Received 20 December 2022, accepted 9 January 2023, date of publication 12 January 2023, date of current version 18 January 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3236406

RESEARCH ARTICLE

Multi-Functional Resource-Constrained Elliptic Curve Cryptographic Processor

BINH KIEU-DO-NGUYEN[1,2], (Graduate Student Member, IEEE), CUONG PHAM-QUOC[2], (Member, IEEE), NGOC-THINH TRAN[2], (Member, IEEE), CONG-KHA PHAM[1], (Member, IEEE), AND TRONG-THUC HOANG[1], (Member, IEEE)

[1]Department of Computer and Network Engineering, The University of Electro-Communications (UEC), Tokyo 182-8585, Japan
[2]Department of Computer Engineering, Ho Chi Minh City University of Technology (HCMUT), VNU-HCM, Ho Chi Minh City 740050, Vietnam

Corresponding author: Cuong Pham-Quoc (cuongpham@hcmut.edu.vn)

This work was supported by the New Energy and Industrial Technology Development Organization (NEDO) under Project JPNP16007.

ABSTRACT With the rising data evolution, the demand for secured communications over networks is rising immensely. Elliptic Curve Cryptography (ECC) provides an attractive solution to fulfill the requirements of modern network applications. Many proposals published over the year over different variants of ECC satisfied some of the issues. Nevertheless, modern network applications such as Internet-of-Thing (IoT) and Software-Defined Networking (SDN) put the requirements on various aspects and can only be solved by different ECC algorithms. Looking at this point of view, an efficient architecture that could combine multiple ECC algorithms becomes an urgent request. In addition, even though many investigations of ECC on Field-Programmable Gate Arrays (FPGA), an efficient architecture that could be well-deployed on Application-Specific Integrated Circuit (ASIC) needs to get more study. Therefore, this paper proposes an area-efficient ECC hardware design that could integrate multiple ECC algorithms. The proposed design is deployed on both ASIC and FPGA platforms. Four well-known ECC-based Digital Signature Algorithms (DSAs), which are the Edwards-curve Digital Signature Algorithm (EdDSA) with Curve25519, Elliptic Curve Digital Signature Algorithm (ECDSA) with National Institute of Standards and Technology (NIST) Curve P-256, P-384, and P-521, are implemented. Furthermore, the design supports all DSA schemes: public-key generation, signature generation, and verification. We also provide optimized calculation flows for modular multiplication, modular inversion, point addition, point doubling, and Elliptic Curve Point Multiplication (ECPM) for two different elliptic curves: the NIST curve and Edward curve, on a unique architecture. The calculation processes are designed in projective coordinates and optimized in time and space to achieve a high level of parallelism. The proposed ECC processor could run up to 102-MHz on ASIC 180-nm and 109.7-MHz on Xilinx Virtex-7. In terms of area, The processor occupies 377,471 gates with 4.87-mm^2 on the ASIC platform and 11,401 slices on the FPGA platform. The experimental results show that our combinational design achieves area-efficient even when compared with other single-functional architecture on both ASIC and FPGA.

INDEX TERMS ASIC, cryptography processor, elliptic curve, FPGA.

I. INTRODUCTION

Cryptography's role is increasingly important due to the growing of attended devices in traffic networks. Personal users, governments, and companies rely on security devices to protect their communications over the Internet. Elliptic Curve Cryptography-based (ECC-based) Digital Signature Algorithm (DSA) has gained more consideration than conventional DSAs due to its high performance without reducing the security level. Among the current ECC-based DSAs, Elliptic Curve Digital Signature Algorithm (ECDSA) and Edwards-curve Digital Signature Algorithm (EdDSA) are widely used as digital signature methods to guarantee the validity of communications. The current

The associate editor coordinating the review of this manuscript and approving it for publication was S. K. Hafizul Islam.

implementation of ECDSA, despite the proven vulnerability against Side-Channel Attacks (SCAs) [1], [2], [3], is still improving in the recent proposals [4], [5], [6]. Moreover, it is used in many applications such as Short Message Service (SMS) security [7], authorization [8], and Wireless Sensor Network (WSN) [9]. EdDSA is a DSA base on the Edwards curve. It targets backdoor issues of ECDSA while promoting a higher performance. Hence, it is currently used in many applications such as Internet of Things (IoT) devices [10], Blockchain [11], and Domain Name System (DNS) [12]. The attacks on EdDSA are currently investigating in these years [13], [14].

There are proofs that quantum computers will break the ECC-based DSA [15], but the Post-Quantum Cryptography (PQC) algorithms still require both classics and PQC [16]. Therefore, investigating ECC-based digital signatures is still needed. ECC-based DSAs nowadays play an important role in many applications and protocols, such as Transport Layer Security (TLS) [17], IPsec [18], and OpenSSH protocols [19]. They are also found in environments such as web servers, personal computers, embedded devices, or IoT devices, which constrain different aspects. Some applications, such as Software-Defined Networking (SDN), require different DSAs in a single system. However, the efficient architecture that could be integrated with multiple ECC-based DSAs has yet to be published. In addition, almost existing investigations in this field only target Field-Programmable Gate Arrays (FPGAs). An effective Application-Specific Integrated Circuit (ASIC) design needs to receive sufficient studies. Therefore, an investigation of the hardware implementation of multi-functional ECC-based DSA processors on both ASIC and FPGA is required, considering the exploitation of parallel architecture to promote the overall system's performance.

In 2017, Panjwani [20] presented one of the first scalable ECC architectures in prime fields over the National Institute of Standards and Technology (NIST) Curve P-521. The authors designed a system with a participant of both hard Intellectual Property (IP) processors and hardware accelerators. An optimized flow for hardware-software co-design is provided. Such an approach could achieve higher performance because the hard-IP processor can work better with the operations that could be performed in parallel. The data exchange could be leaked during movement. Furthermore, a software-hardware co-design can only be deployed on the FPGA platform with a System on Chip (SoC). In 2008, Güneysu and Paar [21] described a high-performance ECC implementation over two NIST primes. Their design takes advantage of Digital Signal Processing (DSP) blocks available in FPGAs, which could solve the multiplication in a few cycles. This paper reports a latency of 749-μs for a multiple point multiplication over the NIST P-256 Curve. The resource consumption is 1715 slices and 32 DSPs On the Xilinx Virtex-4 XC4VFX12-12 Xilinx FPGA board. Guneysu et al. next provides the first Curve25519 implementation using a DSP-based architecture [22]. This work is supplemented by adding side-channel countermeasures in [23] and [24] to protect against some physical attacks. The same issue is reported in the work of Vliegen et al. [25]. They provide a resource-saved core over the NIST P-256 Curve that could resist Simple Power Attacks (SPA). The work of Koppermann et al. [26], [27] presented a high-speed prime field multiplier with a latency of 92-μs for a point multiplication. In addition, in [28], a low-latency Elliptic Curve Point Multiplication (ECPM) was proposed employing a pipelined arithmetic architecture on FPGA and ASIC platforms. The authors in [29] provide a Crypto-Processor for Real-Time IoT Applications with both FPGA and ASIC deployment, including three versions that support NIST P-256, P-521, and P-256/521. The provided crypto-processor accelerates the point operations, including point addition, point doubling, inversion, and double point multiplication. The supported operations could be used for many protocols, such as ECDSA, Elliptic Curve Integrated Encryption Scheme (ECIES), Elliptic Curve Diffie Hellman (ECDH), and Elliptic Curve Menezes Qu Vanstone (ECMQV).

Another approach to increase the ECPM is using Residue Number Systems (RNS), also called the carry-free number system. In 2006, Schinianakis et al. [30] explored the use of RNS for building finite-field arithmetic operations. Their design occupies 36,000 Lookup Tables (LUT) on Xilinx Virtex XCV1000E and is running at 39.7-MHz, which requires 3950-μs for a point multiplication. Guillermin provides another implementation on RNS in [31]. His architecture consumes 9117 Adaptive Logic Modules (ALMs) on the P2S30F484C3 Altera FPGA. Their results show that their core could run up to 157-MHz. Their implementation takes 680-μs for a single point multiplication. In 2019, Asif et al. in [32] provided a novel method to apply the RNS to implement ECPM. Even though their design provides high throughput, the core consumes up to 24,200 slices and 276 blocks of Random Access Memory (RAM). These designs cannot be deployed on resource-constrained applications like WSN and SDN.

To the best of our knowledge, almost hardware implementations on ECC only focus on the EPCM. In comparison, the proposals that provide an ECC-based DSA cryptosystem must be optimized for ASIC deployment. Moreover, there is no research on architecture with multiple ECC DSAs in a single processor optimized for both ASIC and FPGA. In this work, we provide three main contributions listed below.

- **Multi-functional finite field arithmetic unit**. The arithmetic units in the previous design only target a single function, such as modular addition, subtraction, and multiplication. This way, data must be moved to specific units during the calculation process. Furthermore, connections among units to all others are required. It leads to an increase in resource consumption and delay. This situation worsens when data width is high in some curves. Our multi-functional finite field arithmetic unit design allows all unit could switch their function depending on the current requirement. Based on this, we suggest an effective switching system that could be used for different calculation flow of different curve.

- **Multi-functional ECC-based DSA design**. We propose a compelling architecture that integrates different ECC-based DSAs. The design takes advantage of parallel arithmetic units. Furthermore, on the unique hardware architecture, the suggested flows can perform operations from two different curves, which are NIST P-256/384/521 and Curve25519. These flows are optimized to reduce the delay and the number of data exchanges among the arithmetic units.
- **ASIC and FPGA implementation**. We implement the design using Complementary Metal-Oxide Semiconductor (CMOS) 180-nm technology and FPGA. The design is optimized to achieve the same performance on these platforms. The results provide not only an estimation of the resource consumption of a multi-functional ECC-based hardware accelerator but also the implementation results of a high bit-width Curve on ASIC, which are limited in the previous publications.

The remainder of this paper is organized as follows. Section II reviews the background of ECC and two well-known ECC-based DSAs, ECDSA and EdDSA. Section III conducts the multi-functional hardware architecture. Section IV shows the experimental results with comparison and discussion. Section V gives a glance at the limitations of the current design. Finally, Section VI concludes the paper and presents our future works.

II. BACKGROUND KNOWLEDGE

A. DIGITAL SIGNATURE ALGORITHM

Digital Signatures Algorithm is a method of signing a message stored in electronic form. As such, a signed message can be transmitted over a computer network. Digital signatures effectively provide authentication, integrity, and non-repudiation of messages. The DSA was adopted as a Digital Signature Standard (DSS) by NIST in 1994 [33]. The DSA algorithms include three schemes. The first is the key-generation stage, where the sender generates a public key from a random number. The generated key is then sent with the message and used to verify the signature. The second one is the signature-generation stage, where the sender generates the signature for the message based on a secret key and the hash of the message. The message is then digitally signed with this signature. The last one is the verification stage. In this stage, the receiver verifies the message's validity using the public key and the signature. The message is considered valid if the recovered signature matches the received signature. Figure 1 illustrates the three stages of DSA.

B. ELLIPTIC CURVE CRYPTOGRAPHY

The ECC was first proposed by Miller [34] and Koblitz [35] based on the difficulty level of the Elliptic Curve Discrete Logarithm Problem (ECDLP). It provides a higher level of security in comparison with other cryptosystems. In comparison with Rivest Shamir Adleman (RSA), [36], or ElGamal [37], ECC requires significantly smaller parameters while providing equivalent levels of security. That leads to higher performance in computations and reductions in processing power, storage space, and bandwidth. The mathematical operations of ECC are defined over the elliptic curve $y^2 = x^3 + a * x + b$, where $4a^3 + 27b^2 \neq 0$. Different elliptic curves are decided by the a and b [38]. All points $P(x, y)$ is a point of an elliptic curve only when it satisfies the above equations. In an ECC system, the public key is a point in the curve, and a private key is a scalar number produced by Random Number Generator (RNG). The public key is obtained by multiplying the private key with the base point of the selected curve. Let P, and Q be two points on an elliptic curve such that $k * P = Q$, where k is a scalar. Even if P and Q are known, it is infeasible to obtain k if k is sufficiently large. Hence, computing $k * P$ (a point or scalar multiplication) is the most important operation in ECC. This operation can be performed by repeated point additions and point doublings with an effective flow.

Algorithm 1 Key Generation (ECDSA).

```
Input: Prime $p$, base point $G$
Output: Public key $Q$
1: Randomize $d$ $(1 \leq d \leq n-1)$
2: $Q = d * G$ over $GF(p)$
```

Algorithm 2 Signature Sheme (ECDSA).

```
Input: Prime $p$, base point $G$, private key $d$
Output: Signature $(r, s)$ of the message $m$
1: Randomize $k$ $(1 \leq k \leq p-1)$
2: $P(x_1, y_1) = k * G$ over $GF(p)$
3: $r = x_1$
4: $e = Hash(m)$
5: $s = k^{-1} * (e + d * r)$ over $GF(p)$
```

ECDSA is a variant of DSA, which uses ECC. It was accepted in 1999 as an American National Standards Institute (ANSI) standard [39], in 2000 as IEEE [40], and NIST standards [41]. Such as DSA, ECDSA describes three security schemes to protect the integrity of electronic data. In the key-generation stage, a random number is multiplied by the base point of the selected curve to compute the public key. In the signature-generation stage, a signature is generated from the base point and the secret input key. In the verification stage, the receiver recovers the signature from the received message and the public key, then compares it with the received signature. If the two signatures are identical, the received message is valid. Algorithm 1, 2, and 3 explains ECDSA in detail.

EdDSA was developed by Bernstein et al. [42]. EdDSA has gained more consideration among the existing digital signature algorithms due to its fast operations without affecting security. EdDSA includes two instances, which are Ed25519

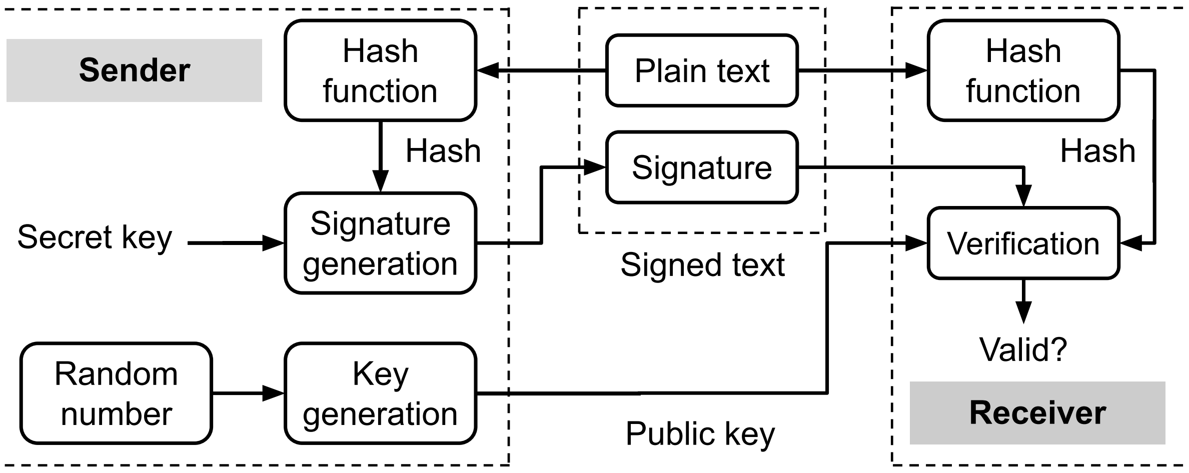


FIGURE 1. Digital signature algorithm.

Algorithm 3 Verification Scheme (ECDSA).

```
Input: Prime $p$, base point $G$, signature $(r, s)$ of the input
    message $m$, public key $Q$
Output: Valid or invalid
 1: $e = Hash(m)$
 2: $w = s^{-1}$ over $GF(p)$
 3: $v_1 = e * w$ over $GF(p)$
 4: $v_2 = r * w$ over $GF(p)$
 5: $(x_1, y_1) = v_1 * G + v_2 * Q$ over $GF(p)$
 6: if $(x_1, y_1) = O$ where $O$ is the point at infinity then
 7:     $m$ is invalid
 8: else
 9:     $v = x_1$
10: end if
11: if $v = r$ then
12:     $m$ is valid
13: else
14:     $m$ is invalid
15: end if
```

and Ed448 [43]. The Ed25519, the most popular instance of EdDSA, is widely used as a digital signature method to guarantee the validity of communications. Most websites are switching to Ed25519, which is suitable for higher-level security requirements. EdDSA is noticeable for high-speed and low-resource implementations. It will replace the other DSA in many standards, such as TLS and OpenSSH protocols. Hence, it has to be implemented in various platforms subject to the performance requirement of the target application, such as constrained IoT devices. However, EdDSA has not received sufficient study of hardware implementation, especially with the ASIC platform. Hardware implementation of this algorithm requires considering the advantages of parallelism, which leads to improvements in the overall system's efficiency. As the other DSA, Ed25519 has three stages, which are explained in Algorithm 4, 5, and 6. In the key-generation stage, a public key is generated from a secret key and the base point of Curve25519. In the signature-generation stage, the signatures pair is produced from the private key and the hash of the input message. The signature is then attached to the message and sent to the receiver. Finally, the received signature and the public key by the receiver are used to verify the message in the verification stage. Group arithmetic of a specific Elliptic curve could be performed on the different coordinates. In this project, we select the projective coordinates [44] for both NIST Curve and Curve25519. Projective coordination provides a chance to optimize for parallelism. In addition, a mapping between affine coordinates (x, y) and projective coordinates (X, Y, Z) for a point P is defined by the following equation. When $Z = 1$ for the input point P, then $P(x, y) = P(X, Y, 1)$; the transformation is performed without additional cost.

$$P(x, y)_{affine} \rightarrow P(X, Y, Z)_{projective}$$
$$x = \frac{X}{Z}, y = \frac{Y}{Z}$$

Algorithm 4 Key-Pair Generation (EdDSA).

```
Input: Prime $p$, base point $G$, private key $k$
Output: Public key $Q$
 1: $(hk, lk) = Hash(k)$ $hk$ is 32 bytes Most Significant Bit
    (MSB), $lk$ is 32 bytes Least Significant Bit (LSB).
 2: $a = hk$ as little-endian notation
 3: $Q = a * G$ over $GF(p)$
```

C. DIGITAL SIGNATURE ALGORITHM HIERARCHICAL ARCHITECTURE

Figure 2 describes four abstraction levels of a DSA. The top level is DSA, where the control flows for different schemes of a DSA, including key generations, signature generation,

Algorithm 5 Signature Scheme (EdDSA).

```
Input: Prime $p$ and $q$, base point $G$, private key $k$, public key $Q$, and message $m$
Output: Signature $(R, s)$ for the message $m$
1: $(hk, lk) = Hash(k)$ where $hk$ is 32 bytes MSB and $lk$ is 32 bytes LSB.
2: $r = Hash(lk, m)$ over $GF(q)$
3: $R = r * G$ over $GF(p)$
4: $h = Hash(R, Q, m)$ over $GF(q)$
5: $s = (r + h * a)$ over $GF(q)$
```

Algorithm 6 Verification Scheme (EdDSA).

```
Input: Prime $p$ and $q$, base point $G$, signature $(R, s)$ of the input message $m$, public key $Q$
Output: The input message $m$ is valid or invalid
1: $h = Hash(R, Q, m)$ over $GF(q)$
2: $hQ = R + h * Q$ over $GF(p)$
3: $sG = s * G$ over $GF(p)$
4: if $hQ = sG$ then
5:     $m$ is valid
6: else
7:     $m$ is invalid
8: end if
```

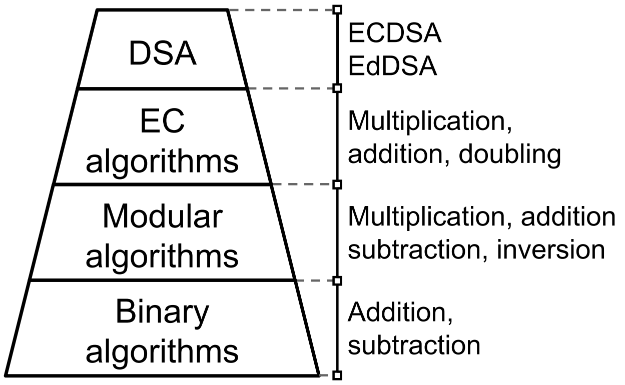


FIGURE 2. Digital signature algorithm hierarchical architecture.

and verification, are specified. The second level defined the flow for operations of an elliptic curve, including point multiplication, addition, and doubling. The point multiplication algorithm could be used for different curves, while the point addition and subtraction vary depending on the curve and select coordinate. In the third level, finite-field algorithms are implemented, including modular multiplication, addition, subtraction, and inversion. These operations are not dependent on either Elliptic Curve or DSA. Moreover, the fourth level includes implementing binary operations, such as binary addition or subtraction.

III. HARDWARE ARCHITECTURE

A. MODULAR PROCESSING ELEMENT

Elliptic Curve Point Addition (ECPA) and Elliptic Curve Point Doubling (ECPD) are used in the accumulation process during ECPM. Depending on the selected curve, modular addition, doubling, and sometimes subtraction are used. In this work, we propose an architecture that could perform modular addition and subtraction in a single architecture. This addition/subtraction unit is driven by a controller that helps to perform the point multiplication based on the additional accumulation. We called this unit a Processing Element (PE). The PE architecture is constructed from two parallel Carry Save Adder (CSA). Carry Save Adder (CSA) reduces the logical delay of three-operands addition. CSA offers a lower logic delay than conventional adders but consumes the same resource. The CSA architecture includes two stages. In the first stage, the corresponding bits of the operands are added together by parallel Full Adders (FAs). In the second stage, the generated Sum (S) and Carry bits (C) from the FAs are added to produce the results of three operands addition. Figure 4 shows the role of CSA in our modular addition/subtraction design, and Figure 3 illustrates the architecture of the CSA in detail. The CSA consumes the same resources as the two-layer ripple carry adder. However, because the bit-additions are performed in parallel in the first layer, almost logical delay is dependent on the second layer. On the contrary, the two-layer ripple carry design suffers the ripple delay from all layers.

A PE can operate modular addition, subtraction, and multiplication with an optimized architecture. Figure 4 compares the architecture of conventional modular subtraction (Figure 4(a)) and modular addition (Figure 4(b)) with our multi-functional modular addition/subtraction design (Figure 4(c)). The figure shows that, instead of implementing different modules for addition and subtraction, the design could perform both addition and subtraction by adding some supplemented basic gates. Our modular addition/subtraction architecture could be split into two branches. The left branch operates without modular value p, while the right branch operates with p at the same time. The two branches are designed to achieve a balanced architecture. When the *sub* signal is set, the circuit is logically equivalent to the modular subtraction as Figure 4(a). When the *add* signal is activated, the circuit performs the modular addition as Figure 4(b). Finally, the final multiplexer selects the appropriate results depending on the working mode. The combined architecture reduces resource usage. The authors in [29] also suggest a combined architecture of modular addition/subtraction architecture. But it requires two consecutive CSA to provide the final results. As a consequence, this design leads to a higher logical delay than ours.

The PE is built based on a modular addition/subtraction unit. The controller is implemented to allow PEs could perform addition, subtraction, or multiplication. When the processing is working, one cycle is needed, and the results can be read from the results registers. When the PE works in multiplication mode, more cycle is needed to accumulate the addition. Figure 5 shows the architecture of the PE. The A, B, and result registers store the multiplier, the multiplicand, and the product, respectively. Depending on the multiplicand's bit, the result register is also shifted and added at each cycle.

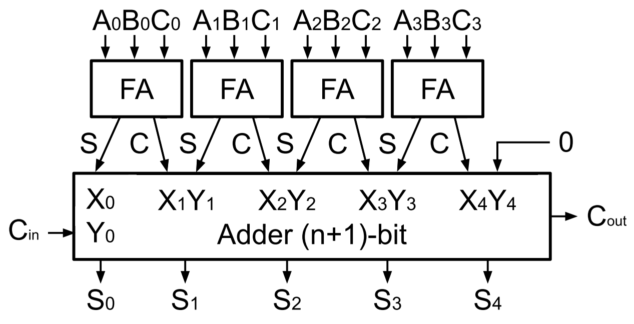


FIGURE 3. Carry-save adder.

When the multiplication is completed, the finish signals are activated. At the same time, with the rising of the finish signal, the result is written back to register B, which helps to reduce one cycle when the next operation is based on the current results.

In order to adapt to different elliptic curves, Binary Euclidean Algorithm (BEA) is used to implement the modular inversion. The modular inversion is used after the modular multiplication to reverse the point $P(X, Y, Z)$ computed in coordinate to $P(x, y)$ in affine coordinate. BEA is the traditional method that can calculate the inversion of a number over the prime field and not depend on a specific elliptic curve. Therefore, it is the most suitable for a multi-functional design. Algorithm 7 explains the BEA. Figure 6 shows the hardware implementation of BEA in our proposed architecture. The proposed BEA architecture includes four registers to store the temporal data during calculation. The new data generated by two modular addition, two right-shift circuits, and a modular subtraction are written back to the temporal registers under the control of the BEA controller. The BEA controller also determines the stop condition of the inversion process. After the calculation, the final result is assigned for y, and the system raises the finish signal. Then the system could read the inversed result from y.

Algorithm 7 Binary Euclidean Algorithm (BEA).

```
Input: Prime $p$ and $a \in [1, p-1]$
Output: $a^{-1} \bmod p$
1:  $u \leftarrow a, v \leftarrow p, x \leftarrow 1, y \leftarrow 0$
2:  while $u \neq 1$ and $v \neq 1$ do
3:      while $u \bmod 2 = 0$ do
4:          $u \leftarrow u \gg 1$
5:          if $x \bmod 2 = 0$ then
6:              $x \leftarrow x \gg 1$
7:          else
8:              $x \leftarrow (x + p) \gg 1$
9:          end if
10:     end while
11:     while $v \bmod 2 = 0$ do
12:         $v \leftarrow v \gg 1$
13:         if $y \bmod 2 = 0$ then
14:             $y \leftarrow y \gg 1$
15:         else
16:             $y \leftarrow (y + p) \gg 1$
17:         end if
18:     end while
19:     if $u \geq v$ then
20:         $u \leftarrow (u - v) \bmod p$
21:         $x \leftarrow (x - y) \bmod p$
22:     else
23:         $v \leftarrow (v - u) \bmod p$
24:         $y \leftarrow (y - x) \bmod p$
25:     end if
26: end while
27: if $u = 1$ then
28:     $y \leftarrow x$
29: end if
30: return $y$
```

B. ELLIPTIC CURVE ARITHMETIC UNIT

The Elliptic Curve Arithmetic Unit (ECAU) includes four main components: controller, switching system, memory, modular inversion, and PEs. Figure 7 illustrates the architecture of the ECAU. The controller of ECAU implements the control flows for point operations (e.g., point addition, point doubling, and point multiplication). Different control flows could be integrated depending on the selected curve, the coordinate, and the number of PEs. In our design, four PEs are used, and the control flow is optimized to perform the ECPA and ECPD. Projective coordinates are used for the NIST and Twisted-Edward curves in this work.

The projective coordinate is optimized for parallel processing by reducing the number of data-dependency among different steps of ECPA and ECPD. The switching system is organized as Table 1. All the processing elements connect with other processing elements through the switch system. They already have a local connection, as shown in Figure 5. Such connections allow the reduction in the complexity of the switching system while the components can efficiently exchange data with each other. In this design, memory has connections with all components. It is used to store the temporal data during the calculation processes; exchange data among the components that are not connected. For instance, the Inversion module (Inv.) does not connect with the processing elements. Therefore, the data exchange with the Inversion module must be performed through memory. This approach could take more cycles, but because the inversion is only required at the final step of the point multiplication process, it does not affect the overall performance.

C. POINT ADDITION AND POINT DOUBLING SCHEDULE

ECPA and ECPD are the two basic operations to calculate ECPM. The calculation flow of ECPA and ECPD is optimized to promote parallelism from four processing elements. When designing the calculation flow for ECPA and ECPD, four targets are considered: (1) Minimize the idle time of PEs,

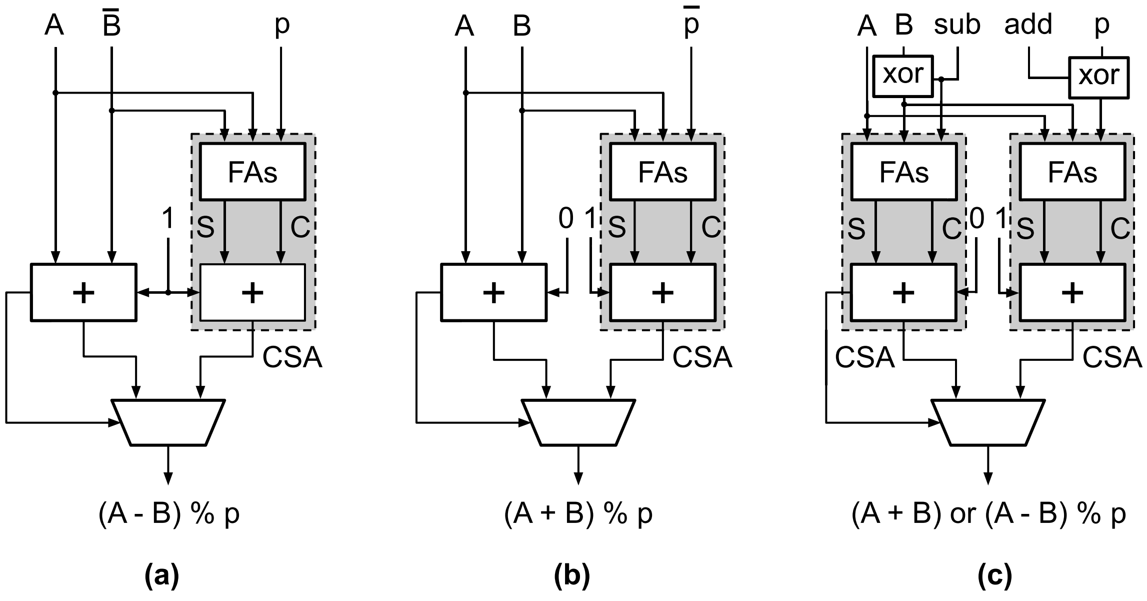


FIGURE 4. (a) Modular substraction. (b) Modular addition. (c) Modular addition/substraction.

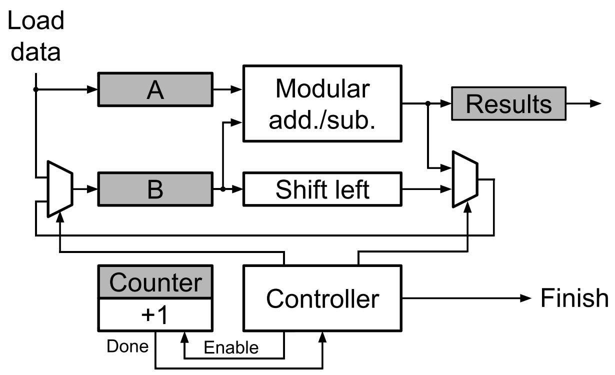


FIGURE 5. Processing element.

TABLE 1. Switching system. ◇ represents for local connection. ● represents for connection via switch system. ○ represents for not connection.

	PE_0	PE_1	PE_2	PE_3	Inv.	Mem.
PE_0	◇	●	●	●	○	●
PE_1	●	◇	●	●	○	●
PE_2	●	●	◇	●	○	●
PE_3	●	●	●	◇	○	●
Inv.	○	○	○	○	○	●
Mem.	●	●	●	●	●	○

◇ represents for local connection.
● represents for connection via switch system.
○ represents for not connection.

which means all or majority of processing elements should be working for the whole processing period; (2) Promote the parallelism to reduce the number of calculation steps;

Algorithm 8 Point Addition (NIST Curve).

```
Input: Point P($x_1, y_1, z_1$) and Q($x_2, y_2, z_2$) on Curve projective coordinate.
Output: Point R($x_2, y_2, z_2$)
1: $w_1 \leftarrow y_1 * z_2$
2: $w_2 \leftarrow x_1 * z_2$
3: $u \leftarrow y_2 * z_1 - w_1$
4: $v \leftarrow x_2 * z_1 - w_2$
5: $w_3 \leftarrow u * u$
6: $w_4 \leftarrow v * v$
7: $w_5 \leftarrow w_4 * v$
8: $w_6 \leftarrow z_1 * z_2$
9: $w_8 \leftarrow w_2 * w_4$
10: $w_9 \leftarrow w_7 - w_5 - 2 * w_8$
11: $w_{10} \leftarrow u * (w_8 - w_9)$
12: $x_3 \leftarrow v * w_9$
13: $y_3 \leftarrow w_{10} - w_1 * w_5$
14: $z_3 \leftarrow w_5 * w_6$
15: return R($x_3, y_3, z_3$)
16: Total: 14×Multplications, 7×Addition/Substraction
```

(3) Reduce the number of data movements among the processing elements through the switching system, each PE performs the next task based on its previous calculated results; (4) Reduce the number of memory access.

Algorithm 8 and 9 explain the flows for ECPA and ECPD over NIST Curve. Figures 8 and 9 review how algorithms 8 and 9 are deployed on four PEs. The flows are optimized to ensure that each processing element keeps its results for its next calculation. Therefore, the communication through the switch system and the memory access is reduced. The

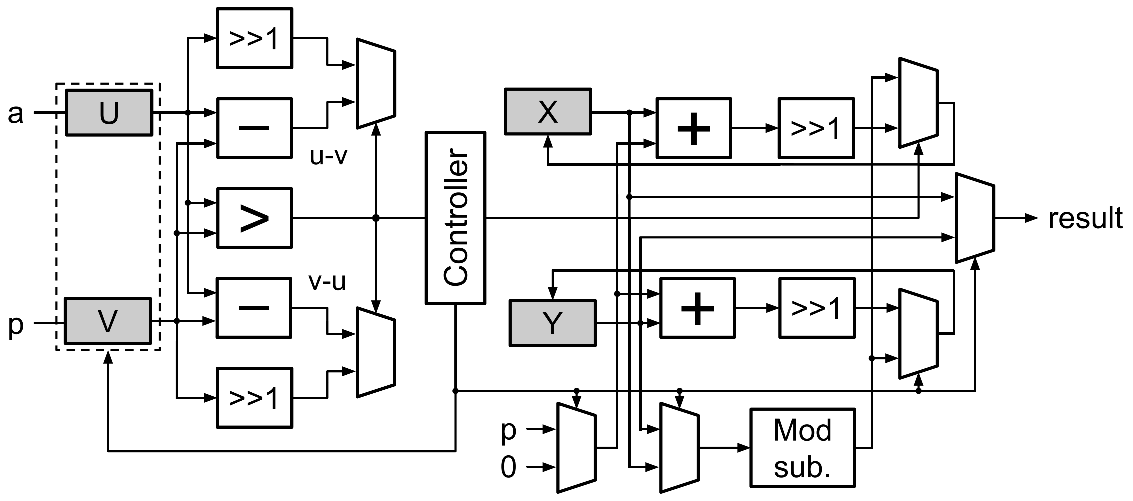


FIGURE 6. Modular inversion.

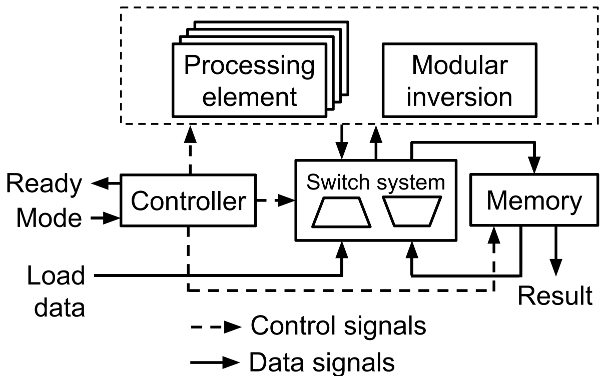


FIGURE 7. Elliptic curve cryptography processing core.

Algorithm 9 Point Doubling (NIST Curve).

```
Input: Point P($x_1, y_1, z_1$) on Curve projective coordinate.
Output: Point R($x_3, y_3, z_3$)
 1: $w_1 \leftarrow y_1 * z_1$
 2: $w_2 \leftarrow z_1 * z_1$
 3: $w_3 \leftarrow x_1 * x_1$
 4: $u \leftarrow w_3 - w_2$
 5: $w_4 \leftarrow 3 * u$
 6: $w_5 \leftarrow w_1 * y_1$
 7: $w_6 \leftarrow x_1 * w_5$
 8: $w_7 \leftarrow w_4 * w_4 - 8 * w_6$
 9: $w_8 \leftarrow w_5 * w_5$
10: $x_3 \leftarrow 2 * w_7 * w_1$
11: $y_3 \leftarrow w_4 * (4 * w_6 - w_7) - 8 * w_8$
12: $z_3 \leftarrow (2 * w_1)^3$
13: return R($x_3, y_3, z_3$)
14: Total: 11×Multplications, 14×Addition/Substraction
```

flow clearly shows that the delay for ECPA and ECPD corresponds to four consecutive multipliers. However, they are 14 and 11 multiplications for ECPA and ECPD as described in Algorithm 8 and 9, respectively.

Algorithm 10 Point Addition (Curve25519).

```
Input: Point P($x_1, y_1, z_1$) and Q($x_2, y_2, z_2$) on Curve projective coordinate.
Output: Point R($x_2, y_2, z_2$)
    $w1 \leftarrow z1 * z2$
    $w2 \leftarrow w1 * w1$
    $w3 \leftarrow x1 * x2$
    $w4 \leftarrow y1 * y2$
    $w5 \leftarrow d * w3 * w4$
    $w6 \leftarrow w2 - w5$
    $w7 \leftarrow w2 + w5$
    $u \leftarrow x1 + y1$
    $v \leftarrow x2 + y2$
    $t \leftarrow w3 + w4$

    $x_3 \leftarrow w1 * w6 * (u * v - t)$
    $y_3 \leftarrow w1 * w7 * t$
    $z_3 \leftarrow w6 * w7$
    return R($x_3, y_3, z_3$)

    Total: 12×Multplications, 6×Addition/Substraction
```

Algorithm 10 and 11 illustrate the optimized ECPA and ECPD algorithms for Curve25519. Moreover, the corresponding calculation flows are reviewed by Figure 10 and 11, respectively. Such optimization only works only when each processing element can perform all kinds of modular operations. Hence, they do not need to transfer data to a specific operation processing element. Therefore, the essential key is an all-operations design explained in Figure 4(c).

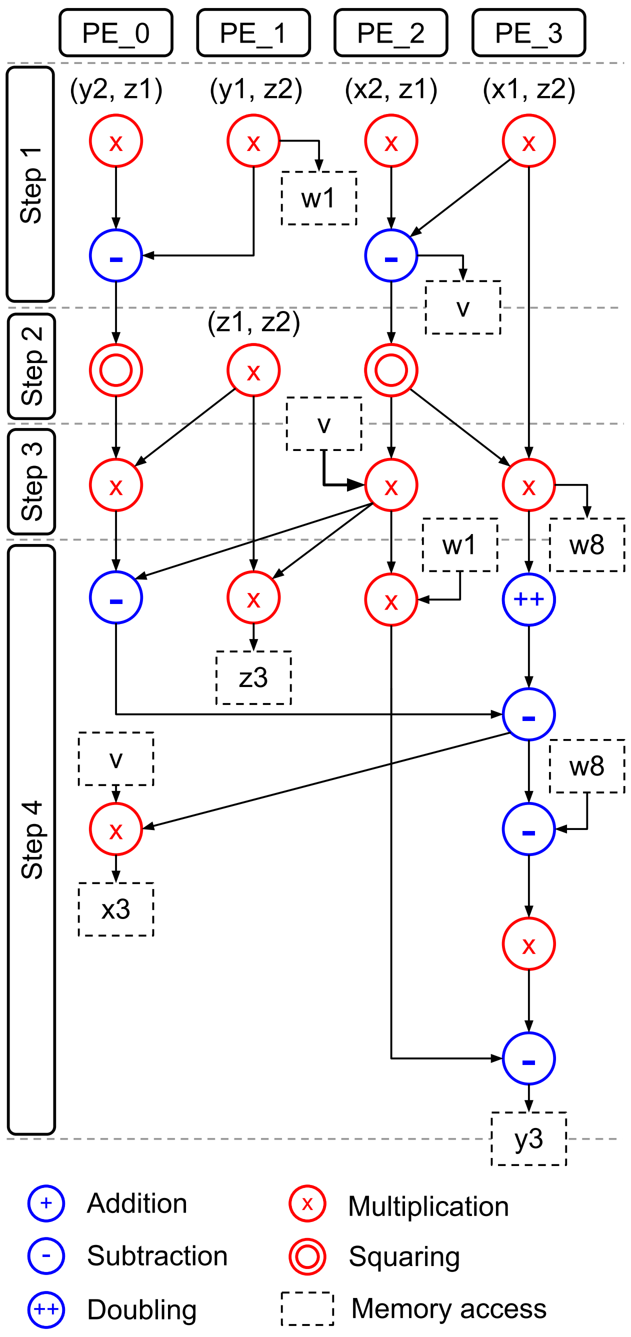


FIGURE 8. **Point addition flow over NIST Curve.**

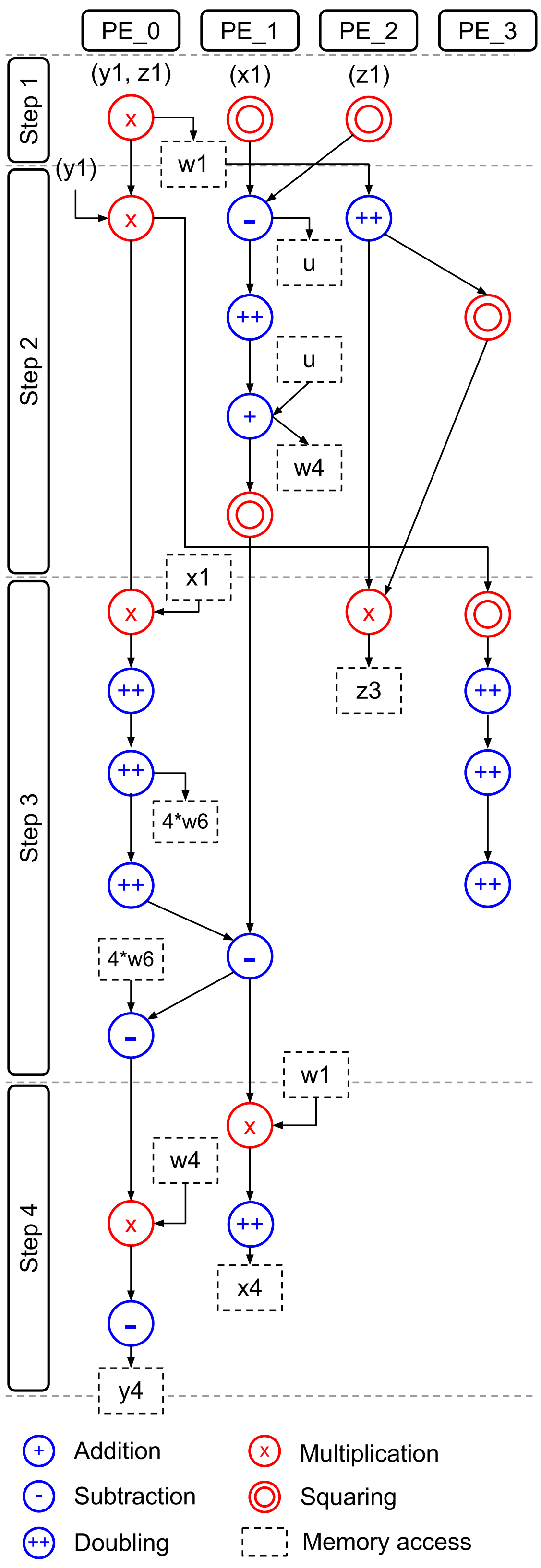


FIGURE 9. **Point doubling flow over NIST Curve.**

D. ELLIPTIC CURVE BASED DIGITAL SIGNATURE ALGORITHM SYSTEM

All the components are combined on a single chip as a multi-functional ECC-based DSA system. The top-level architecture is illustrated in Figure 12. It includes four components: the system controller with four stages, the DSA controller; the ECAU; and the storage system. The system controller determines the state of the ECC core, including the Idle state, the Initialize state, the Working state, and the Wait state. In the Idle state, the input key and the hash data are updated in the memory. In the Initialize state, the parameters are loaded from initial data memory to update the related registers of ECAU to

Algorithm 11 Point Doubling (Curve25519).

```
Input: Point P($x_1, y_1, z_1$) on Curve projective coordinate.
Output: Point R($x_3, y_3, z_3$)
    $w_1 \leftarrow (x1 + y1)^2$
    $w_2 \leftarrow x1 * y1$
    $w_3 \leftarrow y1 * y1$
    $w_4 \leftarrow -w2 + w3$
    $w_5 \leftarrow z1 * z1$
    $w_6 \leftarrow w4 - 2 * w5$

    $x_3 \leftarrow (w1 - w2 - w3) * w6$
    $y_3 \leftarrow -w4 * (w2 + w3)$
    $z_3 \leftarrow w4 * w6$
    return R($x_3, y_3, z_3$)

    Total: 7×Multplications, 7×Addition/Substraction
```

prepare for the computation process. In the Working state, the DSA controller controls the operations of ECAU to perform the finite fields operations to produce the key, the signature, or to verify a signature. The ECAU performs the finite field operations, which its controller controls. The ECAU controller is implemented to perform point multiplication, point addition, point doubling, and inversion.

IV. EXPERIMENTAL RESULT

Table 2 summarizes the resource utilization for our proposed multi-functional design. The core could perform four DSAs, which are Ed25519, and ECDSA with NIST Curve P-256, P-384, and P-521. It utilizes 11,401 slices without any DSP, and the frequency achieves 109.7-MHz. We also implement another version that can run Ed25519 and ECDSA P-256. The resource consumption is 4,929 slices, and the frequency achieves 135.96-MHz. In this implementation, the resource reduces by 2.31×, and the frequency increase by 1.24× in comparison with the four-DSAs version. In Table 2, we estimate a DSP as 100 slices as [28]. Table 2 shows that our design consumes fewer resources than the other single-functional design while it can work at a higher frequency. It should emphasize that our design is integrated with various DSAs and all DSA schemes. Our two-DSAs implementation provides better efficiency than other 256-bit single-functional designs (see Table 3). Moreover, the four-DSAs version can be compared with 384-bit and 521-bit architecture. Table 3 reveals the delay in ms and the corresponding Area Delay Production (ADP). ADP is calculated by multiplying the area (see Table 2) and the delay. The results show that our ECC Processor gets a better ADP than the other single-functional design. Figure 13 and Figure 14 illustrate the comparison of ADP among our 2-DSAs and 4-DSAs design with the other proposals when verifying and generating signatures. The results are collected from Virtex-7 by using Vivado 2020.1. The input message

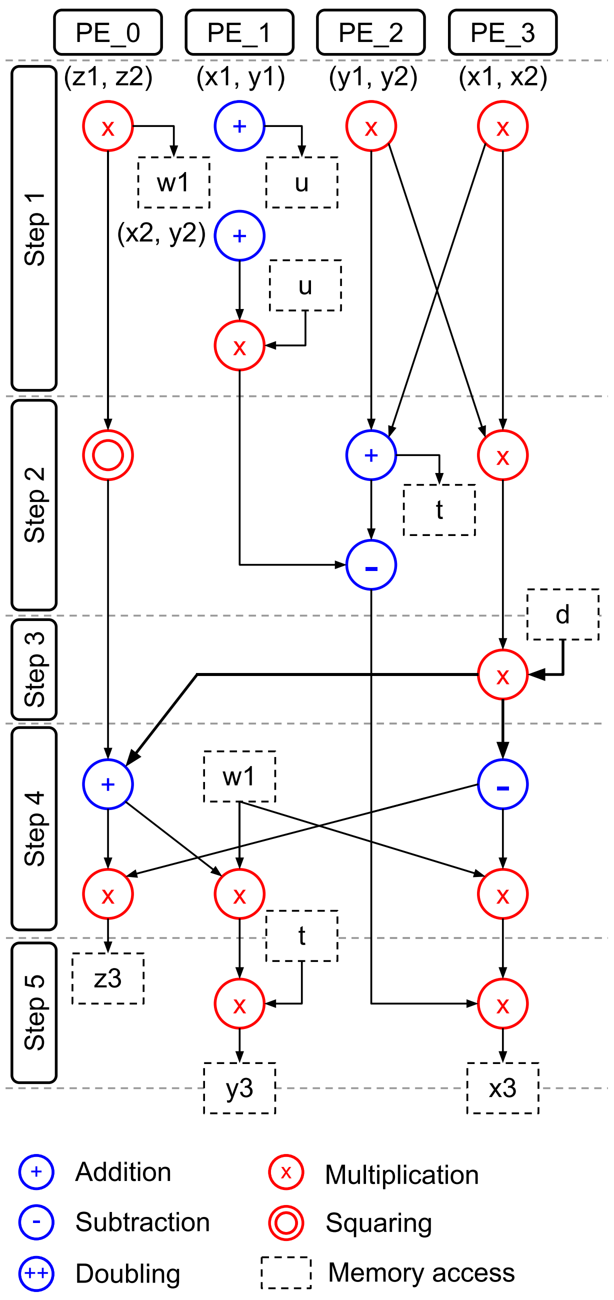


FIGURE 10. Point addition flow over Curve25519.

and parameters are taken from [18] for the ECDSA mode and from [43] for the Ed25519 mode.

The proposed ECC-based DSA architecture is verified on Xilinx VC707 Evaluation Kit. The experimental system includes a softcore processor, a Secure Hash Algorithm 2 (SHA2) core, and the ECC-based DSA core with Curve25519 and NIST Curve P-256, P-384, and P-521. The used SHA2 core in the test system could perform three SHA2 variants, which are SHA2-256, SHA2-384, and SHA2-512. The processor configures the operating mode for the SHA2 and the ECC-based DSA core. The hash data from the SHA2 are used

TABLE 2. FPGA implementation results.

			This work		[20]			[45]	[28]*
Number of supported DSAs			2-DSAs	4-DSAs	1-DSA			1-DSA	EPCM
Support DSA	**EdDSA**	**Ed25519**	●	●	○	○	○	●	●
	ECDSA	**P-256**	●	●	●	○	○	○	○
		P-384	○	●	○	●	○	○	○
		P-521	○	●	○	○	●	○	○
Support scheme	**Keygen**		●	●	○	○	○	●	○
	Sign		●	●	●	●	●	●	○
	Verify		●	●	●	●	●	●	○
FPGA familly			Virtex-7	Virtex-7	Virtex-6	Virtex-6	Virtex-6	Artix-7	Artix-7
Slice count			4,929	11,401	21,056	32,914	47,768	4,303	3,362
DSP count			0	0	250	358	514	16	182
Slice equivalent			4,929	11,401	46,056	68,714	99,168	5,903	21,562
Frequency (MHz)			135.96	109.7	100	83.3	66.6	82	87

**Only ECPM*

TABLE 3. FPGA comparison.

			Delay (s) × 1000				Area × Delay			
Support DSA		**Support scheme**	**EdDSA**	**ECDSA**			**EdDSA**	**ECDSA**		
			Ed25519	**P-256**	**P-384**	**P-521**	**Ed25519**	**P-256**	**P-384**	**P-521**
This work	2-DSAs	Keygen	6.89	7.93	-	-	33.96	39.09	-	-
		Sign	6.80	7.93	-	-	33.52	39.09	-	-
		Verify	15.87	15.87	-	-	78.22	78.22	-	-
	4-DSAs	Keygen	8.54	9.9	22.22	41.67	97.36	112.87	253.33	475.08
		Sign	8.47	9.9	22.22	41.67	96.57	112.87	253.33	475.08
		Verify	19.61	19.61	45.45	83.33	223.57	223.57	518.18	950.05
[20]	1-DSA	Keygen	-	-	-	-	-	-	-	-
		Sign	-	1.27	-	-	-	58.49	-	-
		Verify	-	1.55	-	-	-	71.39	-	-
	1-DSA	Keygen	-	-	-	-	-	-	-	-
		Sign	-	-	3.43	-	-	-	235.69	-
		Verify	-	-	3.62	-	-	-	248.74	-
	1-DSA	Keygen	-	-	-	-	-	-	-	-
		Sign	-	-	-	9.17	-	-	-	909.37
		Verify	-	-	-	12.05	-	-	-	1194.77
[45]	1-DSA	Keygen	11.11	-	-	-	65.52	-	-	-
		Sign	8.47	-	-	-	50.00	-	-	-
		Verify	12.82	-	-	-	75.68	-	-	-

as input for the ECC-based DSA core. The read results from the ECC-based DSA core are then compared with the correct results in [43] and [18]. The SHA2 core is split with the DSA processor in this experimental system. Therefore, it could be used independently.

Table 4 compares the implementation results among our ECC-based DSA core, with four different curves, with other proposals. The design is placed and routed with ROHM 180-nm CMOS technology. The design tools are Cadence with Genus for logic synthesis and Innovus for place and route. In order to perform a fair comparison, the area and maximum working frequency are converted to the corresponding value with ROHM 180-nm technology depending on the report [51], [52]. Depending on [51], the area scale is calculated by the formula 1. Where the $Area_{180nm}$ is the corresponding area of the design when using 180-nm CMOS technology, $Area_{target}$ is the area of the design when using other CMOS technology, and the $ScalingFactor$ is defined by [51]. The $ScalingFactor$ for 90-nm, 65-nm, and 45-nm CMOS technology are 6.6, 12, and 19, respectively. The authors in [51] also provide the equation 2 to calculate the corresponding delay among different CMOS technology. Where the $Delay_{180nm}$ is the corresponding delay of the design when deploying with 180-nm CMOS technology, $Delay_{target}$ is the

TABLE 4. ASIC implementation results.

			This work	[46]	[47]	[48]	[49]			[28]*	[50]*	
Number of supported DSAs			4-DSAs	1-DSA	1-DSA	1-DSA	1-DSA			EPCM	EPCM	
Support DSA	**EdDSA**	**Ed25519**	●	○	○	○	○	○	○	●	○	○
	ECDSA	**P-163**	○	●	●	○	●	○	○	○	○	○
		P-256	●	○	○	●	○	●	○	○	●	○
		P-384	●	○	○	○	○	○	○	○	○	○
		P-521	●	○	○	○	○	○	●	○	○	●
Support scheme	**Keygen**		●	○	○	○	○	○	○	○	○	○
	Sign		●	●	●	●	●	●	●	○	○	○
	Verify		●	●	●	○	○	○	○	○	○	○
Technology			ROHM 180-nm	CMOS 65-nm	CMOS 45-nm	CMOS 65-nm	1P9M 90-nm			CMOS 45-nm	1P9M 90-nm	1P9M 90-nm
#Gate (Kilo-NAND2)			377	260	-	13	170	170	170	-	170	170
Area^{+} (mm^2)			4.87	14.93	4.85	0.36	3.63	3.63	3.63	8.15	3.67	3.67
Frequency^{+} (MHz)			102	111.1	91.88	18.66	49.29	47.05	42.25	17.62	53.65	48.18

$^{+}$*Area and Frequency are estimated as [51], [52]*
**Only ECPM*

TABLE 5. ASIC comparison.

			Delay (ms)					**Area × Delay**				
Support DSA		**Support scheme**	**EdDSA**	**ECDSA**				**EdDSA**	**ECDSA**			
			Ed 25519	**P-163**	**P-256**	**P-384**	**P-521**	**Ed 25519**	**P-163**	**P-256**	**P-384**	**P-521**
This work	4-DSAs	Kegen	8.54	-	10	22.73	41.67	41.59	-	48.70	110.69	202.93
		Sign	8.47	-	10	22.73	41.67	41.25	-	48.70	110.69	202.93
		Verify	19.61	-	20	47.62	83.33	95.50	-	97.40	231.91	405.82
[46]	1-DSA	Kegen	-	-	-	-	-	-	-	-	-	-
		Sign	-	2.95	-	-	-	-	44.04	-	-	-
		Verify	-	2.80	-	-	-	-	41.80	-	-	-
[47]	1-DSA	Kegen	-	-	-	-	-	-	-	-	-	-
		Sign	-	10.75	-	-	-	-	52.13	-	-	-
		Verify	-	9.80	-	-	-	-	47.53	-	-	-
[48]	1-DSA	Kegen	-	-	-	-	-	-	-	-	-	-
		Sign	-	-	563.40	-	-	-	-	202.82	-	-
		Verify	-	-	-	-	-	-	-	-	-	-
[49]	1-DSA	Kegen	-	-	-	-	-	-	-	-	-	-
		Sign	-	5.06	13.75	-	59.99	-	18.37	49.91	-	217.76
		Verify	-	-	-	-	-	-	-	-	-	-

delay of the design when deploying with the reported CMOS technology. Depending on [51] and [52], the *DelayFactor* for 180-nm, 90-nm, 65-nm, and 45-nm technology are 79.01, 25.59, 19,15, and 13.64, respectively.

$$Area_{180nm} = Area_{target} \times ScalingFactor \tag{1}$$

$$Delay_{180nm} = Delay_{target} \times \frac{DelayFactor_{180nm}}{DelayFactor_{target}} \tag{2}$$

Table 5 compares the delay and ADP of our design with other ASIC implementations of ECC hardware. The author in [46] reports a high-performance ASIC design of ECDSA with NIST Curve P-163 Curve that can perform both sign and verify. Their reports show that their design needs 2.95-ms and 2.80-ms for each signature and verification, respectively. Nevertheless, the size occupies 14.93-mm^2. The proposed ECDSA on CMOS 45-nm of the authors in [47] requires 10.75-ms for sign generation and 9.80-ms for verifications. The area is nearly the same as ours, but it could perform only NIST P-163, while our design could perform up to NIST P-521. We also compare the ADP of our design and other proposals to evaluate the effectiveness of our processor. Although our core support four DSAs in a single implementation, it provides nearly the same ADP with other single-functional design with the same scenarios. Figure 15 illustrates the comparison of ADP among our processor with other ECC cores when generating signatures. In the Ed25519 mode, our 4-DSAs implementation provides the same ADP with ECDSA P-163 design in the other proposals,

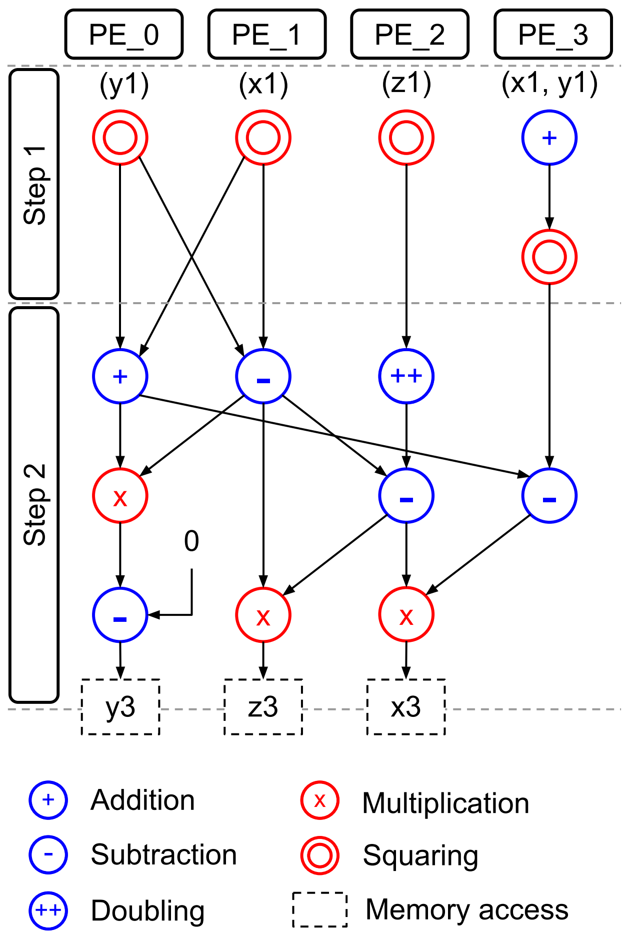


FIGURE 11. Point doubling flow over Curve25519.

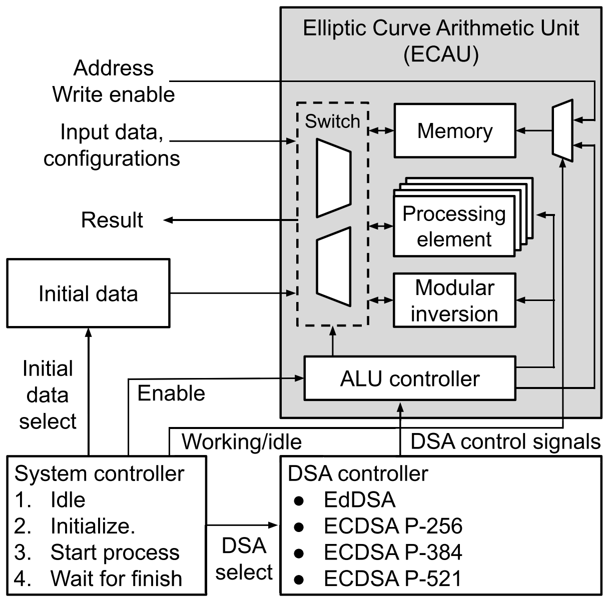


FIGURE 12. Digital signature algorithm system.

although it provides a longer key length and better security level. In the other scenarios, it is clear that our design always archives a better ADP.

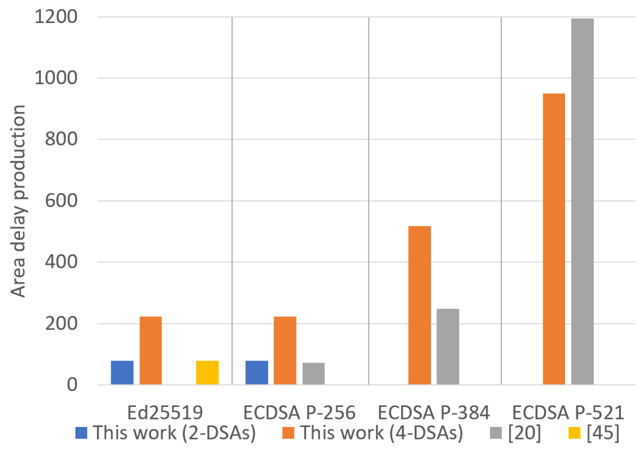


FIGURE 13. ADP comparison of sign generation on FPGA.

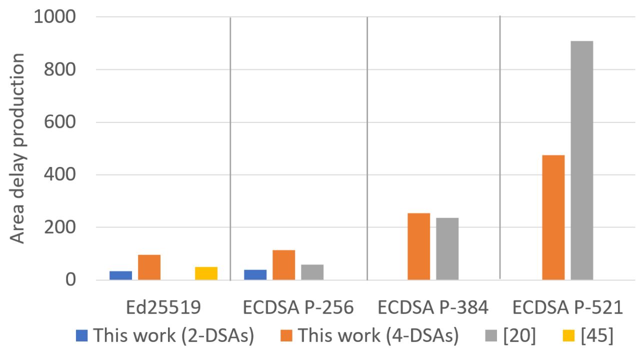


FIGURE 14. ADP comparison of verification on FPGA.

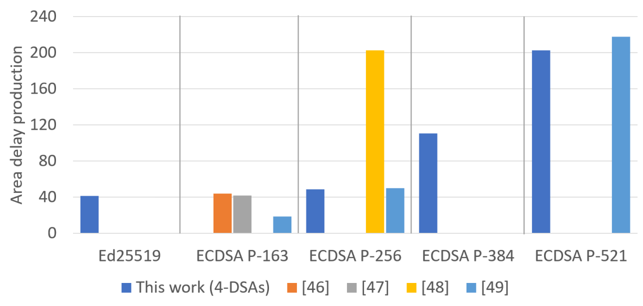


FIGURE 15. ADP comparison for sign generation on ASIC.

Nevertheless, our core is a multiple functions design and can perform up to the ECDSA P-521. On the contrary, when architecture from [46] and [47] can only perform the ECDSA with NIST Curve P-163. In 2019, R. Salarifard and S. Bayat-Sarmad proposed the first ASIC implementation of point multiplication of Curve25519 [28]. There are no previous ASIC designs on this curve before [28]. Despite this architecture only performing the point multiplication, the area when deploying on ASIC is 1.67× higher than our ECC DSA design. This implementation area is also 1.89× higher than ours when deploying on FPGA (see Table 2). There are not any proposals for the ASIC implementation on NIST Curve P-384 and P-521 despite the research of J.W. Lee et al. in [50].

They implement an ASIC design for point multiplication over the NIST Curve P-256 and another design over NIST Curve P-521. Although the design only performs point multiplication, it occupies a higher area than ours but works at a lower frequency. We successfully laid out our multi-functional design with ROHM 180-nm CMOS technology using Genus and Innovus from Cadence for synthesis and place and route, respectively. Figure 16 reveals the floorplan regions of our ECC processor architecture, which is illustrated in Figure 12. Figure 17 shows the layout with a density of 76.34%. The implementation occupies 2.2-*mm* × 2.2-*mm* and provides a maximum processing rate at 102-MHz. Table 6 summarizes the technical detail of the ECC Processor after placing and routing on ROHM 180-nm. The power consumption result is obtained from the place and route tool report. It is worth mentioning that our ASIC implementation of the combined ECC-based DSA architecture over multiple curves appears to be the first work, and no previous one exists.

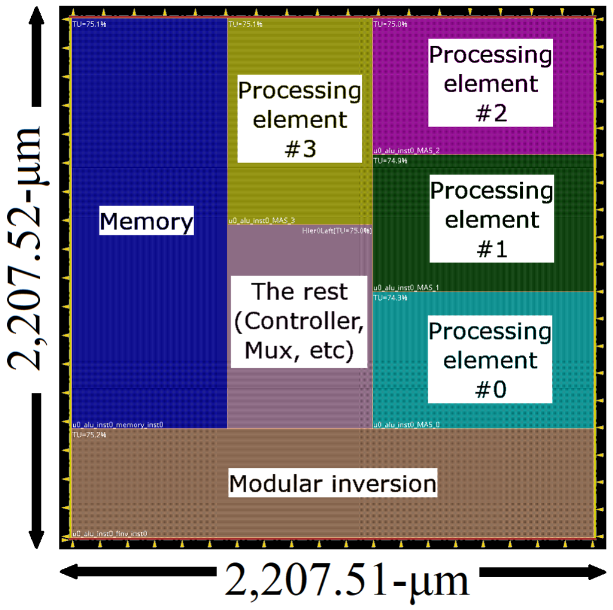


FIGURE 16. ASIC floorplanning.

TABLE 6. ASIC summary.

Process	ROHM 180-nm
#Gate (NAND2)	377,471
#Cell	167,426
#MOSFET	1,682,954
Area (μm^2)	4,873,122
Density	76.344%
Max Freq. (MHz)	102
Power (mW) *(@ 1.8V & 100MHz)*	627.4

V. LIMITATIONS AND FUTURE WORKS

Our proposal suggests an architecture that could integrate multiple ECC-based DSAs in a single design. In addition, we provide the optimized flow for both delay and communication among PEs to perform the ECPA and ECPD over Curve25519 and NIST curves to take advantage of parallel architecture. The timing attack is also considered in the design of the processor. However, there are still some limitations to this article.

First of all, because the main focus of this paper is design area-efficient multi-functional design, not all SCAs are practiced for this version. Therefore, such issues will be considered in future implementation. The second improvement is also related to the SCA. Despite the timing attack being targeted at the processing element and EPCM, the design of point inversion based on BEA is still unsafe with such attacks. Therefore, in the next update, we will add an appropriate countermeasure for this module. The next issue is about the number of DSA options. The available selection is Ed25519, ECDSA with NIST Curve P-256, P-384, and P-521. In future work, more DSAs are integrated, such as ECDH [53] and Ed448.

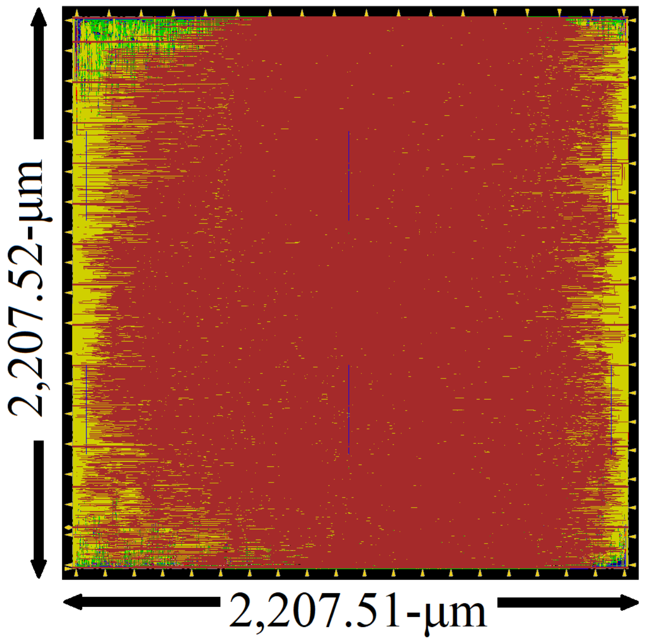


FIGURE 17. ASIC layout.

VI. CONCLUSION

This study presents the first multi-functional ECC-based DSA processor on ASIC. We propose an optimized design and calculation flows for point operations of two different elliptic curves: the NIST curve and the Edward curve. Based on it, a processor can perform four well-known ECC algorithms, which are Ed25519, and ECDSA with NIST Curve P-256/384/521 are implemented. The core is deployed on the FPGA Virtex-7 platform and ROHM 180-nm CMOS technology. The design consumes 11,401 slices and can run up to 109.7-MHz on FPGA. Another version that only runs the 256-bit DSA is implemented with only 4,929 slices and runs up to 135.96-MHz. The deployment on ASIC occupies 4.87-mm^2, which is significantly lower than the other proposal in the same condition and can run up to 102-MHz. The novel hardware architecture satisfies multiple requirements that were trade-offs from the previous designs. It is optimized to achieve an area-efficient to consume fewer resources when providing satisfactory performance. The design can be switched between different DSAs without re-implement; it could be used in many modern applications such as IoT and SDN.

For future work, we are working on higher performance for the core. Firstly, the architecture of the processing element could be optimized to fit with the low-level architecture of both FPGA and ASIC. Secondly, we target optimization of the control flow to reduce the delay of ECPM. Next, the optimized algorithm for verification for EdDSA and ECDSA will be applied in the next design. Furthermore, SCAs will also be considered to promote safety in subsequent works.

ACKNOWLEDGMENT

The VLSI chip in this study has been fabricated in the chip fabrication program through the activities of VDEC, The University of Tokyo, in collaboration with ROHM Corporation and Toppan Printing Corporation.

REFERENCES

[1] A. C. Aldaya, C. P. García, and B. B. Brumley, "From a to Z: Projective coordinates leakage in the wild," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, pp. 428–453, Jun. 2020.

[2] K. Ryan, "Return of the hidden number Problem.: A widespread and novel key extraction attack on ECDSA and DSA," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2019, pp. 146–168, Nov. 2018.

[3] L. Angrisani, P. Arpaia, F. Bonavolonta, and A. Cioffi, "Experimental test of ECDSA digital signature robustness from timing-lattice attack," in *Proc. IEEE Int. Instrum. Meas. Technol. Conf. (I2MTC)*, May 2020, pp. 1–6.

[4] Y. Shang, "Efficient and secure algorithm: The application and improvement of ECDSA," in *Proc. Int. Conf. Big Data, Inf. Comput. Netw. (BDICN)*, Jan. 2022, pp. 182–188.

[5] X. Yang, M. Liu, M. H. Au, X. Luo, and Q. Ye, "Efficient verifiably encrypted ECDSA-like signatures and their applications," *IEEE Trans. Inf. Forensics Security*, vol. 17, pp. 1573–1582, 2022.

[6] H. Xiong, "On the design of blockchain-based ECDSA with fault-tolerant batch verification protocol for blockchain-enabled IoMT," *IEEE J. Biomed. Health Informat.*, vol. 26, no. 5, pp. 1977–1986, May 2022.

[7] Y. Sazaki, M. Mulya, and M. Arisandi, "The development android-based SMS security software using ECDSA with Boolean permutation," in *Proc. 2nd Int. Conf. Wireless Telematics (ICWT)*, Aug. 2016, pp. 26–30.

[8] H. Li and Y. Ping, "A simple one-time limited authorization mechanism based on ECDSA," in *Proc. Int. Conf. Adv. Commun. Tech. (ICACT)*, vol. 2, Feb. 2010, pp. 1580–1582.

[9] R. K. Kodali, "Implementation of ECDSA in WSN," in *Proc. Int. Conf. Control Commun. Comput. (ICCC)*, Dec. 2013, pp. 310–314.

[10] A. Saini, Q. Zhu, N. Singh, Y. Xiang, L. Gao, and Y. Zhang, "A smart-contract-based access control framework for cloud smart healthcare system," *IEEE Internet Things J.*, vol. 8, no. 7, pp. 5914–5925, Apr. 2021.

[11] S. J. Basha, V. S. Veesam, T. Ammannamma, S. Navudu, and M. V. V. S. Subrahmanyam, "Security enhancement of digital signatures for blockchain using EdDSA algorithm," in *Proc. 3rd Int. Conf. Intell. Commun. Technol. Virtual Mobile Netw. (ICICV)*, Feb. 2021, pp. 274–278.

[12] R. van Rijswijk-Deij, K. Hageman, A. Sperotto, and A. Pras, "The performance impact of elliptic curve cryptography on DNSSEC validation," *IEEE/ACM Trans. Netw.*, vol. 25, no. 2, pp. 738–750, Apr. 2017.

[13] Y. Romailler and S. Pelissier, "Practical fault attack against the Ed25519 and EdDSA signature schemes," in *Proc. Workshop Fault Diagnosis Tolerance Cryptogr. (FDTC)*, Sep. 2017, pp. 17–24.

[14] D. Poddebniak, J. Somorovsky, S. Schinzel, M. Lochter, and P. Rosler, "Attacking deterministic signature schemes using fault attacks," in *Proc. IEEE Eur. Symp. Secur. Privacy (EuroS&P)*, Apr. 2018, pp. 338–352.

[15] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proc. 35th Annu. Symp. Found. Comput. Sci.*, Nov. 1994, pp. 124–134.

[16] M. Bisheh-Niasar, R. Azarderakhsh, and M. Mozaffari-Kermani, "High-speed NTT-based polynomial multiplication accelerator for post-quantum cryptography," in *Proc. Symp. Comput. Arithmetic (ARITH)*, Lyngby, Denmark, May 2021, pp. 94–101.

[17] E. Rescorla. (Aug. 2018). *The Transport Layer Security (TLS) Protocol Version 1.3*. [Online]. Available: https://www.rfc-editor.org/info/rfc8446

[18] D. Fu and J. Solinas. (Jan. 2007). *IKE and IKEv2 Authentication Using the Elliptic Curve Digital Signature Algorithm (ECDSA)*. [Online]. Available: https://www.rfc-editor.org/info/rfc4754

[19] World Health Organisation. *Things That Use Ed25519*. [Online]. Available: https://ianix.com/pub/ed25519-deployment.html

[20] B. Panjwani, "Scalable and parameterized hardware implementation of elliptic curve digital signature algorithm over prime fields," in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, Sep. 2017, pp. 211–218.

[21] T. Güneysu and C. Paar, "Ultra high performance ECC over NIST primes on commercial FPGAs," in *Proc. Crypto. Hardw. Embedded Syst. (CHES)*, Aug. 2008, pp. 62–78.

[22] P. Sasdrich and T. Güneysu, "Efficient elliptic-curve cryptography using Curve25519 on reconfigurable devices," in *Proc. Reconfigurable Comput., Arch., Tools, Appl. (ARC)*, Apr. 2014, pp. 25–36.

[23] P. Sasdrich and T. Güneysu, "Implementing Curve25519 for side-channel-protected elliptic curve cryptography," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 9, no. 1, pp. 1–15, Nov. 2015.

[24] P. Sasdrich and T. Güneysu, "Exploring RFC 7748 for hardware implementation: CURVE25519 and curve448 with side-channel protection," *J. Hardw. Syst. Secur.*, vol. 2, no. 4, pp. 297–313, Dec. 2018.

[25] J. Vliegen, N. Mentens, J. Genoe, A. Braeken, S. Kubera, A. Touhafi, and I. Verbauwhede, "A compact FPGA-based architecture for elliptic curve cryptography over prime fields," in *Proc. ASAP 21st IEEE Int. Conf. Appl.-Secific Syst., Archit. Processors*, Jul. 2010, pp. 313–316.

[26] P. Koppermann, F. D. Santis, J. Heyszl, and G. Sigl, "X25519 hardware implementation for low-latency applications," in *Proc. Euromicro Conf. Digit. Syst. Design (DSD)*, Aug. 2016, pp. 99–106.

[27] M. A. Mehrabi and C. Doche, "Low-cost, low-power FPGA implementation of ED25519 and Curve25519 point multiplication," *Information*, vol. 10, no. 9, pp. 1–16, Sep. 2019.

[28] R. Salarifard and S. Bayat-Sarmadi, "An efficient low-latency point-multiplication over Curve25519," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 66, no. 10, pp. 3854–3862, Oct. 2019.

[29] D. M. Stefano, B. Luca, C. Luca, N. Pietro, F. Luca, and S. Sergio, "Secure elliptic curve crypto-processor for real-time IoT applications," *Energies*, vol. 14, no. 15, pp. 1–20, Aug. 2021.

[30] D. M. Schinianakis, A. P. Fournaris, A. P. Kakarountas, and T. Stouraitis, "An RNS architecture of an F_p elliptic curve point multiplier," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2006, pp. 3369–3373.

[31] G. Nicolas, "A high speed coprocessor for elliptic curve scalar multiplications over $\mathbb{F}_p$," in *Proc. Crypto. Hardw. Embedded Syst. (CHES)*, Aug. 2010, pp. 48–64.

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

[33] National Institute of Standards and Technology (NIST). (1994). *Digital Signature Standard (DSS)*. [Online]. Available: https://csrc.nist.gov/publications/detail/fips/186/archive/1996-12-30

[34] V. S. Miller, "Use of elliptic curves in cryptography," in *Proc. Adv. Cryptol. (CRYPTO)*, 1986, pp. 417–426.

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

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

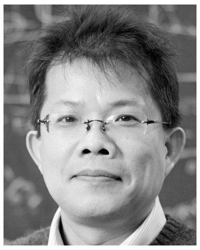
[37] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. Inf. Theory*, vol. IT-31, no. 4, pp. 469–472, Jul. 1985.

[38] P. Fei, Q. Shui-Sheng, and L. Min, "A secure digital signature algorithm based on elliptic curve and chaotic mappings," *Circuits, Syst. Signal Process.*, vol. 24, no. 5, pp. 585–597, Oct. 2005.
[39] American National Standards Institute (ANSI). (Nov. 2005). *Public Key Cryptography For The Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)*. [Online]. Available: https://standards.globalspec.com/std/1955141/ANSI%20X9.62
[40] *IEEE Standard Specifications for Public-Key Cryptography*, Standard 1363–2000, Aug. 2000, pp. 1–228.
[41] National Institute of Standards and Technology (NIST). (Jun. 2009). *Digital Signature Standard (DSS)*. [Online]. Available: https://csrc.nist.gov/publications/detail/fips/186/3/archive/2009-06-25
[42] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B. Y. Yang, "High-speed high-security signatures," *J. Crypto. Eng.*, vol. 2, pp. 124–142, Sep. 2012.
[43] S. Josefsson and I. Liusvaara. (Jan. 2017). *Edwards-Curve Digital Signature Algorithm (EdDSA)*. [Online]. Available: https://www.rfc-editor.org/info/rfc8032
[44] D. J. Bernstein and T. Lange, "Faster addition and doubling on elliptic curves," in *Proc. Adv. Cryptol. (ASIACRYPT)*, Dec. 2007, pp. 29–50.
[45] F. Turan and I. Verbauwhede, "Compact and flexible FPGA implementation of Ed25519 and X25519," *ACM Trans. Embedded Comput. Syst.*, vol. 18, no. 3, pp. 1–21, 2019.
[46] N. B. H. Youssef, W. El H. Youssef, M. Machhout, R. Tourki, and K. Torki, "A low-resource 32-bit datapath ECDSA design for embedded applications," in *Proc. Int. Carnahan Conf. Secur. Technol. (ICCST)*, Oct. 2014, pp. 1–6.
[47] A. Sghaier, M. Zeghid, C. Massoud, and M. Mahchout, "Design and implementation of low area/power elliptic curve digital signature hardware core," *Electronics*, vol. 6, no. 2, pp. 1–23, Jun. 2017.
[48] M. Ikeda, "Hardware acceleration of elliptic-curve based crypto-algorithm, ECDSA and pairing engines," in *Proc. IEEE 14th Int. Conf. ASIC (ASICON)*, Oct. 2021, pp. 1–4.
[49] J.-W. Lee, Y.-L. Chen, C.-Y. Tseng, H.-C. Chang, and C.-Y. Lee, "A 521-bit dual-field elliptic curve cryptographic processor with power analysis resistance," in *Proc. ESSCIRC*, Sep. 2010, pp. 206–209.
[50] J.-W. Lee, Y.-L. Chen, C.-Y. Tseng, H.-C. Chang, and C.-Y. Lee, "A 521-bit dual-field elliptic curve cryptographic processor with power analysis resistance," in *Proc. ESSCIRC*, Sep. 2010, pp. 206–209.
[51] Q. Xie, X. Lin, Y. Wang, M. J. Dousti, A. Shafaei, M. Ghasemi-Gol, and M. Pedram, "5nm FinFET standard cell library optimization and circuit synthesis in near-and super-threshold voltage regimes," in *Proc. IEEE Comput. Soc. Annu. Symp. (VLSI)*, Jul. 2014, pp. 424–429.
[52] A. Stillmaker and B. Baas, "Scaling equations for the accurate prediction of CMOS device performance from 180nm to 7nm," *Integration*, vol. 58, pp. 74–81, Jun. 2017.
[53] D. Stebila and J. Green. (Dec. 2009). *Elliptic Curve Algorithm Integration in the Secure Shell Transport Layer*. [Online]. Available: https://www.rfc-editor.org/info/rfc5656

BINH KIEU-DO-NGUYEN (Graduate Student Member, IEEE) received the B.Sc. and M.S. degrees from the Department of Computer Science and Engineering, Ho Chi Minh City University of Technology (HCMUT), Ho Chi Minh City, Vietnam, in 2017 and 2019, respectively. He is currently pursuing the Ph.D. degree in information and network engineering with The University of Electro-Communications (UEC), Tokyo, Japan. From 2017 to 2021, he was a Lecturer Assistant with HCMUT. Since 2022, he has been a Research Assistant with UEC. His research interests include computer architecture, hardware security, and digital systems design.

CUONG PHAM-QUOC (Member, IEEE) received the B.Sc. and M.S. degrees in computer science and engineering from the Ho Chi Minh City University of Technology (HCMUT), Ho Chi Minh City, Vietnam, in 2007 and 2009, respectively, and the Ph.D. degree in computer engineering from the Delft University of Technology, The Netherlands, in 2015. He is currently an Associate Professor and the Head of the Department of Computer Engineering, HCMUT. His research interests include FPGA-based hardware accelerators, multi/many-core systems, and the IoTs-enabled smart cities.

NGOC-THINH TRAN (Member, IEEE) received the B.E. degree in computer engineering from the Ho Chi Minh City University of Technology (HCMUT), Vietnam, in 1999, and the M.E. and Ph.D. degrees from the King Mongkut's Institute of Technology Ladkrabang, Thailand, in 2006 and 2009, respectively. He is currently an Associate Professor with the Faculty of Computer Science and Engineering, HCMUT. His research interests include network security, AI on reconfigurable devices, wireless sensor networks, and the IoT applications.


CONG-KHA PHAM (Member, IEEE) received the B.S., M.S., and Ph.D. degrees in electronics engineering from Sophia University, Tokyo, Japan. He is currently a Professor with the Department of Information and Network Engineering, The University of Electro-Communications (UEC), Tokyo. His research interest includes the design of analog and digital systems using integrated circuits.

TRONG-THUC HOANG (Member, IEEE) received the B.Sc. and M.S. degree in electronic engineering from the Ho Chi Minh City University of Science (HCMUS), Ho Chi Minh City, Vietnam, in 2012 and 2017, respectively, and the Ph.D. degree in engineering from The University of Electro-Communications (UEC), Tokyo, Japan, in 2022. From 2012 to 2017, he was a Lecturer Assistant at HCMUS. From 2019 to 2020, he was a Research Assistant at UEC. From 2019 to 2022, he was a Research Assistant at the Cyber-Physical Security Research Center (CPSEC), National Institute of Advanced Industrial Science and Technology (AIST), Tokyo, Japan. Since April 2022, he has been an Assistant Professor with the Department of Computer and Network Engineering, UEC. His research interests include digital signal processing, computer architecture, cyber-security, and ultra-low power systems-on-a-chip.