 = (c14, 0, c15, c14, c13, 0, c14, c13)$, $s4 = (c13, 0, 0, 0, 0, 0, c15, c14)$,
$s5 = (c12, 0, 0, 0, 0, 0, 0, c15)$, $s6 = (c11, c11, c10, c15, c14, 0, c13, c12)$,
$s7 = (c10, c15, c14, c13, c12, 0, c11, c10)$, $s8 = (c9, 0, 0, c9, c8, 0, c10, c9)$,
$s9 = (c8, 0, 0, 0, c15, 0, c12, c11)$, $s10 = (c15, 0, 0, 0, 0, 0, 0, 0)$,
$s11 = (0, 0, 0, 0, 0, c14, 0, 0)$, $s12 = (0, 0, 0, 0, 0, c13, 0, 0)$,
$s13 = (0, 0, 0, 0, 0, c9, 0, 0)$, $s14 = (0, 0, 0, 0, 0, c8, 0, 0)$
$Z = s1 + s2 + s3 + 2s4 + 2s5 + s6 + s7 + s8 + s9 + s10 -$
$s11 - s12 - s13 - s14$
(2) Return Z mod p

Algorithm 2: Traditional modular reduction algorithm in SCA-256.

Input: integera a and $c = (c_{15}, c_{14}, \ldots, c_0)$ in base 2^{32}; $a \in [0, p - 1], c \in [0, p^2 - 1]$.
Output: $(c+a)$ mod p
(1) $s1 = (c7, c6, c5, c4, c3, c2, c1, c0)$; $s2 = (c15, 0, 0, 0, 0, 0, 0, c8)$;
$s3 = (c14, 0, 0, c14, c14, 0, c14, c14)$; $s4 = (c13, 0, 0, 0, c13, 0, c13, c13)$;
$s5 = (c12, 0, c15, 0, 0, 0, c15, c15)$; $s6 = (c11, c11, c13, c13, c11, 0, c11, c11)$;
$s7 = (c10, c15, c10, 0, 0, 0, c10, c10)$; $s8 = (c9, c14, c14, c15, c15, 0, c9, c9)$;
$s9 = (c8, 0, 0, c9, c8, 0, 0, 0)$; s10 = (0, 0, 0, c12, c12, 0, c12, c12);
s11 = (0, 0, 0, 0, c14, c14, 0, 0); s12 = (0, 0, 0, 0, 0, c9, 0, c8);
s13 = (0, 0, 0, 0, 0, c13, c13, 0); s14 = (0, 0, 0, 0, 0, c8, 0, c8);
$Z1 = s1 + 3s2 + 2s3 + 2s4 + 2s5 + s6 + s7 + s8 + s9 + s10 - s11 - s12 - s13 - s14 - a + p = (r8, r7, r6, r5, r4, r3, r2, r1, r0)$
(2) s15 = (r7, r6, r5, r4, r3, r2, r1, r0); s16 = (r8, 0, 0, 0, r8, 0, 0, r8); s17 = (0, 0, 0, 0, 0, r8, 0, 0);
$Z2 = s15 - s16 - s17$;
$Z3 = Z2 - p$;
If $Z3 \geq 0$, return Z3
else return Z2

Algorithm 3: Two-stage modular reduction algorithm in SCA-256 (TSMR).

Input: A: 256-bit integer, satisfy $A = a_1 \times 2^{128} + a_0$
B: 256 bit integer, satisfy $B = b_1 \times 2^{128} + b_0$.
Output: C: 512 bit product, satisfy $C = A \times B$.
(1) $P_{00} = a_0 \times b_0$; $a_{sum} = a_0 + a_1$;
(2) $P_{11} = a_1 \times b_1$; $b_{sum} = b_0 + b_1$;
(3) $P_{ss} = a_{sum} \times b_{sum}$, $C = (P_{11}, P_{00}) - P_{00} \times 2^{128}$;
(4) $C = C - P_{11} \times 2^{128}$;
(5) $C = C + P_{ss} \times 2^{128}$;
(6) return C

Algorithm 4: Karatsuba–Ofman multiplication algorithm.

and each of the columns has 10, 5, 2, 1, and 1 rows, respectively, which compute the result $Z_1[261: 256]$.

The second stage reduction operation is designed to compute the result of Z_2, and it has 4 operands at most and needs 4 257-bit CSAs. Two 257-bit CSAs compute Z_2 = s15 + s16 − s17, one of which computes (ss2_1, sc2_1) = s15 + s16 + ∼s17, while the other computes Z_2 = ss2_1 + sc2_1 + 1. Besides, the remaining two 257-bit CSAs compute Z_3 = ss2_1 + sc2_1 − p, one of which computes (ss2_2, sc2_2) = ss2_1 + (sc2_2, 1'b1) + ∼p, while the other computes Z_3 = ss2_2 + sc2_2 + 1'b1.

3.3. Integer Multiplication. Most of the traditional high-performance architectures are based on multipliers. Due to the disadvantages of full-word multipliers, long multiplication should be split into small bits and more operation cycles. Even though the one-cycle 256-bit multiplier in [20] possesses the best speed, it also consumes the most hardware area and the worst latency. To balance hardware consumption and performance, the KO multiplication algorithm based on divide-and-conquer is adopted in this paper, as shown in Algorithm 4:

$$\begin{aligned} A * B &= \left(a_1 2^{128} + a_0\right) * \left(b_1 2^{128} + b_0\right) \\ &= a_1 b_1 2^{256} + \left((a_0 + a_1)(b_0 + b_1) - a_1 b_1 - a_0 b_0\right) 2^{128} + a_0 b_0, \end{aligned} \tag{5}$$

where $A, B \in GF(p)$, $a_0, a_1, b_0, b_1 \in [0, 2^{128} - 1]$. Compared with the cascading 128-bit × 128-bit unsigned multipliers in [16] which use four amounts of half-word multiplication, the KO algorithm just uses three at the cost of one extra full-word

	ss14[260]	ss14[259]	ss14[258]	ss14[257]	ss14[256]	ss14[255]	…	ss14[0]
sc14[260]	sc14[259]	sc14[258]	sc14[257]	sc14[256]	sc14[255]	sc14[254]	…	1
	1	1	1	1	1	$\bar{s}$12[255]	…	$\bar{s}$12[0]
	ss15[260]	ss15[259]	ss15[258]	ss15[257]	ss15[256]	ss15[255]	…	ss15[0]
sc15[260]	sc15[259]	sc15[258]	sc15[257]	sc15[256]	sc15[255]	sc15[254]	…	1
	1	1	1	1	1	$\bar{s}$13[255]	…	$\bar{s}$13[0]
	ss16[260]	ss16[259]	ss16[258]	ss16[257]	ss16[256]	ss16[255]	…	ss16[0]
sc16[260]	sc16[259]	sc16[258]	sc16[257]	sc16[256]	sc16[255]	sc16[254]	…	1
	1	1	1	1	1	$\bar{s}$14[255]	…	$\bar{s}$14[0]
	ss17[260]	ss17[259]	ss17[258]	ss17[257]	ss17[256]	ss17[255]	…	ss17[0]
sc17[260]	sc17[259]	sc17[258]	sc17[257]	sc17[256]	sc17[255]	sc17[254]	…	1
	1	1	1	1	1	$\bar{s}$15[255]	…	$\bar{s}$15[0]
	ss18[260]	ss18[259]	ss18[258]	ss18[257]	ss18[256]	ss18[255]	…	ss18[0]
sc18[260]	sc18[259]	sc18[258]	sc18[257]	sc18[256]	sc18[255]	sc18[254]	…	1
	1	1	1	1	1	$\bar{d}$[255]	…	$\bar{d}$[0]
	ss19[260]	ss19[259]	ss19[258]	ss19[257]	ss19[256]	ss19[255]	…	ss19[0]
sc19[260]	sc19[259]	sc19[258]	sc19[257]	sc19[256]	sc19[255]	sc19[254]	…	0
	0	0	0	0	0	p[255]	…	p[0]
	ss20[260]	ss20[259]	ss20[258]	ss20[257]	ss20[256]	ss20[255]	…	ss20[0]
sc20[260]	sc20[259]	sc20[258]	sc20[257]	sc20[256]	sc20[255]	sc20[254]	…	0
sc21[260]	sc21[259]	sc21[258]	sc21[257]	sc21[256]	sc21[255]	sc21[254]	…	0
Z1[261]	Z1[260]	Z1[259]	Z1[258]	Z1[257]	Z1[256]	Z1[255]	…	Z1[0]

Figure 2: The carry-save addition in the first reduction.

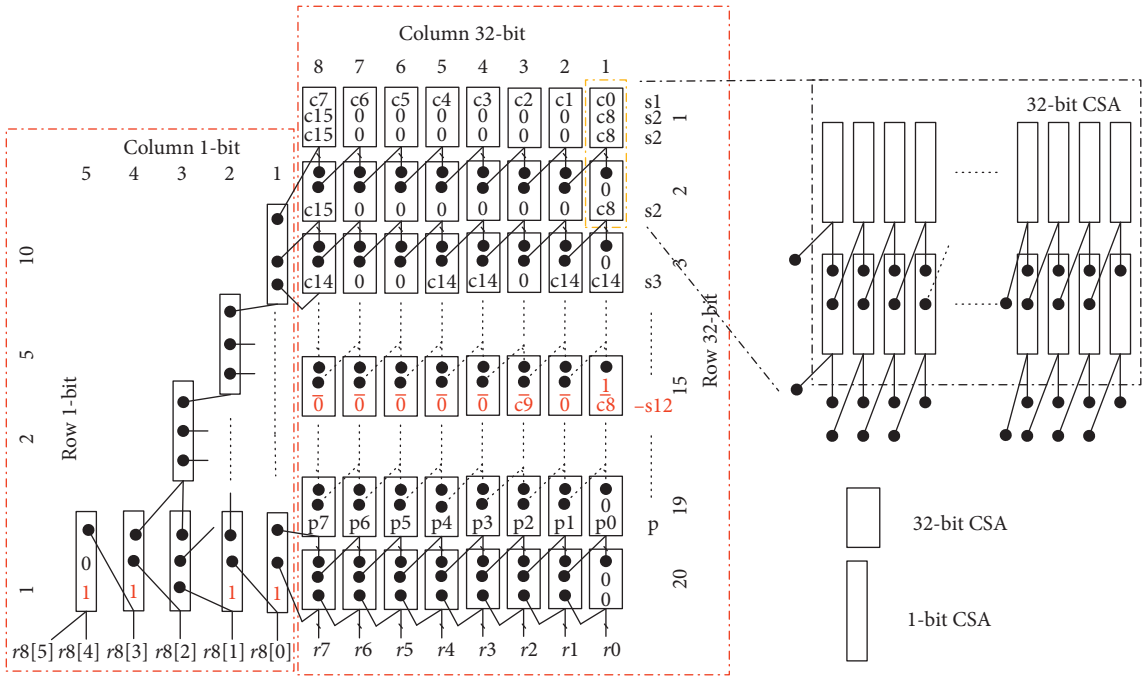

Figure 3: The first stage reduction with the carry-save addition.

subtraction and two extra half-word additions. While the KO algorithm presented in [11] requires six cycles, the KO algorithm presented in [27] requires only five cycles, shown in Algorithm 4.

3.4. Point Addition and Point Doubling. A series of ECADD and ECDBL operations make up PM. For no-idle cycles, a good ECADD and ECDBL algorithm proposed in [27] is chosen for this architecture, given as Algorithm 5 below. The algorithm has three advantages. To be specific, firstly, the multiplication and MR are performing in parallel except for one case. It is noted that the second multiplication of the point addition must wait until the first modular multiplication finishes, because the one input of the second multiplication, multiplier T1, is the output of the first modular multiplication. Secondly,

Input: $P1 = (X1, Y1, Z1)$, $P2 = (x2, y2)$
Output: $P3 = P1 + P2$
(1) $T1 = Z1Z1$
(2) $T2 = T1Z1$
(3) $T1 = T1x2$
(4) $T2 = T2y2$, $T1 = T1 - X1$
(5) $T3 = T1T1$, $T2 = T2 - Y1$
(6) $Z3 = Z1T1$
(7) $T4 = T3T1$
(8) $T3 = T3X1$
(9) $T5 = T2T2 - T4$, $T1 = 3T3$
(10) $T4 = T4Y1$, $T1 = T1 - T5$
(11) $Y3 = T1T2 - T4$, $X3 = T3 - T1$
(12) return $(X3, Y3, Z3)$
Input: $P1 = (X1, Y1, Z1)$
Output: $P3 = 2P1$
(1) $T1 = Z1Z1$, $Y3 = 2Y1$
(2) $T4 = Y3Y3$, $T2 = X1 - T1$, $T1 = X1 + T1$
(3) $T2 = T2T1$
(4) $T3 = T4X1$, $T2 = 3T2$
(5) $Z3 = Y3Z1$, $T1 = 2T3$
(6) $X3 = T2T2 - T1$
(7) $T3 = T4T4$, $T1 = T3 - X3$
(8) $T3 = T3/2$, $Y3 = T1T2 - T3$
(9) return $(X3, Y3, Z3)$

Algorithm 5: Point addition and point doubling algorithm.

multiplication operation is constantly running, no matter whether shifting from ECDBL to ECDBL or switching between ECADD and ECDBL. Thirdly, hardware consumption is minimized by using only one modular multiplication unit and two modular addition/subtraction units. The proposed high-performance architecture is displayed in Figure 4. The MA/MS unit is designed to perform multiple functions, such as T1 − X1, $X1 + T1$, 3T1, and Y3/2.

4. Implementation and Validation

The architecture described above is implemented with the Verilog-HDL language. It is synthesized using Design Compilers with the SMIC 130 nm CMOS standard cell library and is evaluated based on the 2-way NAND gate. Apart from that, for comparison with other designs on FPGA platform, it is also implemented on Xilinx Virtex-6 xc6vlx760, using Xilinx ISE 14.7. The performance is obtained by ModelSim simulation. The testing data meet the ECC cryptography protocol and are randomly generated. For a hardware design, the performance and hardware consumption are two main evaluation metrics. Besides, the time-area product is a metric to validate the trade-off between performance and hardware consumption.

With the window NAF recoding method, the time executing point multiplication is denoted as

$$\frac{m}{w+1}A + mD, \tag{6}$$

where $m = \log_2 p$; w refers to the width of NAF; A is the cycle that ECADD required, while D is ECDBL's cycle

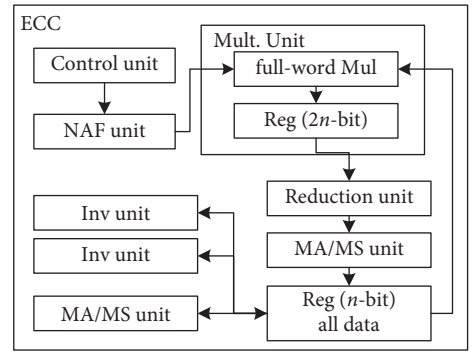

Figure 4: ECC architecture.

consumption. In this work, w is set to 4. The calculations of 1P, 3P, 5P, and 7P are precalculated.

Table 1 shows the clocks that are required by each operation. In the fixed point, MM operation uses NAF4 recoding of scalar k and takes an average of 14242 cycles by testing 1000 times. After PM operation, two MI operations are required for coordinates conversion from Jacobian coordinates to affine coordinates.

Table 2 shows the comparison among other designs over 256-field-order GF (p). The architecture in [9] is using 256-bit multipliers. In this case, its area is large and there are 659 K gates. As it consumes many large hardware resources, it is not suitable for teleoperated robots. The architecture in [18] relies on two multiplier units using interleaved modular multiplication algorithms. Hence, it is featured with a smaller area but worse computation efficiency. The proposed design is 32.7 times faster in [18]. The architecture in [22] adopts a systolic arithmetic unit and obtains smaller areas but takes more clock cycles. The AT (area-time products) of our architecture are smaller than those of [18, 22].

The design in [28] adopts projective coordinates to avoid MI and employs a radix-2 modular multiplication algorithm for MM. In [29], Shah et al. presented a high-speed processor on the basis of redundant signed digit (RSD) arithmetic to prevent lengthy carry propagation delay. It is able to run at a high frequency of 327 MHz and requires 0.47 ms to perform a single PM operation. The architecture in [11] uses half-word multipliers based on the Barrett modular multiplication algorithm. In [19], a unified architecture of computing MA, MS, and MM is proposed. The designs in [30, 31] only apply adder results in a worse performance than ours. The radix-4 booth encoding interleaved modular multiplication algorithm is adopted in [30, 31]. Besides, the NAF point multiplication algorithm is applied in [31], while the double-and-always-add point multiplication algorithm is employed in [30]. As NAF2 has the merits of decreasing PM complexity from $(m/2 * A + m D)$ to $(m/3 * A + m D)$, the design in [30] takes more LUTs to get the comparable clock cycle consumption in the same platform compared with the design in [31]. The architecture proposed here needs fewer clock cycles and is faster when concerning performing point multiplication than those architectures in [11, 18, 19, 21, 30, 31].

The security concern is one of the most important issues in teleoperated robotics systems. In a harsh condition, time

Table 1: Cycle number of EC operations.

Operation	Cycle	Operation	Cycle
MA	1	ECADD	40
MS	1	ECDBL	56
MM	5	PM (fixed point)	14242
MI	544	PM (unfixed point)	18375

Table 2: Performance comparison.

Design	Platform	Field order	Area	Frequency (MHz)	Cycles (k)	PM (ms)	At
Ours	0.13um	SCA-256	341.98 k gate	153.8	14.24	0.092	31.66
[9]	0.13um	SCA-256	659 k gate	163.7	3.3	0.02	13.18
[18]	0.13um	256	167.5 k gate	110	331.1	3.01	504.18
[22]	0.13um	256	122 k gate	556	562	1.01	123.22
Ours	Virtex-6	SCA-256	10.06 k Slice, 80 DSP	38.388	14.24	0.37	3.72
[28]	Virtex-7	256	32.781 k LUTs	177.7	262.7	1.48	13.17
[29]	Virtex-6	256	65.6 k LUTs	327	153.16	0.47	30.83
[11]	Virtex-II	256	15.8 k Slice, 64 DSP	35.6	35	0.98	15.48
[19]	Virtex-4	256	13.16 k Slice	40	200	5.00	65.80
[30]	Virtex-4	256	35.7 k Slice	70	207.10	2.96	105.67
[31]	Virtex-4	256	20.579 k Slice	49	191.82	3.91	80.46

delay and power consumption are important, so using hardware to realize cryptographic algorithms has become an imperative tendency. The ECC processor we proposed here is implemented in hardware and can provide a high performance. The most complicated operations, such as PM, PA, and modular operations, are implemented by the hardware proposed here and this hardware module can be called by software to realize digital signature/verification and encryption/decryption to resolve the safety issue of teleoperated systems.

5. Conclusion

In a teleoperated system, robots interact with and are controlled by human operators through a communication network. Therefore, security becomes an import issue and ECC is the well choice among different cryptographic algorithms due to its lower key length. In this work, a high-performance ECC architecture of SM2 is proposed, which is suitable for the teleoperated robot's security. To reduce latency owing iterated subtractions, a TSMR algorithm on SCA-256 is presented. Thus, the intermediate result $Z \in [0, 2p)$ is improved when compared with $Z \in [0, 14p)$ of traditional algorithms. To avoid iterated subtractions, a TSMR algorithm in SCA-256 is shown and implemented with a carry-save adder architecture with the subtraction. To the area/performance trade-off, the half-word multiplier is adopted, equipped with pipeline design fully enhancing the calculation parallelism. The experimental results show that the proposed design takes 0.092 ms to perform 256-bit PM with 153.8 MHz frequency and consumes 341.98 k gate areas. Furthermore, the implementation result indicates that the proposed architecture has better performance and smaller AT than previous works.

In the future, the optimization of modular multiplication will be studied to further reduce the hardware overhead. The portability of the hardware modules and the software-hardware codesign will be further studied to extend the application fields. Antiattack technology is another interesting piece of work worth studying.

Data Availability

The raw/processed data required to reproduce these findings cannot be shared at this time as the data also form part of an ongoing study.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the Independent High Safe Security Chip Research for the Metering and Electricity Program under Grant no. ZBKJXM20180014/SEPRI-K185011, and this support is acknowledged by the authors.

References

[1] T. Bonaci and H. J. Chizeck, "Surgical telerobotics meets information security," in *Proceedings of the Robotics, Science and Systems (RSS) Workshop on Algorithmic Frontiers in Medical Robotics*, Freiburg, Germany, July 2012.

[2] J. Guo, C. Liu, and P. Poignet, "A scaled bilateral teleoperation system for robotic-assisted surgery with time delay," *Journal of Intelligent & Robotic Systems*, vol. 95, no. 1, pp. 165–192, 2019.

[3] H. Su, Y. Schmirander, Z. Li et al., "Bilateral teleoperation control of a redundant manipulator with an RCM kinematic constraint," in *Proceeding os the IEEE International Conference on Robotics and Automation (ICRA)*, Montreal, Canada, May 2020.

[4] T. Bonaci, J. Yan, J. Herron et al., "Experimental analysis of denial-of-service attacks on teleoperated robotic systems," in *Proceedings of the 6th ACM/IEEE International Conference on Cyber-Physical Systems*, Seattle, WA, USA, April 2015.

[5] T. Bonaci, J. Herron, T. Yusuf, T. Kohno, and H. J. Chizeck, "To make a robot secure: an experimental analysis of cyber security threats against teleoperated surgical robots," 2015, http://arxiv.org/abs/1504.04339.

[6] T. Bonaci, A. Alva, J. Herron, R. Calo, and H. J. Chizeck, "I did it my way: on law and operator signatures for teleoperated robots," in *Proceedings of the Annual Conference on Robotics, Law and Policy*, Seattle, WA, USA, April 2015.

[7] V. S. Miller, "Use of elliptic curves in cryptography," *Proceeding of CRYPTO*, vol. 1986, pp. 417–426, 1985.

[8] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of Computation*, vol. 48, no. 177, p. 203, 1987.

[9] Z. Zhao and G. Bai, "Ultra-high-speed SM2 ASIC implementation," in *Proceeding of the IEEE 13th International Conference*, pp. 182–188, Washington, DC, USA, October 2014.

[10] State Cryptography Administration of China, *ISO/IEC 14888-3: SM2 Digital Signature Mechanism*, State Cryptography Administration of China, 2017.

[11] X. Feng and S. Li, "A high performance FPGA implementation of 256-bit elliptic curve cryptography processor over GF (p)," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E98.A, no. 3, pp. 863–869, 2015.

[12] A. Satoh and K. Takano, "A scalable dual-field elliptic curve cryptographic processor," *IEEE Transactions on Computers*, vol. 52, no. 4, pp. 449–460, 2003.

[13] H. Marzouqi and M. Al-Qutayri, "Review of elliptic curve cryptography processor designs," *Microprocessors and Microsystems*, vol. 39, no. 2, pp. 97–112, 2015.

[14] I. H. Salah, F. Zhou, F. Gebali, and T. F. Al-Somani, "Review of elliptic curve processor architectures," in *Proceeding of the IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*, pp. 192–200, Victoria, BC, USA, August 2015.

[15] H. Eberle, S. Shantz, V. Gupta, N. Gura, L. Rarick, and L. Spracklen, "Accelerating next-generation public-key cryptosystems on general-purpose CPUs," *IEEE Micro*, vol. 25, no. 2, pp. 52–59, 2005.

[16] C. J. McIvor, M. McLoone, and J. V. McCanny, "Hardware elliptic curve cryptographic processor over," *IEEE Transactions on Circuits and Systems II*, vol. 53, no. 9, pp. 1946–1957, 2006.

[17] A. K. Zia-Uddin and B. Mohammed, "Throughput/area-efficient ECC processor using Montgomery point multiplication on FPGA," *IEEE Transactions on Circuits and Systems II*, vol. 62, no. 11, pp. 1078–1082, 2015.

[18] S. Ghosh and M. Alam, "DR Chowdhury, IS Gupta, "Parallel crypto-devices for GF (p) elliptic curve multiplication resistant against side channel attacks," *Computers & Electrical Engineering*, vol. 35, no. 2, pp. 329–338, 2009.

[19] K. Javeed and X. Wang, "FPGA based high speed SPA resistant elliptic curve scalar multiplier architecture," *International Journal of Reconfigurable Computing*, vol. 5, pp. 1–10, 2016.

[20] T. Güneysu and C. Paar, *Ultra High Performance ECC over NIST Primes on Commercial FPGAs*, Springer Berlin Heidelberg, Berlin, Germany, 2008.

[21] S. Ghosh, M. Alam, I. S. Gupta, and D. R. Chowdhury, "A Robust GF (p) parallel arithmetic unit for public key cryptography," *EUROMICRO DSD*, pp. 109–115, 2007.

[22] G. Chen, G. Bai, and H. 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*, vol. 54, no. 5, pp. 412–416, 2007.

[23] K. Javeed, X. Wang, and M. Scott, *Serial and Parallel Interleaved Modular Multipliers on FPGA Platform*, FPL, Juno Beach, FL, USA, 2015.

[24] J. A. Solinas, "Efficient arithmetic on koblitz curves," *Designs Codes & Cryptography*, vol. 19, no. 2, pp. 195–249, 2000.

[25] 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 & Systems I: Regular Papers*, vol. 58, no. 8, pp. 1798–1812, 2011.

[26] D. Hankerson, A. Menezes, and S. Vanstone, *Guide to Elliptic Curve Cryptography*, Springer, Berlin, Germany, 2004.

[27] X. Hu, X. Zheng, S. Zhang et al., "A high-performance elliptic curve cryptographic processor of SM2 over GF (p)," *Electronics*, vol. 8, no. 4, p. 431, 2019.

[28] M. M. Islam, M. 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.

[29] Y. A. Shah, K. Javeed, S. Azmat et al., "Redundant-signed-digit-based high speed elliptic curve cryptographic processor," *Journal of Circuits, Systems, and Computers*, vol. 28, no. 5, pp. 120–126, 2019.

[30] K. Javeed and X. Wang, "Low latency flexible FPGA implementation of point multiplication on elliptic curves over GF (p)," *International Journal of Circuit Theory & Applications*, vol. 45, no. 2, pp. 214–228, 2017.

[31] K. Javeed, X. Wang, and M. Scott, "High performance hardware support for elliptic curve cryptography over general prime field," *Microprocess & Microsyst*, vol. 51, pp. 331–342, 2016.