

Design of FPGA-based hardware acceleration system for elliptic curve encryption algorithm

Huiwei Yang^{1,*}

¹ Department of Information and Artificial Intelligence, Wuhu Institute of Technology, Wuhu, Anhui, 241000, China

Corresponding authors: (e-mail: yanghuiwei66@whit.edu.cn).

Abstract Traditional cryptographic algorithms have limitations in terms of efficient implementation, especially the need for hardware acceleration is more urgent. In this paper, an FPGA-based hardware acceleration system for elliptic curve encryption algorithm is designed and implemented. The system adopts Xilinx Spartan-6 series development board, and optimizes the acceleration process of elliptic curve encryption algorithm by combining the parallel computing capability of FPGA. The experimental results show that the system significantly improves the encryption speed when performing encryption operations compared with the traditional software implementation. Especially when dealing with complex image data, the system is able to deal with the equalization of the image grayscale histograms, and exhibits strong anti-attack ability. Through testing, the system's encrypted image information entropy value is close to the ideal value of 8, the NPCR value reaches 99.6814%, and the UACI value is 33.3923%, which effectively guarantees the security of the data. Based on these experimental results, the system ensures a high level of security while improving the encryption efficiency, providing an efficient solution for data protection in IoT and cloud computing.

Index Terms elliptic curve encryption algorithm, FPGA, hardware acceleration, NPCR, UACI, information entropy

I. Introduction

With the rapid development of artificial intelligence, big data analysis, Internet of Things and other fields, the demand for computational performance is increasing [1]. Traditional general-purpose processors have performance bottlenecks when dealing with complex algorithms, so the design of hardware acceleration systems has become one of the most important means to improve computational efficiency [2], [3]. And the design of hardware acceleration system for elliptic curve encryption algorithm based on field programmable gate array is gradually gaining attention [4].

Field programmable gate array (FPGA) is a kind of integrated circuit chip that can be programmed to realize specific functions [5]. Compared with fixed-function integrated circuits (ASICs), FPGAs are characterized by high programmability and flexibility [6]. The interior of FPGAs consists of a large number of programmable logic units, storage units, and programmable connectivity networks, which can be used to realize different logic functions according to requirements [7], [8]. And Elliptic Curve Cryptography (ECC) is a public key encryption algorithm based on the mathematical principle of elliptic curve, which can significantly shorten the key length and improve the encryption efficiency while guaranteeing the security and has the characteristic of resisting the quantum computing attack [9]-[12]. ECC is the cryptosystem with the highest security per unit key among all the public key cryptosystems at present [13]. Due to the characteristics of strong anti-interference, short computing time, low resource consumption, low network consumption, fast encryption speed and small hardware area, ECC can be widely used in neighborhoods such as electronic information communication and wireless sensor network terminals [14]-[16]. And the design of FPGA-based ECC hardware acceleration system not only improves its computing speed, but also convenient design and development [17], [18].

Information security is increasingly becoming a global concern, especially driven by emerging technologies such as cloud computing and the Internet of Things (IoT), the security requirements for information transmission and storage are increasing. Traditional encryption algorithms, such as RSA and AES, although perform well in some applications, gradually expose shortcomings in real-time and efficiency due to their large computational burden and high demand for storage resources. Especially when the data volume is huge and the computational requirements are high, the performance of traditional encryption algorithms appears to be out of reach. Elliptic curve cryptography (ECC) has become a hot research topic in recent years due to its short key length and high encryption efficiency. However, the computational process of ECC algorithm relies on a large number of dot product operations and large number operations, and the processing speed is often slow, especially on resource-constrained hardware platforms.

To solve this problem, FPGA (Field Programmable Gate Array) is an ideal platform for hardware-accelerated cryptographic algorithms due to its highly parallel computational capability and customizability. The research is based on FPGA platform, and the key operations in ECC algorithm (such as dot-add, dot-multiply operations, etc.) are ported to hardware through hardware-accelerated design in order to improve the processing speed. Through modular design, the system realizes hardware acceleration of functional modules including key generation, encryption and decryption. In the hardware acceleration part, especially for the optimization of finite domain arithmetic and elliptic curve arithmetic, special hardware modules, such as byte substitution, row shifting, column mixing, etc., are used, which significantly improves the execution efficiency of the algorithm. In addition, by utilizing the parallel processing capability of FPGA, the computational delay can be effectively reduced to meet the demand of large-scale data processing.

II. Elliptic Curve Cipher Theory

Along with the rapid development of cloud computing technology and Internet of Things technology, user sensitive data shows explosive growth, in order to protect the security of user private data in the network. Compared with the traditional RSA cryptographic algorithm, elliptic curve algorithm has a shorter key length, and has important advantages in computation speed, resource storage, data bandwidth, etc. It can be used to realize cryptographic primitives such as key exchange, digital signature, public key encryption and so on, and is one of the most widely used public key cryptographic techniques at present. This chapter will focus on the elliptic curve cryptography theory to provide a theoretical basis for the design of the hardware acceleration system of elliptic curve cryptography algorithm below [19].

II. A. Theory related to elliptic curves

Elliptic curves are derived from elliptic integrals as in the following equation:

$$F(x) = \int \frac{dx}{\sqrt{E(x)}} \quad (1)$$

In the above equation $E(x)$ is a third or fourth degree polynomial in x . Elementary functions cannot express such integrals, so elliptic curve functions are introduced. The Weierstrass equation (2) is generally used to represent the elliptic curve determined by this integral:

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \quad (2)$$

Denoting an elliptic curve by E , a point (x,y) satisfying the above equation is called a point on the elliptic curve E . In addition to this, a special point is included: the infinity point O .

Elliptic curves in cryptography are all based on finite fields, so different finite fields correspond to different forms of elliptic curves.

When the prime field is $F_p (p > 3)$, the corresponding Weierstrass equation is shown in equation (3). where $a, b \in F_p$ and simultaneously satisfies $4a^3 + 27b^2 \neq 0(mod p)$:

$$y^2 = x^3 + ax + b \quad (3)$$

When the domain is chosen to be a binary finite domain, the corresponding Weierstrass equation is, where $a, b \in F_{2^m}, b \neq 0$:

$$y^2 + xy = x^3 + ax^2 + b \quad (4)$$

In the above equation, when $b=0$, such elliptic curves are referred to as Koblitz curves. Such curves run faster and are widely used in cryptography.

In practice, the first step is to choose the appropriate parameters of the curve, usually $T = (p, a, b, P, n, h)$ is used to describe an elliptic curve, in order to make the algorithm better security, the parameters are generally required to meet the following conditions.

(1) p the larger the better, but too large to make the algorithm's computational efficiency is very low, time-consuming too large to make the algorithm's encryption and decryption throughput rate is reduced, usually controlled in about 200 bits can be.

(2) $p \neq n \cdot h$

(3) $p^t \neq 1(mod n), 1 \leq t \leq 20$

(4) $4a^3 + 27b^2 \neq 0(mod p)$

- (5) n is a prime number
 (6) $h \leq 4$

Based on the above conditions, the curve parameters chosen in this paper are:

- (1) The base domain $GF(2^{233})$
 (2) The irreducible polynomial in the domain is $p(x) = x^{233} + x^{74} + 1$
 (3) The equation of the curve is $y^2 + xy = x^3 + 1$
 (4) The number of points on the elliptic curve is: $\text{num}_p = 4n$, using hexadecimal to represent the positive integer n .

II. B. Basic operations on groups of elliptic curves

Take any two points $P = (x_1, y_1), Q = (x_2, y_2)$ from an elliptic curve and make a straight line intersecting with the elliptic curve at the point R' , and make a parallel line of the y-axis through the point intersecting with R , which is agreed that $P + Q = R$. At the same time, define P' as the negative element of P , and the negative element of a point has the same horizontal coordinates as that point, and the vertical coordinates are the opposite of each other. Moreover, for the infinity point O , $O + P = P$. From this equation, it is clear that the infinity point is equivalent to 0 in ordinary addition, and therefore the infinity point is also referred to as the zero element.

Take a point $P(x, y)$ on a curve and make a tangent line, which intersects the curve at another point R' . Then the point of symmetry of the R' point about the x axis is said to be the multiplication point R of the point P .

The rules of arithmetic for the group $y^2 + xy = x^3 + ax^2 + b$ on the binary finite field F_{2^m} are as follows.

- (1) unit element

If the point $P(x, y) \in E(F_{2^m})$, then:

$$P + \infty = \infty + P = P \quad (5)$$

- (2) Negative element

If the point $P(x, y) \in E(F_{2^m})$, then:

$$(x, y) + (x, -y) = \infty \quad (6)$$

The convention point $(x, -y)$ is $-P$, the negative element of the point P , and $-P$ is also a point on the elliptic curve $E(F_{2^m})$. Also $-\infty = \infty$.

- (3) Points plus

If $P(x_1, y_1) \in E(F_{2^m}), Q(x_2, y_2) \in E(F_{2^m})$, and $P \neq \pm Q$, define $-R = P + Q$, and the coordinates of $-R$ as (x_3, y_3) , then:

$$x_3 = \lambda^2 + \lambda + x_1 + x_2 + a \quad (7)$$

$$y_3 = \lambda(x_1 + x_3) + x_3 + y_1 \quad (8)$$

$$\lambda = (y_1 + y_2) / (x_1 + x_2) \quad (9)$$

- (4) Point doubling

If the point $P(x_1, y_1) \in E(F_{2^m})$, $P \neq -P$ then $2P = (x_3, y_3)$. Among them:

$$x_3 = \lambda^2 + \lambda + a = x_1^2 + \frac{b}{x_1^2} \quad (10)$$

$$y_3 = x_1^2 + \lambda x_3 + x_3 \quad (11)$$

$$\lambda = (x_1 + y_1) / x_1 \quad (12)$$

II. C. Elliptic curve discrete logarithm problem

The security of elliptic curve dense encryption algorithms relies on the elliptic curve discrete logarithm problem (ECDLP).

Remember that $E(F_q)$ is an elliptic curve, $P(x, y) \in E(F_q)$, and n is the order of the point P on the curve, the cyclic subgroup constructed from P is $\langle P \rangle$, and a point Q in the subgroup is chosen so that $Q = kP$ (k times the pointwise multiplication operation), which is an easier process, and is iterated in a loop according to Eq. This process is relatively easy and can be computed by iterating through the cycle according to the formula. However, when the points Q and P are known, it is very difficult to compute k . This problem is known as the Elliptic Curve Discrete Logarithm Problem (ECDLP), and the security of elliptic curve cryptographic algorithms formally relies on the intractability of this problem.

II. D. Cryptosystem establishment and key generation and exchange

(1) Establishment of the system

First a finite domain F_q is determined, on which an elliptic curve E is determined and a point $P(x, y)$ on the curve is chosen, the point P being of order n . The corresponding public parameters are the curve parameters a and b , the point P and its order n and the finite field F_q .

(2) Generating the key

After determining the parameters of the cryptosystem, the following operations are performed to generate the key.

- a) Select a random integer k in the closed interval $[1, n-1]$;
- b) Calculate the point $Q = kP$ (P is the point selected when the system is built), using the point doubling operation.
- c) The public key of the user contains the point Q
- d) the user's private key is the integer k chosen in step 1.

(3) Key exchange

Elliptic Curve Cryptographic Algorithm Key Exchange (ECDH) is actually the Diffie-Hellman key exchange protocol combined with elliptic curve parameters.

The key exchange is called:

- a) User A first chooses a random integer d_A as the private key satisfying $d_A \in [1, n-1]$. Then the public key $Q_A = d_A P$ is computed based on the use of dot product operation and the public key is sent to user B.
- b) User B selects a private key d_B in the same way as in step 1, computes the public key $Q_B = d_B P$ by dot product, and sends the public key to user A.
- c) User A and user B calculate the shared key using dot product operation based on the public key received from each other. A calculates the shared key as $K = d_A Q_B = d_A d_B P$, and B calculates the shared key as $K' = d_B Q_A = d_A d_B P$.

Obviously, user A and user B have the same shared key, which makes the initial key exchange successful.

II. E. Elliptic curve cipher algorithm encryption and decryption process

The encryption and decryption process of the ECC algorithm is as follows.

- (1) User A selects a suitable curve parameter to determine an elliptic curve, denoted as $E_p(a, b)$, and chooses a base point P on this curve, with n being the order of P .
- (2) User A randomly selects an integer k in the closed interval $[1, n-1]$ as the private key and computes the point $Q = kP$ as the public key.
- (3) User A sends the cryptosystem parameters to user B.
- (4) After receiving the parameters, user B encodes the data to be transmitted (plaintext) to a point N on the curve $E_p(a, b)$ and generates a random integer r ($r < n$).
- (5) User B computes two points $C_1 = N + rQ, C_2 = rP$ and sends them to user A.
- (6) User A receives the message and calculates the point $N = C_1 - kC_2 = N + rQ - krP$, since $Q = kP$. Then the original data (plaintext) can be obtained by decoding the point N based on the decoding rules.

III. Design of hardware acceleration system for elliptic curve encryption algorithm

In this chapter, the design of hardware acceleration system for elliptic curve encryption algorithm will be realized based on FPGA hardware platform and combined with elliptic curve cryptography theory in the previous chapter [20].

III. A. FPGA hardware platform

The FPGA hardware of the system under study is based on the Spartan-6 series of development boards from Xilinx, with the model number of XC6SLX125T. The development boards are equipped with a rich set of peripheral interfaces, such as Ethernet interfaces, USB interfaces, RS232 serial ports, etc. The FPGAs are used in the design of digital circuits, and are favored by developers for their programmable features.

In digital circuit design, FPGAs are favored by developers for their programmable features. FPGAs are set to work according to the code stored in the on-chip RAM, and when the development board is powered on, the master chip will load the burned program in the on-chip RAM and run the code according to the program logic. When the board is powered down, the program written on the development board still exists, and can still be used after the next power-up. When you need to modify the circuit logic, you only need to modify the FPGA program code, re-burn the program to the development board can be.

FPGA design process is generally divided into five steps:

(1) Requirements definition, according to the function of the module needs to be realized to determine the module design.

(2) Design input, generally schematic input, hardware description language input in two ways, the description language VerilogHDL, VHD.

(3) Design synthesis, this step is mainly divided into the results of comprehensive analysis and functional simulation of two parts.

(4) Design implementation, divided into static timing analysis and timing simulation.

(5) Configuration download, after completing the previous four steps, the module will be embedded into the actual use of the system test integration.

III. B. Hardware part of the overall program design

The hardware architecture of the system is shown in Figure 1. The NICE Bus consists of four handshake protocol channels, namely Request Channel, Response Channel, Memory Request Channel and Memory Response Channel. The four channels are used to receive co-processor instruction requests, feedback co-processor operation results and release co-processor operation status, co-processor request access to memory and receive memory data feedback, The four channels are used to receive coprocessor instruction requests, feedback coprocessor operation results and release coprocessor operation status, coprocessor request memory access and receive memory data feedback.

The top layer of the coprocessor is the e203_subsys_nice_core module, which contains the NICE interface and modules such as instruction decoding, instruction state control, instruction arithmetic, state matrix, and wheel key matrix cache buffer.

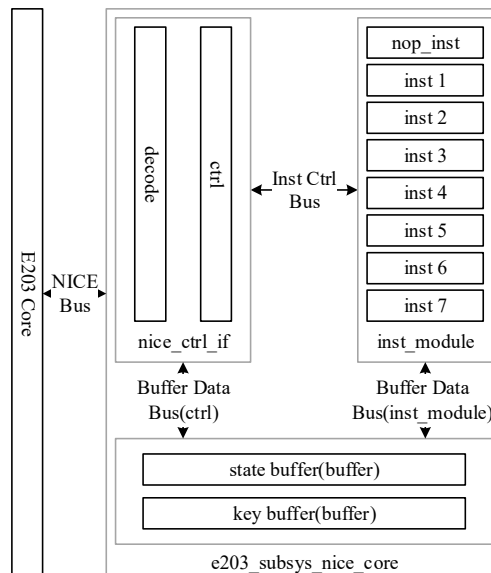


Figure 1: Hardware system architecture diagram

III. C. Implementation of hardware function modules

(1) Buffer module

The system is designed with a buffer module as a cache matrix for 16 bytes of data. In the hardware system, `state_buffer` and `key_buffer` are the instantiations of the buffer module, which are used as the cache groups of the cache state matrix and the round key matrix, respectively. The buffer module is essentially a register heap composed of four 32-bit registers, which provides a read port and a write port, and each of them is used as a cache unit of one column of the algorithmic state matrix or the round key matrix. Each 32-bit register serves as a cache unit for one column of the algorithm state matrix or round key matrix.

(2) NICE Interface Module

The Hummingbird E203 SoC provides a NICE bus to support the use of RISC-V Custom instructions to extend the co-processor for performance enhancement and power saving in specific areas. The Hummingbird E203 core interacts with the coprocessor through the NICE bus. The NICE interface module in this chapter is `nice_ctrl_if`, which contains a decode module and a ctrl module for decoding and controlling RISC-V Custom instructions, respectively.

(3) Instruction Decode Module

The instruction decode module of the hardware accelerated coprocessor, i.e., the decode module, will decode a number of signals according to the 32bits instruction encoding information.

(4) Command Control Module

The instruction control module is the ctrl module, which is mainly used to control the system according to the instruction decoding results output from the decode module, provide read/write operations to memory or buffer for load and store type instructions, and fulfill the handshake protocol timing of the 4 channels of the NICE bus.

(5) Byte Replacement Module

Byte substitution transformation is a difficult point in hardware implementation of elliptic curve encryption algorithm. In this design, we choose to use `inst1` instruction processing module to realize the byte substitution process of coprocessor AES algorithm.

(6) Row Shift Module

The row shift module is the `inst2` module, which reads the state buffer in the first clock cycle of instruction processing, rearranges the bytes of the read data in the second clock cycle, and writes the rearranged data into the state buffer in the third clock cycle. The inverse row shift process is also realized by the `inst2` module, which determines whether the current operation is row shift or inverse row shift. The process of retrograde shift is also implemented by the `inst2` module, which determines whether the currently executed operation is a row shift or a retrograde shift in a similar way as the `inst1` module: the first parameter of the inline assembly function of the `INST2` instruction is judged to be compiled and executed, and whether the NICE Bus Request Channel `rs1` signal is 0 to determine whether the current transformation is a positive or a retrograde operation.

(7) Column Mixing Module

Column mixing transformation is the hardware implementation of elliptic curve encryption algorithm is another difficulty. In this paper, we design a module, `inst4`, which is specialized in the column mixing transformation of the state matrix. Inverse column mixing transformation is also realized by `inst4` module, similar to byte replacement module and row shift module, `inst4` module also determines the positive and negative types of the current operation by judging the value of `rs1` signal.

(8) Wheel key expansion module

Key expansion is a prerequisite for realizing the encryption and decryption process of elliptic curve encryption algorithm. Except for the initial subkey $k[0]$ (i.e., $w[0] \sim w[3]$), each subkey $k[i]$ is obtained from its previous subkey $k[i-1]$ through the expansion operation, so the key to the realization of the wheel key expansion module is the design of the timing of the key expansion process and the realization of the expansion process contains three steps of word loop, byte substitution, and wheel constant iso. The key to realize the wheel key expansion module is the timing design of the key expansion process and the realization of the T function transformation in the expansion process which contains three steps: word loop, byte substitution and wheel constant permutation. In this paper, we design the wheel key expansion module, i.e., `inst6` module. `inst6` module is the corresponding operation unit of `INST6` instruction. In order to realize the T function transformation, the `inst6` module is designed with T registers to store the results of the word loop, byte substitution, and wheel constant difference or in the process of T function transformation.

(9) Wheel key addition and replacement module

The wheel key addition module is the `inst5` module. The process of wheel key addition is simply the process of bitwise dissimilarity between the state matrix and the wheel key matrix and then updating to the state matrix. In the design of this paper, this process can be specifically designed to store the state matrix state buffer and store the wheel key matrix state buffer data by bitwise dissimilarity and then write back to the state buffer process. `inst5` module maximum operation cycle `MAX_CYCLE` is 1, the first cycle of instruction processing `inst5` module through the The maximum operation cycle of `inst5` module is `MAX_CYCLE` is 1. In the first cycle of instruction processing,

inst5 module reads state buffer and key buffer to get state matrix and wheel key matrix, in the second cycle, it performs bitwise permutation of data, and in the third cycle, it writes the transformed data into state buffer to update the state matrix.

III. D. Finite domain F_p operation module design and implementation

III. D. 1) Modal addition and subtraction operations

(1) Multi-precision addition

There are two n -bit non-negative integers $(a_{n-1}a_{n-2}\dots a_1a_0)_m$ and $(b_{n-1}b_{n-2}\dots b_1b_0)_m$ is added, and multi-precision addition yields the m -decimal sum $(s_{n-1}s_{n-2}\dots s_1s_0)_m$. Here s_n is the progression, which is always equal to 0 or 1. Assuming that the value of b is 232, and that c is the progression each time a_i and b_i are added together, the relationship between a_i, b_i, c, b and s_i is as follows: a_i and s_i is the relationship between a_i and b_i have values between $[0, (2^{32}-1)]$, c is 0 or 1, and a_i and b_i are either $a_i < (2^{32}/2)$ or $a_i \geq (2^{32}/2)$, respectively. $b_i < (2^{32}/2)$ or $b_i \geq (2^{32}/2)$, and a_i, b_i, c and s_i have the following three cases:

a) When $a_i < (2^{32}/2), b_i < (2^{32}/2)$

$a_i + b_i + c < (2^{32} - 1)$, in this case s_i will not have an overflow rounding, i.e. c is equal to 0.

b) When $a_i \geq (2^{32}/2), b_i \geq (2^{32}/2)$

$a_i + b_i + c \geq (2^{32} + 1)$, in which case s_i will have the resulting progression, i.e., c equals 1.

c) When $a_i \geq (2^{32}/2), b_i < (2^{32}/2)$ or $b_i \geq (2^{32}/2), a_i < (2^{32}/2)$ $a_i + b_i + c$ the result s_i may or may not have rounding.

(2) Multiple precision subtraction

There are two n -bit non-negative integers $(a_{n-1}a_{n-2}\dots a_1a_0)_m$ and $(b_{n-1}b_{n-2}\dots b_1b_0)_m$ is subtracted, and multi-precision subtraction yields the m -inch sum $(s_{n-1}s_{n-2}\dots s_1s_0)_m$. Here s_n is a debit, which is always either 0 or 1. Equal to 0 means that a is greater than b , and equal to 1 means that a is less than b . Assuming that the value of b is 232 and c is the debit bit for each subtraction of a_i and b_i , the relationship between a_i, b_i, c and s_i is analyzed as follows:

a) When $(a_i - c) < 0$,

$c = 1, s_i = 2^{32} - 1 - b_i$.

b) When $(a_i - c) \geq 0$, and $(a_i - c) < b_i$

$c = 1, s_i = 2^{32} - b_i + a_i - c$.

c) When $(a_i - c) \geq 0$, and $(a_i - c) \geq b_i$

$c = 0, s_i = a_i - b_i - c$.

(3) Modal addition and subtraction

Finite field F_p modulus addition and subtraction and conventional addition and subtraction are basically the same, the difference is that the result must be guaranteed to be a finite field F_p in the elements, so the result of addition and subtraction to add a step "for the remainder of the modulus p " operation, that is, usually referred to as modulus operations.

The traditional modulo addition and modulo subtraction operations in a finite field F_p are to approximate the result of the addition and subtraction operations to the range $[0, p-1]$.

The modulo operation of two n -bit integers a, b on a finite field, first add the sum c of the two numbers, and then determine whether c is greater than p , if $c > p$, subtract c from p and return, otherwise return c directly.

a) Modular addition over a finite field F_p is specified as follows.

Input: $a, b \in [0, p-1]$. Output: $c = (a + b) \bmod p$.

(a) $c = a + b$;

(b) If $c \geq p$, then $c = c - p$;

(c) Return (c) .

Modulo subtraction of two n -bit integers a, b over a finite field, first find the difference c between the two numbers subtracted, and then determine whether c is less than 0. If it is less than 0, add c to p and return, otherwise return c directly.

b) Modulo subtraction over a finite field F_p is specified as follows.

Input: $a, b \in [0, p-1]$.

Output: $c = (a - b) \bmod p$.

- (a) $c = a - b$;
 (b) $c = c + p$ if $c < 0$.
 (c) return (c) .
 c) Multiprecision modulo subtraction over the finite field F_p is specified as follows.
 Input: $a, b \in [0, p-1]$.
 Output: $s = (a - b) \bmod p$.
 (a) Calculate to get (c, s) , where c is a borrowing;
 (b) If $c = 1$, then $s = s + p$;
 (c) Return (s) .

III. D. 2) Serial modulo operations

The domain elements a, b are represented as polynomials $A(x) = a_{n-1}x^{n-1} + \dots + a_1x + a_0$ and $B(x) = b_{n-1}x^{n-1} + \dots + b_1x + b_0$, then the formula for the product of elements a and b is shown below:

(1) Computational Principle:

$$c(x) = A(x).B(x) \bmod P(x) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} a_i b_j x^{i+j} \bmod P(x) \quad (13)$$

Here $P(x)$ is the polynomial representation of the finite field p , and expanding the polynomial $A(x)$ by multiplying it with $B(x)$ respectively, we have:

$$c(x) = a_{m-1}x^{m-1}B(x) + \dots + a_1xB(x) + b_0B(x) \bmod P(x) \quad (14)$$

Extracting the common factor x gives:

$$c(x) = (\dots((a_{m-1}B(x))x + a_{m-2}B(x))x + \dots + a_1B(x)) + b_0B(x) \bmod P(x) \quad (15)$$

This operation is known as serial multiplication.

The computational steps for serial multiplication are:

- a) Let $C(x) = 0$;
 b) Calculate $C(x) = (C(x) + C(x)) \bmod P(x)$ for i decreasing from $n-1$ to 0 in steps of 1.
 c) If a_i is 1, calculate $C(x) = C(x) + B(x)$; otherwise, go back to step 2) and continue until the operation is complete.
 d) FPGA implementation method

In the interface of the serial multiplication module, the input parameters $data_a$ is the multiplied number, $data_b$ is the multiplier, P is the modulus, and $start$ is the pulse signal to start the operation. The output parameter $Ready$ is the completion pulse signal and $Result$ is the result of the operation.

Serial modulo multiplication can be implemented with a shift register and a modulo addition module. The implementation defines a temporary variable c with an initial value of 0. $data_a$ is placed in the shift register and shifted bit by bit from high to low, if the bit bit is 1, then a modulo addition operation $c = (c + data_b) \bmod p$ is done, otherwise the next bit shift judgment is performed directly until all the bit bits of $data_a$ have been judged.

III. D. 3) Fast modulo multiplication operations

The algorithm used in this paper for the speed of the modulo operation of large numbers is the Montgomery modulo algorithm [21].

The Montgomery modulo algorithm expresses the number in terms of the remainder of the modulus p , converting the remainder of p into a division by 2. The division by 2 operation is a simple shift operation in a computer system, thus greatly simplifying the computation. The Montgomery modulo algorithm is described as shown in Equation (16):

$$z = MM(x, y) = x.y.R^{-1} \pmod{p} = xy2^{-n} \pmod{p} \quad (16)$$

where $a, b < p, R$ and p have to satisfy mutual prime.

Montgomery modulus multiplication needs to convert all the computed data to Montgomery domain before computation. For example, the ordinary data x, y are $x' = x.R \pmod{p}$ and $y' = y.R \pmod{p}$ after converting them to the Montgomery domain, respectively, and the data in the Montgomery domain also belong to the range of $[0, p-1]$.

The result obtained by performing the Montgomery modulo multiplication operation on the data x' and y' of two Montgomery domains:

$$c' = cR(\text{mod } p) = MM(x', y') = xyR(\text{mod } p) \quad (17)$$

c' happens to be the $c = ab(\text{mod } p)$ result converted to the Montgomery domain.

III. D. 4) Modulo inverse operations

The inverse of an operation over a finite field is one of the most complex and time-consuming operations. The inverse of a non-zero element a on a prime field Fp is written as $a^{-1} \text{mod } p$, and the Montgomery inverse algorithm will be used in this paper to compute a^{-1} .

(1) The Montgomery part of the inverse algorithm is as follows.

Input: $a \in [1, p-1]$, odd integer $p > 2, n = \lceil \log_2 P \rceil + 1$.

Output: (x, k) , where $a^{-1} 2^k \text{mod } p, n \leq k \leq 2n$.

a) Let $u = a, v = p$;

b) $x = 1, y = 0, k = 0$;

c) When $v > 0$, the loop executes

(a) If v is even, then $v = v/2, x = 2x$;

Otherwise, if u is even, then $u = u/2, y = 2y$;

Otherwise, if $v \geq u$, then $v = (v-u)/2, y = y+x, x = 2x$;

Otherwise, $u = (u-v)/2, x = x+y, y = 2y$;

(b) $k = k+1$;

d) If $x > p$, then $x = x - p$;

e) Return (x, k) .

Having found (x, k) , compute $a^{-1} \text{mod } p$ or $aR^{-1} \text{mod } p$, where $R = 2^{\pi t}$.

(2) The Montgomery inverse algorithm is specified as follows.

Input: integer a' ($a' = aR \text{mod } p$), odd integer $p > 2, R^2 \text{mod } p$.

Output: $aR^{-1} \text{mod } p$.

a) Find (x, k) , where $x = x'^{-1} 2^k \text{mod } p$;

b) If $k < \omega t$, then

(a) $x = MM(x, R^2) = (aR)^{-1} 2^k R^2 R^2 R^{-1} = a^{-1} 2^k \text{mod } p$, MM is a Montgomery modulo multiplier;

(b) $k = k + \omega t$;

Otherwise $x = MM(x, R^2) = (aR)^{-1} 2^k R^2 R^{-1} = a^{-1} 2^k \text{mod } p$;

c) $x = MM(x, 2^{2\pi t - k}) = a^{-1} 2^k 2^{2\pi t - k} R^{-1} = a^{-1} 2^{2\pi t} R^{-1} = a^{-1} R \text{mod } p$ (since $R = 2^{\pi t}$);

d) Return (X) .

Solve for $aR^{-1} \text{mod } p$, and if $a^{-1} \text{mod } p$ is required, only one more $MM(a^{-1}R, 1)$ operation is needed.

III. E. Elliptic curve arithmetic module design and implementation

The point-addition and multiplication operations in affine coordinates each require a finite-domain modulo inverse operation, which is very time-consuming. In order to solve this problem, projective coordinates need to be introduced to eliminate the modulo inverse operation. In this paper, the projective coordinates on the prime field Fp will often be used Jacobi projective coordinates.

In the Jacobi projective coordinate system, the equation of the elliptic curve on the prime domain Fp is:

$$Y^2 = X^3 + aXZ^4 + bZ^6 \quad (18)$$

An infinite point on an elliptic curve is denoted as $(0:1:0)$. At $Z \neq 0$, the point $(X:Y:Z)$ in projective coordinates corresponds to the point $(X/Z^2, Y/Z^3)$ in affine coordinates, and the negative point of the point $(X:Y:Z)$ is $(X:-Y:Z)$.

Let $P = (X_1:Y_1:Z_1), Q = (X_2:Y_2:Z_2)$ be two points on the elliptic curve with $P \neq Q$ respectively. Let $P+Q = (X_3:Y_3:Z_3)$, then:

$$\begin{aligned} X_3 &= F^2 - E^3 - 2BE^2 \\ Y_3 &= F(BE^2 - X_3) - DE^3 \\ Z_3 &= Z_1Z_2E \end{aligned} \quad (19)$$

where $A = X_2 Z_1^2, B = X_1 Z_2^2, C = Y_2 Z_1^3, D = Y_1 Z_2^3, E = A - B, F = C - D$.

Let $2P = (X_3 : Y_3 : Z_3)$, then:

$$\begin{aligned} X_3 &= A^2 - 2B \\ Y_3 &= A(B - X_3) - 8Y_1^4 \\ Z_3 &= 2Y_1 Z_1 \end{aligned} \quad (20)$$

where $A = 3X_1^2 + aZ_1^4, B = 4X_1 Y_1^2$.

Let $P = (X_1 : Y_1 : Z_1), Q = (X_2 : Y_2 : 1)$ where P is the point represented under the Jacobi projective coordinate system, Q is the point represented under the affine coordinate system, and $P \neq \pm Q$. Let the Jacobi coordinate algorithm for $P + Q = (X_3 : Y_3 : Z_3)$ be:

$$\begin{aligned} X_3 &= (Y_2 Z_1^3 - Y_1)^2 - ((X_2 Z_1^2 - X_1)^3 + 2X_1(X_2 Z_1^2 - X_1)) \\ Y_3 &= (X_1(X_2 Z_1^2 - X_1)^2 - X_3)(Y_2 Z_1^3 - Y_1) - Y_1(X_2 Z_1^2 - X_1)^3 \\ Z_3 &= (X_2 Z_1^2 - X_1)Z_1 \end{aligned} \quad (21)$$

The above algorithm can be transformed by storing some intermediate variables:

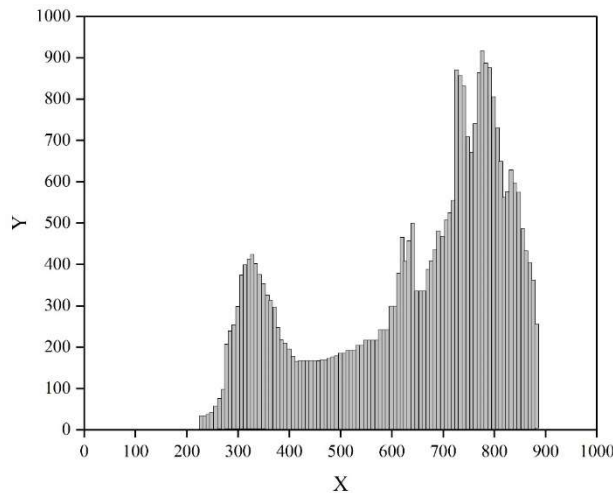
$$\begin{aligned} X_3 &= E^2 - (I + 2H) \\ Y_3 &= E(H - X_3) - Y_1 I \\ Z_3 &= Z_1 F \end{aligned} \quad (22)$$

Of these, $A = Z_1^2, B = Z_1 A, C = Y_2 B, D = X_2 A, E = C - Y_1, F = D - X_1, G = F^2, I = GF, H = X_1 G$.

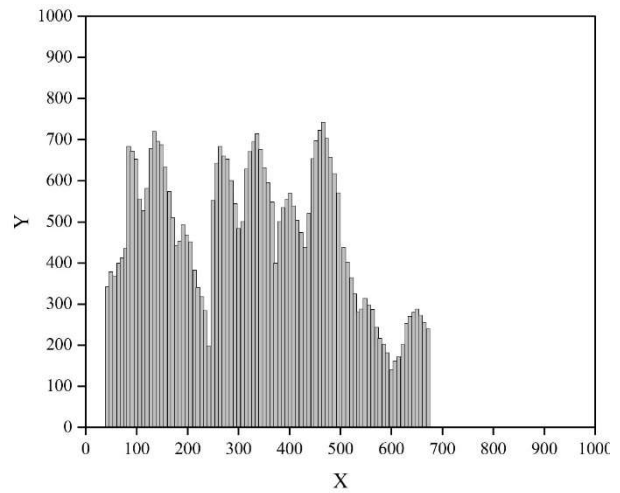
IV. Performance Experiments of Hardware Accelerated System for Elliptic Curve Encryption Algorithm

In this chapter, the performance of the algorithms used in the FPGA-based hardware acceleration system for elliptic curve encryption algorithm designed in this paper and the security performance of the overall system will be examined in terms of image gray level histogram, differential attack, information entropy and sensitivity.

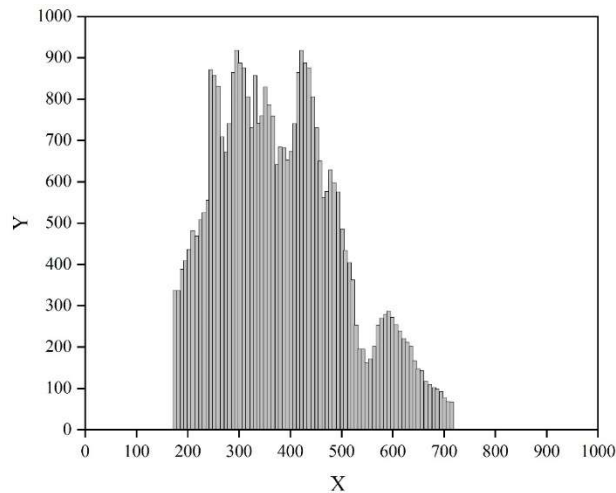
The simulation experiments are carried out in a hardware environment with Intel Core i7 CPU and 8 GB RAM using Matlab R2019a on Mac OS platform, and the test images are obtained from USC-SIPI dataset.



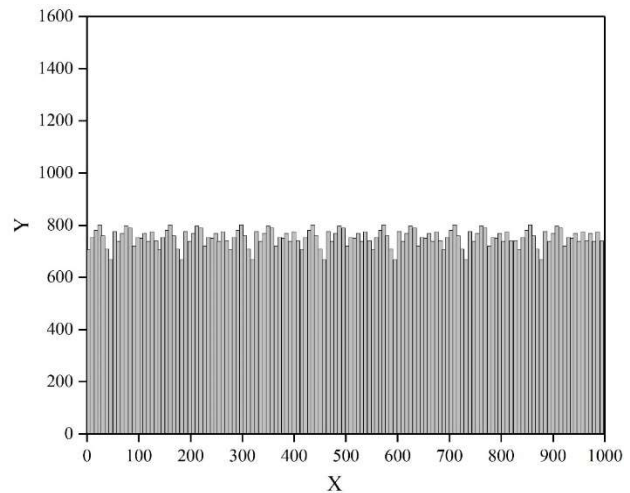
(a) R channel of the original image



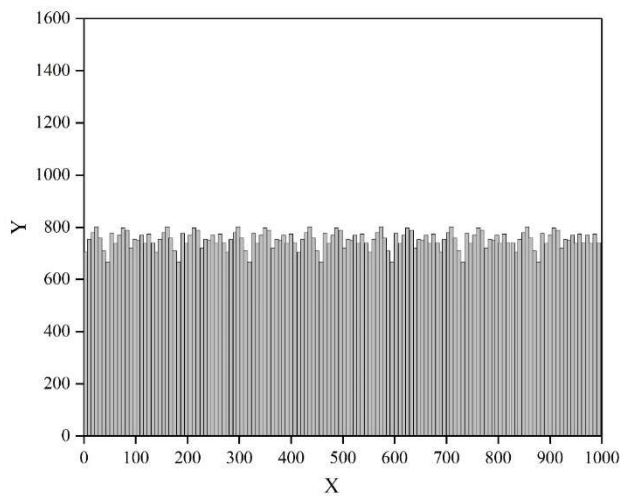
(b) G channel of the original image



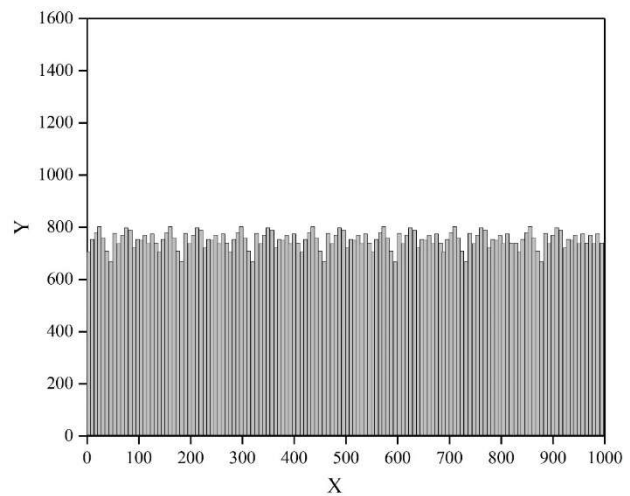
(c)B channel of the original image



(d)R channel of encrypted image



(e)G channel of encrypted image



(f)B channel of encrypted image

Figure 2: Image gray histogram

IV. A. Algorithm performance experiments

IV. A. 1) Image Gray Scale Histogram Analysis

The image histogram represents the density of the distribution of pixels in the image, and a good encryption algorithm should make the ciphertext image useful for smoother and more balanced distribution of pixels so that it can better resist statistical analysis attacks. Taking Lena image as an example, the grayscale histogram of the image before and after encryption is shown in Fig. 2. Figures (a)~(c) show the pixel statistical histograms of the R, G, and B channels, and Figures (d)~(f) show the pixel histograms of the three channels corresponding to the encrypted ciphertext image. From the figure, we can see that the distribution of the histogram of the original image is more centralized, and the distribution of the histogram of the encrypted image using the algorithm of the system in this paper is more uniform, and the overall trend is smooth; it is difficult for the cryptographic attacker to obtain effective information from the histogram, which further improves the security of the image.

IV. A. 2) Analysis of differential attacks

Differential attack is an attack on an encryption algorithm by comparing the changes in the plaintext with the ciphertext generated after encryption. In image encryption, the attacker usually makes minor changes to the original Lena image and then uses the attack target algorithm to encrypt the original image with the altered image, and then analyzes the differences between the encryption of the two different images to look for the correlation between the original image and the encrypted image. Attackers using differential attacks usually modify only one pixel of the

original image to observe the changes after encryption. If a small change is enough to cause a serious change in the encrypted image in the diffusion and confusion effects, the efficiency of the differential attack will be greatly reduced, which is almost the same as ineffective.

In order to test the effect caused by one pixel change, two measures of pixel change rate (NPCR) and Uniform Average Change Intensity (UACI) are introduced, where the larger value of NPCR indicates the better security performance of the encryption algorithm. The NPCR and UACI values of the three components of the image encrypted by the elliptic curve encryption algorithm used by the system in this paper and the RSA algorithm and the AES algorithm are shown in Table 1. It can be seen that the elliptic curve encryption algorithm used in this paper's system is closer to 1 than the RSA algorithm and AES algorithm, and the UACI value is closer to the ideal value of 0.33.

Table 1: NPCR and UACI

Algorithm	NPCR(%)			UACI(%)		
	R	G	B	R	G	B
Elliptic curve encryption algorithm	99.6814	99.6955	99.639	33.3923	33.241	33.1358
RSA algorithm	99.493	99.5281	99.4076	33.4428	33.4574	33.3728
AES algorithm	99.5482	99.4729	99.3381	33.5129	33.322	33.3758

IV. A. 3) Information Entropy and Sensitivity Analysis

The information entropy analysis of each color image before and after encryption is shown in Table 2. It can be seen that the information entropy of the encrypted image changed based on the system of this paper is close to the theoretical value, and the NPCR and UACI of the encrypted image are also close to the ideal value, which has a very satisfactory randomization characteristic, and there is rarely any accidental information leakage, so as to ensure the reliability and security of encryption.

Table 2: Information entropy and differential attack test results of different images

Pictures	R	G	B	R	G	B	NPCR (%)	UACI (%)
Lena	7.21026	7.15581	6.99309	7.99967	7.99962	7.99927	99.62748	33.48966
Logo	4.2561	3.87463	4.78309	7.99967	7.99962	7.99927	99.61142	33.53539
Pure black / white	0	0	0	7.99967	7.99962	7.99927	99.58264	33.42706
Electron microscope	7.40448	7.73889	7.90053	7.99967	7.99962	7.99927	99.60352	33.42787
Ideal value	-	-	-	8	8	8	100%	33%

In terms of information entropy and pixel correlation of encrypted images, the elliptic curve encryption algorithm used in this paper's system is compared with other encryption algorithms, as shown in Table 3. From the results in the table, it can be seen that the image encrypted by the elliptic curve encryption algorithm used in this paper's system has better correlation in all directions, and its information entropy test results are more excellent, comparing with other algorithms, the information entropy is closer to the ideal value of 8, and the NPCR and the UACI are similarly closer to the ideal value of 100% and 33%. The comparison results further prove the reliability of the elliptic curve encryption algorithm used in this paper's system in image encryption and its ability to resist statistical attacks.

Table 3: Performance comparison of different encryption methods

Algorithm	Pixel correlation coefficient			Information entropy	NPCR (%)	UACI (%)
	Horizontal direction	Vertical direction	Diagonal direction			
This article	-0.00143	0.00666	-0.00541	7.99932	99.68664	33.30822
AES	-0.01915	0.018593	0.005989	7.86941	99.60514	33.4762
DES	0.003679	0.005773	0.011368	7.99832	99.61531	33.39882
3DES	-0.00041	0.00105	0.00106	7.99012	99.6008	33.47992
RSA	0.00173	0.006807	0.002071	7.99619	99.61363	33.48001
EIGamal	-0.00082	0.00254	0.00361	7.99896	99.61928	33.40968
TSL	0.000371	0.000653	0.002213	9.99566	99.61104	33.37335
Argon2	-0.00033	0.00061	-0.00039	7.99909	99.61052	33.47972
SHA-256	0.000353	0.00527	0.00099	7.99717	99.65319	33.43395
Bcrypt	0.0012	0.00238	0.01271	7.99787	99.50077	33.39989

IV. B. Overall system performance experiment

In order to verify the overall effectiveness of the hardware acceleration system for elliptic curve encryption algorithm designed in this paper, further testing is required. The experimental platform integrates Solidity OOP, JavaScript ES4, and the hypertext preprocessor PHP5.6.31. The sample data for the experiments come from the CQ500 dataset, which includes key pairs of legitimate and illegitimate users as well as messages to be signed.

IV. B. 1) Analysis of user identification effects

Two indexes, FAR (False Identification Rate) and FRR (False Rejection Rate), are introduced to verify the performance of the system in this paper, in which FAR denotes the probability of misidentifying an illegal user as a legitimate user and FRR denotes the probability of a legitimate user being wrongly rejected. During the experimental process, the signatures and verification results between legitimate and illegitimate users are recorded in detail. The system of this paper is compared with the encryption system based on traditional AES algorithm (“AES system”), and the comparison results are shown in Table 4. From the table, it can be concluded that the system in this paper shows higher accuracy in correctly authenticating legitimate users, with lower FAR and FRR values of only 1.05% and 1.07%, compared to the higher 3.22% and 3.15% for the AES system, which indicates that the system in this paper is more reliable in verifying the identity of legitimate users. This can show that the system in this paper has higher authentication security and reliability, and can provide more powerful support in the encryption task of sensitive information.

Table 4: FAR and FRR test results

Results	System of this article	AES system
Correctly identify legitimate users (true positive)	97.88%	93.63%
Misidentification of illegal users (false positive)	1.05%	3.22%
Legitimate users were wrongly rejected (false negative)	1.07%	3.15%
FAR	1.05%	3.22%
FRR	1.07%	3.15%

IV. B. 2) Analysis of the effect of encryption of sensitive information

In this section, we will compare the performance of this paper's system with the traditional AES system and DES system in encrypting sensitive information, and focus on their data leakage under different experimental conditions. A total of five experiments are carried out, and the information provided in each experiment is of different sizes, thus obtaining the comparison results of the number of sensitive information leakage of these methods, as shown in Fig. 3. It can be seen that when the system in this paper encrypts sensitive information, the amount of sensitive information leakage obtained in five experiments is stable within 10, showing excellent information protection ability. In contrast, the leakage amount of traditional AES system and DES system fluctuates greatly and is prone to protection deviation. Thus, it can be seen that the system in this paper has better encryption performance in practical applications.

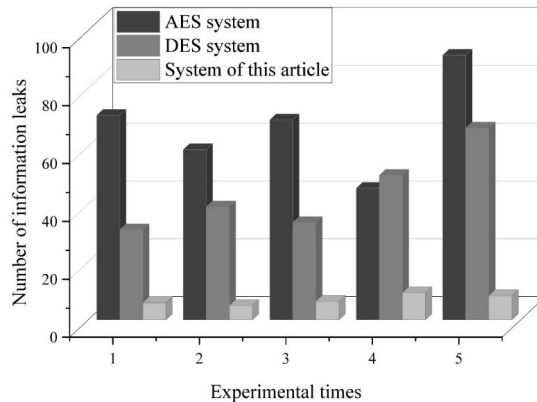


Figure 3: Sensitive information leakage of different systems

V. Conclusion

The FPGA-based hardware acceleration system for elliptic curve encryption algorithm designed in this paper is experimentally verified to have significant performance advantages. The system performs well in image encryption and successfully realizes efficient data encryption and decryption. The test results show that the system achieves a NPCR value of 99.6814% and a UACI value of 33.3923% when processing images, which means that the encryption algorithm has a high degree of randomization and is able to effectively resist differential attacks. In addition, the encryption efficiency of the system is significantly improved, and the information entropy value is close to the theoretical value of 8, which indicates that the encrypted data has a good information distribution and guarantees the security of the data. In terms of hardware acceleration, compared with the traditional computing platform, the FPGA platform not only improves the computing speed, but also reduces the energy consumption, showing good scalability and application prospects. This system provides an efficient and secure solution for hardware-based secure communication, which is especially suitable for areas requiring high speed and security, such as data protection in IoT and cloud computing environments.

Funding

This work was supported by the Anhui University Natural Science Research Project (no. 2024AH052016; no. KJ2021A1332) and Anhui Province Quality Engineering Teaching Research Project (no. 2022jyxm1778).

References

- [1] Leiserson, C. E., Thompson, N. C., Emer, J. S., Kuszmaul, B. C., Lamson, B. W., Sanchez, D., & Schardl, T. B. (2020). There's plenty of room at the Top: What will drive computer performance after Moore's law?. *Science*, 368(6495), eaam9744.
- [2] Dally, W. J., Turakhia, Y., & Han, S. (2020). Domain-specific hardware accelerators. *Communications of the ACM*, 63(7), 48-57.
- [3] Chen, A. T. Y., Biglari-Abhari, M., Wang, K. I. K., Bouzerdoum, A., & Tivive, F. H. C. (2018). Convolutional neural network acceleration with hardware/software co-design. *Applied Intelligence*, 48, 1288-1301.
- [4] Grout, I., & Mullin, L. (2018). Hardware considerations for tensor implementation and analysis using the field programmable gate array. *Electronics*, 7(11), 320.
- [5] Ruiz-Rosero, J., Ramirez-Gonzalez, G., & Khanna, R. (2019). Field programmable gate array applications—A scientometric review. *Computation*, 7(4), 63.
- [6] Badhouthiya, A., Jaffer, Z., Hussein, H. M., Juyal, A., Mittal, M., & Anand, R. (2024, February). Field Programmable Gate Array: An Extensive Review, Recent Trends, Challenges and Applications. In *2024 11th International Conference on Computing for Sustainable Global Development (INDIACom)* (pp. 1084-1090). IEEE.
- [7] Katam, N. K., Mukhanov, O. A., & Pedram, M. (2018). Superconducting magnetic field programmable gate array. *IEEE Transactions on Applied Superconductivity*, 28(2), 1-12.
- [8] Bertoldi, A., Feng, C. H., Eneriz, H., Carey, M., Naik, D. S., Junca, J., ... & Prevedelli, M. (2020). A control hardware based on a field programmable gate array for experiments in atomic physics. *Review of Scientific Instruments*, 91(3).
- [9] Adeniyi, A. E., Jimoh, R. G., & Awotunde, J. B. (2024). A systematic review on elliptic curve cryptography algorithm for internet of things: Categorization, application areas, and security. *Computers and Electrical Engineering*, 118, 109330.
- [10] Qazi, R., Qureshi, K. N., Bashir, F., Islam, N. U., Iqbal, S., & Arshad, A. (2021). Security protocol using elliptic curve cryptography algorithm for wireless sensor networks. *Journal of Ambient Intelligence and Humanized Computing*, 12, 547-566.
- [11] Maimuț, D., & Matei, A. C. (2022). Speeding-Up Elliptic Curve Cryptography Algorithms. *Mathematics*, 10(19), 3676.
- [12] Zargar, A. J., Manzoor, M., & Mukhtar, T. (2017). ENCRYPTION/DECRYPTION USING ELLIPTICAL CURVE CRYPTOGRAPHY. *International journal of Advanced Research in computer science*, 8(7).
- [13] Adeniyi, A. E., Jimoh, R. G., & AWOTUNDE, J. (2024). A review on elliptic curve cryptography algorithm for Internet of Things: Categorization, application areas, and security. *Application Areas, and Security*.
- [14] Marzouqi, H., Al-Qutayri, M., & Salah, K. (2015). Review of elliptic curve cryptography processor designs. *Microprocessors and Microsystems*, 39(2), 97-112.
- [15] Rahman, H. A. F. I. Z. U. R., & Azad, S. A. I. F. U. L. (2014). Elliptic curve cryptography. *PRACTICAL CRYPTOGRAPHY*, 147.
- [16] Wang, D., Lin, Y., Hu, J., Zhang, C., & Zhong, Q. (2023). FPGA implementation for elliptic curve cryptography algorithm and circuit with high efficiency and low delay for IoT applications. *Micromachines*, 14(5), 1037.
- [17] Marzouqi, H., Al-Qutayri, M., Salah, K., Schinianakis, D., & Stouraitis, T. (2015). A high-speed FPGA implementation of an RSD-based ECC processor. *IEEE Transactions on very large scale integration (vlsi) systems*, 24(1), 151-164.
- [18] Morales-Sandoval, M., Flores, L. A. R., Cumplido, R., Garcia-Hernandez, J. J., Feregrino, C., & Algreto, I. (2021). A Compact FPGA-Based Accelerator for Curve-Based Cryptography in Wireless Sensor Networks. *Journal of Sensors*, 2021(1), 8860413.
- [19] Chandan Goswami, Avishek Adhikari & Pinaki Sarkar. (2025). SecureID authenticated key pre-distribution scheme for IoT networks using elliptic curve cryptography. *Wireless Networks*, (prepublish), 1-22.
- [20] Ghada Elsayed & Eman S. Abass. (2024). FPGA design and implementation for montgomery multiplication algorithm using MATLAB HDL coder. *Bulletin of the National Research Centre*, 48(1), 129-129.
- [21] Muhammad Alhammami. (2024). FPGA hardware kit for remote training platforms. *Discover Education*, 3(1), 102-102.