Hindawi
Security and Communication Networks
Volume 2020, Article ID 8868672, 24 pages
https://doi.org/10.1155/2020/8868672

Research Article

Mimic Encryption Box for Network Multimedia Data Security

Xiabing Zhou,[1] Bin Li,[2] Yanrong Qi,[2] and Wanying Dong[2]

[1]*School of Computer Science and Technology, Soochow University, Suzhou 215006, China*
[2]*School of Information Engineering, Zhengzhou University, Zhengzhou 450001, China*

Correspondence should be addressed to Bin Li; cctvlibin@163.com

Received 19 August 2020; Revised 25 September 2020; Accepted 13 October 2020; Published 29 October 2020

Academic Editor: Zhihua Xia

With the rapid development of the Internet, the security of network multimedia data has attracted increasingly more attention. The moving target defense (MTD) and cyber mimic defense (CMD) approaches provide a new way to solve this problem. To enhance the security of network multimedia data, this paper proposes a mimic encryption box for network multimedia data security. The mimic encryption box can directly access the network where the multimedia device is located, automatically complete the negotiation, provide safe and convenient encryption services, and effectively prevent network attacks. According to the principles of dynamization, diversification, and randomization, the mimic encryption box uses a reconfigurable encryption algorithm to encrypt network data and uses IP address hopping, port number hopping, protocol camouflage, and network channel change to increase the attack threshold. Second, the mimic encryption box has a built-in pseudorandom number generator and key management system, which can generate an initial random key and update the key with the hash value of the data packet to achieve "one packet, one key." Finally, through the cooperation of the ARM and the FPGA, an access control list can be used to filter illegal data and monitor the working status of the system in real time. If an abnormality is found, the feedback reconstruction mechanism is used to "clean" the FPGA to make it work normally again. The experimental results and analysis show that the mimic encryption box designed in this paper has high network encryption performance and can effectively prevent data leakage. At the same time, it provides a mimic security defense mechanism at multiple levels, which can effectively resist a variety of network attacks and has high security.

1. Introduction

With the rapid development of information, network, communication, image processing, and other technologies, the security requirements of network multimedia are becoming increasingly more urgent [1, 2]. To obtain economic benefits, occupy commercial competitive advantages, and achieve certain military needs, attackers usually use techniques such as reverse analysis, vulnerability attacks, and network sniffing to steal sensitive data, thereby triggering data security protection problems.

The life cycle of network multimedia data can be divided into four main parts: collection, transmission, storage, and processing. Regarding these functions, in order to protect the confidentiality of data transmission and storage and prevent the leakage of private data, it is necessary to encrypt the data [3]. However, the current network equipment and components have a large number of security vulnerabilities, allowing attackers to directly bypass the encryption link and steal the original data [4]. Second, security protection for massive amounts of multimedia data, including images, voices, and video, entails higher requirements on the efficiency of cryptographic algorithms. In the process of high-speed data transmission, traditional CPU encryption algorithms make it difficult to meet computing requirements [5]. Thus, a more efficient implementation of cryptographic algorithms is required in order to improve encryption speed. Finally, due to the staticity and similarity of traditional network components [6], such as the same hardware equipment, network protocol, fixed IP address, and port number, it is easy for an attacker to study the network operating rules, dig out security flaws, and conduct detection and continuous intrusion.

To improve the security of the multimedia network, the academic community has proposed a moving target defense (MTD) [7] technology and a cyber mimic defense (CMD) technology [8]. On the basis of MTD, CMD technology provides a dynamic heterogeneous redundancy (DHR) architecture, which uses the active transformation of functionally equivalent heterogeneous executive bodies to change the components of the information system, therefore realizing network, platform, environment, software, structure, and data dynamic changes or migrations. For attackers, it is difficult to observe and predict the target changes; therefore, system security risks are greatly reduced. At the same time, CMD introduces a negative feedback mechanism. According to the configured negative feedback control strategy, if the current system is found to be abnormal, the system will be cleaned by self-reorganization and reconstruction. In addition, new functionally equivalent heterogeneous executors will be randomly selected "online." Obviously, CMD not only greatly increases the difficulty and cost of attacks but also can detect the attack behavior of successful intrusions in real time.

Among various network security defense measures, the security of the hardware structure and the operating system is the foundation, and cryptography is the key technology. In this paper, combining the concepts of MTD and CMD, using the FPGA as the hardware platform, a mimic encryption box is designed. This device provides dynamic encryption, network structure transformation, and a feedback reconstruction mechanism in order to realize the security protection of multimedia data.

Our main contributions in this paper can be outlined as follows:

(1) A mimic encryption box with dynamics, diversity, and randomness is designed and will be able to directly connect to the network where the multimedia device is located and provide secure encrypted transmission of data.

(2) The TCP/IP layer data encryption and variable network configuration are realized on an FPGA, and system management and a feedback mechanism are realized on an ARM. In this system, software and hardware work cooperatively.

(3) The reconfigurable underlying cryptographic algorithm, pseudorandom number generator, packet filtering, and storage functions are optimized on the FPGA, and the programmability and high anti-interference of the FPGA make it difficult for attackers to establish a continuous and reliable attack chain.

(4) The hash value of the packet is used as the key to achieve "one packet, one key," and autonegotiation is used to change the IP address, port number, and protocol type in order to achieve dynamic network transformation. Through cooperation between the ARM and the FPGA, abnormal condition-related "cleaning" and the reconfiguration of FPGA are achieved.

(5) The encryption and decryption performance, network performance, security, and antiattack of the mimic encryption box are evaluated from several aspects.

The remainder of this paper is organized as follows: Section 2 introduces the existing network multimedia protection technologies. Section 3 describes the design of the mimic encryption box in detail. Section 4 analyzes and evaluates the mimic encryption box from multiple perspectives. Section 5 discusses the applicability and limitations of this method. Finally, Section 6 concludes this paper.

2. Related Work

Regarding MTD research, Aydeger et al. [9] analyzed crossfire attack planning and utilized the analyzed results to develop the defense mechanism that in turn reorganizes the routes in such a way that the congested links are avoided during packet forwarding. In addition, for use when implementing MTD mechanisms via route mutation, Aydeger et al. [10] proposed various virtual shadow networks created through network functions virtualization (NFV), which can dynamically change the routes for specific reconnaissance packets so that attackers will not be able to easily identify the actual network topologies. Zeitz et al. [11] explored the uses of a micromoving target IPv6 defense (μMT6D). The μMT6D is designed to work on low-power and low-resource devices and can prevent targeted attacks through rotating the IPv6 address. Wang et al. [12] proposed a network defense method based on random domain name and address mutation (RDAM). This method increases the scanning space of the attacker through a dynamic domain name method and reduces the probability that a host will be hit by an attacker scanning IP addresses. Zheng and Namin [13] presented the results of an analysis performed on simulating a simplified attack scenario against hosts on a network. They investigated the influence of the host IP address change rate and host complexity on the success rate of attacks. Hong and Kim [14] incorporated MTD techniques into a security model, namely, a hierarchical attack representation model (HARM), in order to assess the effectiveness of the techniques. Zhao et al. [15] proposed an SDN-based double hopping communication (DHC) approach. The DHC approach is able to increase the overhead of a sniffer attack, as well as the difficulty of communication data recovery. Jafarian et al. [16] proposed random host address mutation (RHM), which uses hierarchical fast transitions based on the address space and the IP address in order to distort attacker reconnaissance and deter attacks. On this basis, Jafarian et al. [17] proposed the proactive-adaptive defense technique, which monitors an attacker's behavior in real time and performs active address hopping, thus significantly raising the bar against stealthy scanning.

In terms of CMD, Hu et al. [18] designed and implemented a mimic network operating system (MNOS), an active defense architecture based on mimic security defense, in order to ensure SDN control plane security. This

architecture can effectively reduce the probability of successful attack and has good fault tolerance. Then, Hu et al. [19] introduced the mimic defense (MD) framework and detailed the "dynamic, heterogeneity, and redundancy" core mechanism. Their results showed that MD can significantly increase the difficulty faced by attackers and enhance the security of cyber systems. Ma and Zhang [20] described the mimic defense system formally and, through results from Monte Carlo simulations, analyzed the security effects of redundancy in mimic defense systems. Qi et al. [21] proposed Mcad-SA, an aware decision-making security architecture with multiple controllers, which exploits heterogeneity and redundancy from different controllers to prevent an attack proactively. Based on the mimic defense theory and technology, Liu et al. [22] proposed a framework against zero-day attacks. To protect the security of distributed storage systems, Li et al. [23] presented a storage architecture for mimic defense (SAMD). This architecture adopts a heterogeneous multirandom coding defense mechanism to actively and dynamically defend against indeterminate attacks.

In terms of network encryption, Abusukhon et al. [24] focused on data encryption techniques and proposed a new method for data encryption based on encrypting the plain text into a white page image. In addition, Abusukhon et al. [25] proposed a Diffie–Hellman text-to-image encryption algorithm (DHTTIE), adding a new security level to the TTIE algorithm. Tang et al. [26] proposed a dynamic three-layer encryption scheme based on DES and network coding, with a low-complexity partial key update mechanism, which increases its adaptability to various cyber conditions. Khan et al. [27] further reduced the cost of a network coding mechanism by reducing the size of data used for permutation. They also proposed an algorithm for key generation and random permutation confusion key calculation. Jiang et al. [28] presented a compromising method to take both the security level and the speed of data transmission into account by means of mixing the RSA and DES algorithms. In addition, the security interceptor of Spring Security was extended, and a series of security filters were added to keep Web attackers away. Li et al. [29] proposed a new attribute-based data-sharing scheme suitable for resource-limited mobile users in cloud computing. For the sake of data security, a Chameleon hash function was used against adaptive chosen-ciphertext attacks. Jia et al. [30] designed an identity-based anonymous authenticated key agreement (AAKA) protocol for the mobile edge computing environment. This protocol achieves mutual authentication in only a single message exchange round and assures both user anonymity and untraceability. Indira et al. [31] proposed a standard two-phase implementation of round key- and random key-based cryptosecurity encryption standard (R2R-CSES) for improving security system in a cloud environment.

In many security applications, a variety of defense technologies were realized based on FPGA. Maciel et al. [32] proposed a high-performance and energy-efficient reconfigurable FPGA-based K-means/K-modes architecture for network intrusion detection. Joseph et al. [33] studied the efficiency of an intrusion detection system by using an FPGA with a string-matching system design and a predecoder finite state machine for use in the high-speed network intrusion detection system. Lin et al. [34] introduced the design of a Gigabit Ethernet firewall based on FPGA. The FPGA functions were implemented to achieve legitimacy in network packet inspection and for internal network protection. Keni and Mande [35] used the highly parallelized structure of FPGA to form a rule set for allowing incoming and outgoing IP addresses to filter the IPv4 protocol. Ricart-Sanchez et al. [36] proposed a fully functional, FPGA-based 5G firewall that is capable of effectively detecting cyber-attacks in 5G multitenant scenarios with user mobility support.

In summary, although the use of IP address, port number hopping, and other mechanisms increases the difficulty of the attack to some extent, if plaintext transmission is used, the attacker can still obtain useful information. Second, simple network data encryption cannot resist key exhaustive attacks and ciphertext-only attacks. Third, FPGA is a suitable and popular hardware platform for many network security applications. In addition to being used in firewalls and packet inspection, it can also be used in MTD and CMD. The combination of ARM and FPGA can be used to expand the attack surface and improve security. Therefore, this paper combines MTD, CMD, network encryption, and hardware protection technologies and proposes a dynamically reconfigurable mimic encryption box to solve the abovementioned problems.

3. Design Method of the Mimic Encryption Box

3.1. Overall Structure. The mimic encryption box is located between the network multimedia device and the user. It provides secure communication services to the user by binding the terminal multimedia device. The system architecture is shown in Figure 1. In the architecture, there are management and feedback modules running on an ARM. These modules mainly communicate with the key management center to complete the following: the initialization of the parameters, key distribution, the generation of FPGA status statistics, self-reconstruction cleaning, and the dynamic loading of bitstreams. In addition, an FPGA is mainly used for the realization of core cryptographic algorithms, including hash algorithms, symmetric encryption algorithms, and asymmetric encryption algorithms. By using the reconfigurable ability of the FPGA, various keys are generated pseudorandomly, and cryptographic algorithms are dynamically invoked to complete efficient data encryption and decryption processing. Second, through the access control list and memory management, the rule filtering and storage of data are realized. Finally, multiple 10G and 1G communication network ports are integrated on the FPGA, and the network ports and channels are dynamically switched according to the configuration of the ARM, making full use of the flexibility and scalability of the FPGA to confuse attackers and prevent attacks such as network sniffing.

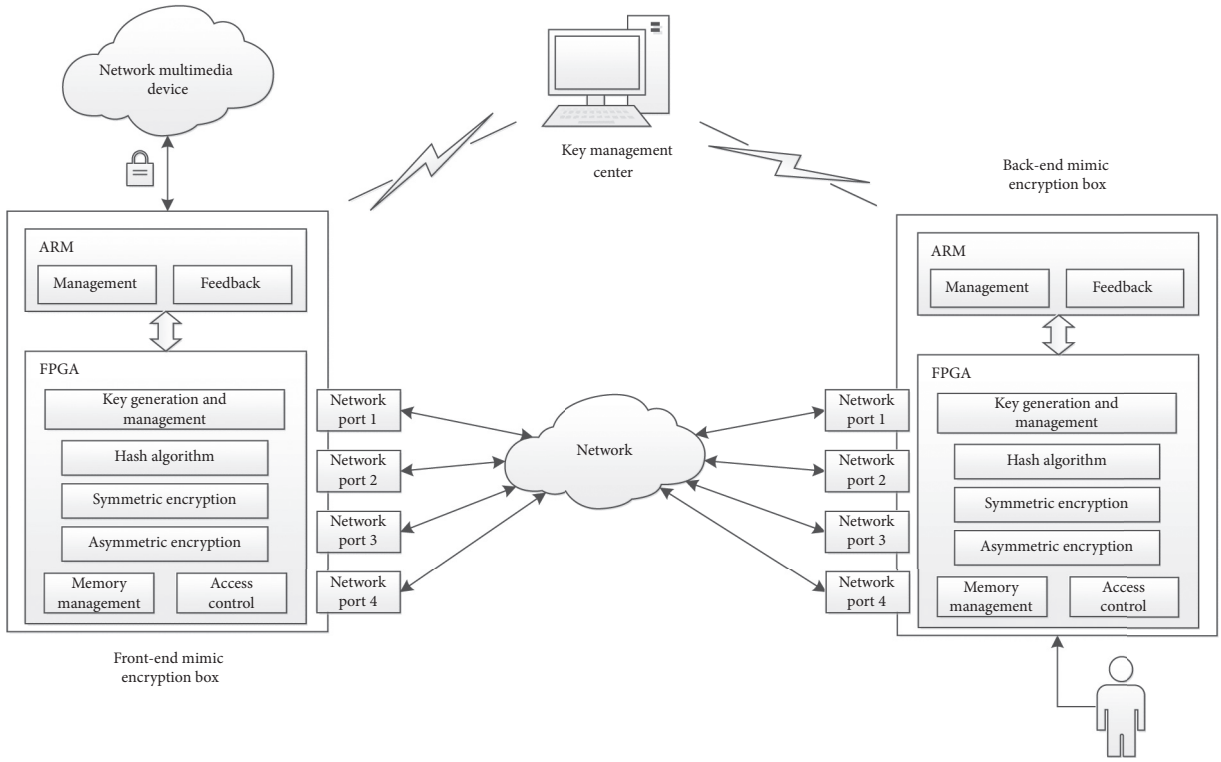

Figure 1: Mimic encryption box system architecture.

3.2. Work Flow. Currently, network multimedia devices mostly work at the TCP/IP layer in order to provide stable and reliable data transmission. Therefore, in this paper, mainly TCP/IP data encryption and camouflage is completed. As shown in Figure 2, the comparison before and after TCP/IP packet encryption is given. Here, the IP address, protocol, and port number are hopped and disguised. Then, a 1-byte PID field and a 4-byte CRC32 field are added: the PID is used to address the key, and the CRC32 is used to calculate the TCP checksum. Finally, the other TCP fields, the data, the pad, and the CRC32 are encrypted.

The mimic encryption box completes the packet encryption based on the principles of randomization, dynamization, and diversification. As shown in Figure 3, it is mainly composed of the MAC layer interface, packet parsing, access control, encryption and decryption algorithms, key generation management, packet encapsulation, memory, arbitration, state information collection, parameter initialization, and the ARM system. Among the components, through filtering rules, access control manages the plaintext path, the ciphertext path, discarding, and other processing and forwarding to the ARM. At the same time, in the front-end mimic encryption box, network port F is used to connect to the network multimedia device, and network port B is connected to the external network.

The encryption process of the entire system is as follows:

(1) The key management center establishes a secure tunnel with the ARM of both communicating parties to obtain the mimic encryption box information and distribute the initial information and working key.

(2) The communication parties negotiate to complete the dynamic configuration of the IP address, port number, protocol type, encryption algorithm, hash algorithm, and network interface, and the system parameters are initiated by the ARM. Then, the initial encryption key is generated by the pseudo-random number generator.

(3) After network port F receives the packet, it parses the packet and forwards it to the corresponding processing channel by the filtering rules.

(4) In the encryption processing, the PID and CRC32 fields are added to the TCP/IP packet, and the corresponding initial key for the encryption is selected. Then, a hash operation on the first 64 bytes of the encrypted packet is performed to generate a new key, and the key corresponding to the serial number is updated.

(5) The negotiated IP address, port number, and protocol type are used to disguise and encapsulate the original packet.

(6) The encrypted packets, plaintext packets, and ARM packets are cached in memory and, then, sent out from network port B.

(7) Network port B receives the packet from the external network, parses it, and forwards it to the corresponding processing channel according to the filtering rules.

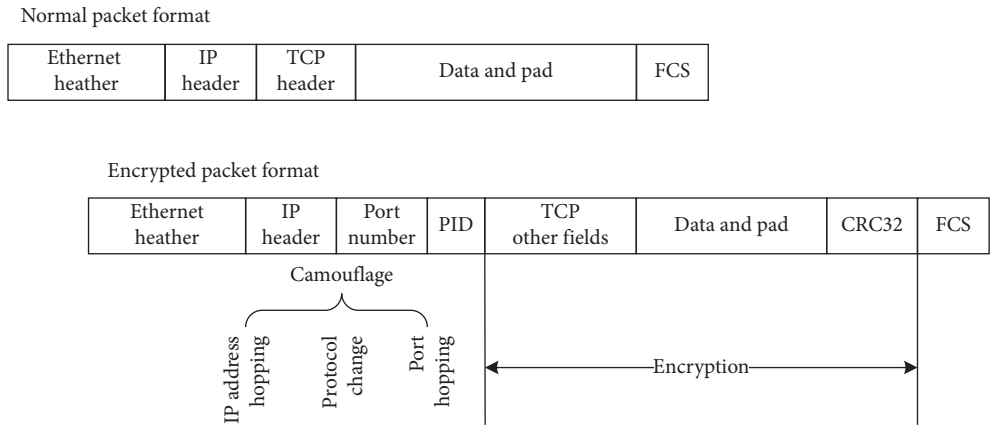

Figure 2: Comparison of normal and encrypted packet formats.

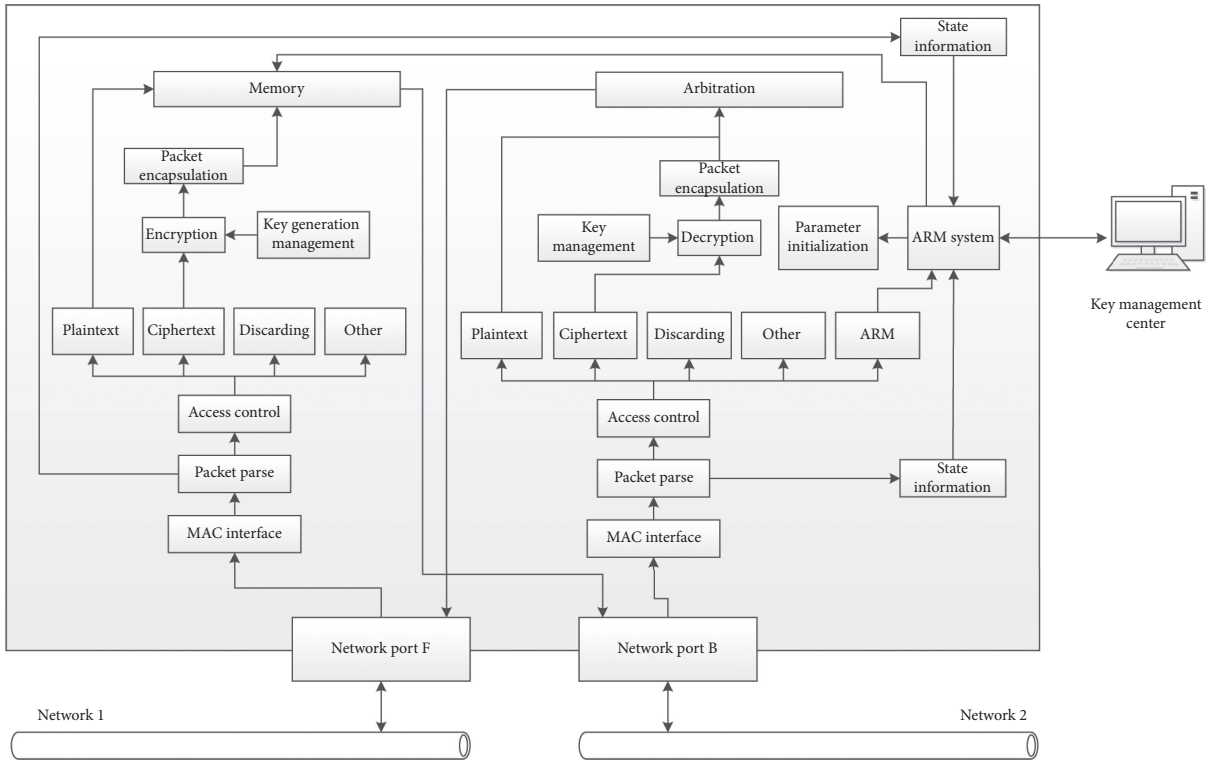

Figure 3: Front-end mimic encryption box structure.

(8) In the decryption processing, the key corresponding to the PID field is selected in order to decrypt and check the CRC32, and at the same time, the corresponding key from the hash value of the first 64 bytes of the current encrypted packet is directly updated.

(9) The camouflage is removed, the original IP address, port number, protocol type is restored, and the packet is encapsulated.

(10) In the ARM path, the data are transmitted to the ARM system, and the ARM system hands them over to the key management center to complete the corresponding processing.

(11) Then, the arbitration module is used to directly send the decrypted and encapsulated packet to the network multimedia device.

In the process of communication between the two sides, the packet information of network port F and B is obtained in real time, and after being collected by the ARM system, it is sent to the key management center to realize the real-time monitoring of both sides.

For the back-end mimic encryption box, the functional modules are similar to the front-end, and the workflow is basically the same, but the network connection method is slightly different. Specifically, network port F is connected to the external network to provide the decryption channel and

ARM management; network port B is connected to the user and provides an encrypted channel.

3.3. Optimization of Core Encryption Algorithm

3.3.1. Hash Algorithm. A hash algorithm is an irreversible one-way function that can output any length of data. At present, the commonly used hash algorithms are MD5, SHA1, SHA256, SHA512, RIPEMD160, and SHA3. These hash functions are all based on logical operations. The data are filled, grouped, and then, iteratively compressed by the round function, and the result is generated after n rounds of calculation. Since the structure of each round is similar, it can be implemented in a full-pipeline parallel manner.

Here, according to the number of iterations of the hash algorithm, all the loops are expanded [37, 38] to form a full-pipeline structure. When it is working at full capacity, in the overall pipeline, each clock cycle can calculate a set of hash values. Then, precalculation and carry-save adders (CSA) are used to optimize and reconstruct the round function to reduce the critical path delay.

We take SHA1 as an example. The SHA1 algorithm fills the initial information into 512 bits and initializes it to 16 32-bit groups $w[15:0]$. Let $A = 0x67452301$, $B = 0xEFCDAB89$, $C = 0x98BADCFE$, $D = 0x10325476$, and $E = 0xC3D2E1F0$ be the initial link variables, and let a, b, c, d, and e be 5 intermediate variables used to perform 80 rounds of iterative computations. Each iteration is performed as follows:

$$\begin{aligned} a &= a\_\text{next};\\ b &= a;\\ c &= c\_\text{next};\\ d &= c;\\ e &= d,\\ a\_\text{next} &= \{a[26:0], a[31:27]\} + f_t + e + k_t + w_t;\\ c\_\text{next} &= \{b[1:0], b[31:2]\}, \end{aligned} \tag{1}$$

where f_t is a nonlinear function, k_t is a constant, and w_t is a grouped data block.

Finally, the 160-bit hash value cascaded output is given by $a = a + A$; $b = b + B$; $c = c + C$; $d = d + D$; and $e = e + E$.

Clearly, b, c, d, and e can be obtained directly via value passing, while a requires complex operations, and the delay consumption is concentrated along the critical path of a. For the FPGA, the delay of addition is much larger than the bit operation. Therefore, to reduce the use of adders, we define the CSA as follows:

$$\begin{aligned} \text{CSA}(x, y, z) = {} & \left(x \hat{} y \hat{} z\right) + (((x\&y)|(x\&z)|(y\&z)) \\ & \ll 1) = x + y + z. \end{aligned} \tag{2}$$

At the same time, a variable g is introduced for precalculation, as follows: $g = \text{CSA}(d, k_{t+1}, w_{t+1})$. That is, we use the result d of the next round e and k_{t+1}, w_{t+1} to calculate g in advance. Then, the calculation of a can be simplified as

$$a\_\text{next} = \text{CSA}\left(\{a[26:0], a[31:27]\}, f_t(b, c, d), g\right). \tag{3}$$

Figure 4 shows an optimized round function structure.

Then, a unit kernel is built with round functions, the kernels are interconnected by registers, and all kernels are executed in parallel. The overall structure of the hash algorithm is shown in Figure 5.

Comprising FPGA storage resources, the registers are widely distributed. BRAM is located in a fixed area, and mixed storage is adopted, enabling the full utilization of FPGA resources and effectively shortening the critical path delay. Therefore, in Figure 5, the initialization variables are stored by using Shift RAM. Due to its small scale and scattered values, the constant list uses a direct assignment strategy. For grouped data blocks, a two-dimensional register array is used, values are assigned through circular shifts in one-dimensional space, and values are transferred through a register copy in two-dimensional space, thereby realizing data multiplexing and reducing data overlap. For the output of the results, to achieve a balance between resources and performance, data cascading is used, as it is more conducive to constraining the data concentration in a logical area.

In addition, by reconstructing the round functions of different hash algorithms and using units and hybrid storage methods, a hash algorithm with higher performance and better expansibility can be formed to meet the needs of various encryption computing.

3.3.2. Symmetric Encryption Algorithm. The current mainstream encryption algorithms are 3DES, AES128, AES256, Twofish, and Serpent. The encryption scheme based on the FPGA chip level is fast, safe, and of low cost. As the FPGA technology is reconfigurable, through user programming to change the circuit structure on the chip, different encryption and decryption algorithms can be realized. In this paper, AES128 is taken as an example. Based on the idea of reconfigurability, the cryptographic algorithm is modularized, providing the encryption and decryption architecture of AES.

The main modules of AES include KeyExpansion, AddRoundKey, SubBytes, ShiftRows, and MixColumns. The SubBytes module enables the realization of the obfuscation principle, and ShiftRows and MixColumns modules mainly allow realization of the diffusion principle [39, 40]. The AES decryption algorithm key selection sequence is exactly the opposite of encryption, and the key expansion process is irreversible. Therefore, to ensure that encryption and decryption have the same execution cycle, all round keys need to be generated at once. The AES structure is shown in Figure 6.

The AES single-round encryption process is shown in Figure 7. Each round of operations is compressed to 1 clock cycle in a cascaded manner. The entire encryption process requires 23 clocks, of which the key expansion is 11 clocks and encryption and decryption is 12 clocks. In addition, compared with the LUT method, the method in which the BRAM embedded in the FPGA is used to realize the storage

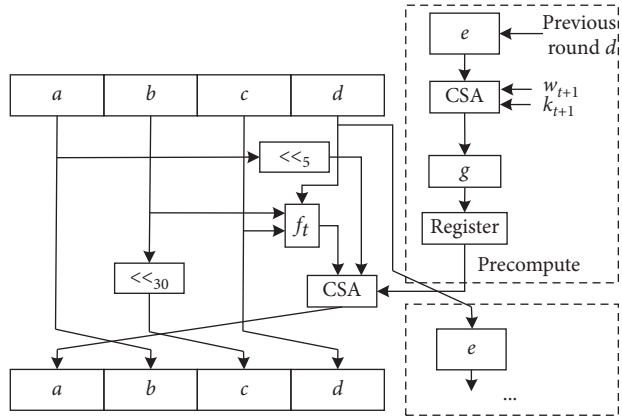

Figure 4: SHA1 single iteration structure.

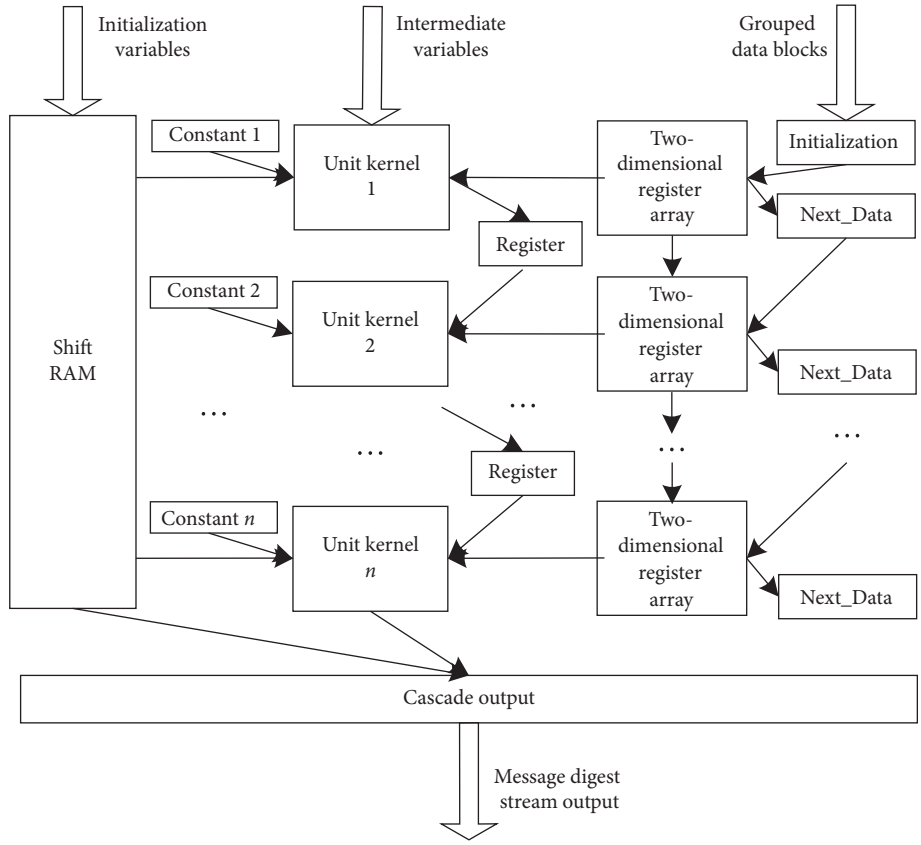

Figure 5: Full-pipeline architecture of the hash algorithm.

of the Sbox is better, as this method can reduce the occupation of resources and increase the routing frequency.

Finally, symmetric encryption algorithms have four commonly used encryption modes: ECB, CBC, CFB, and OFB. Among them, in ECB and CBC, the data are divided into blocks and padding operations are performed, while in CFB and OFB, change block ciphers can be changed into self-synchronized stream ciphers. Therefore, to pad the last data block, the CFB and OFB encryption methods are used. In addition, the previous block of encrypted data are regarded as the IV (initial vector), and the current data are XORed with IV after calculation and output.

3.3.3. Elliptic Curve Algorithm. In the prime domain, the description of the elliptic curve $E(F_p)$ is as follows: $y^2 = x^3 + ax + b \pmod{p}$ [41]. The security of the elliptic curve cryptosystem is mainly based on the difficulty of point multiplication inversion. Point multiplication, also called a multiple point operation, refers to the multiplication

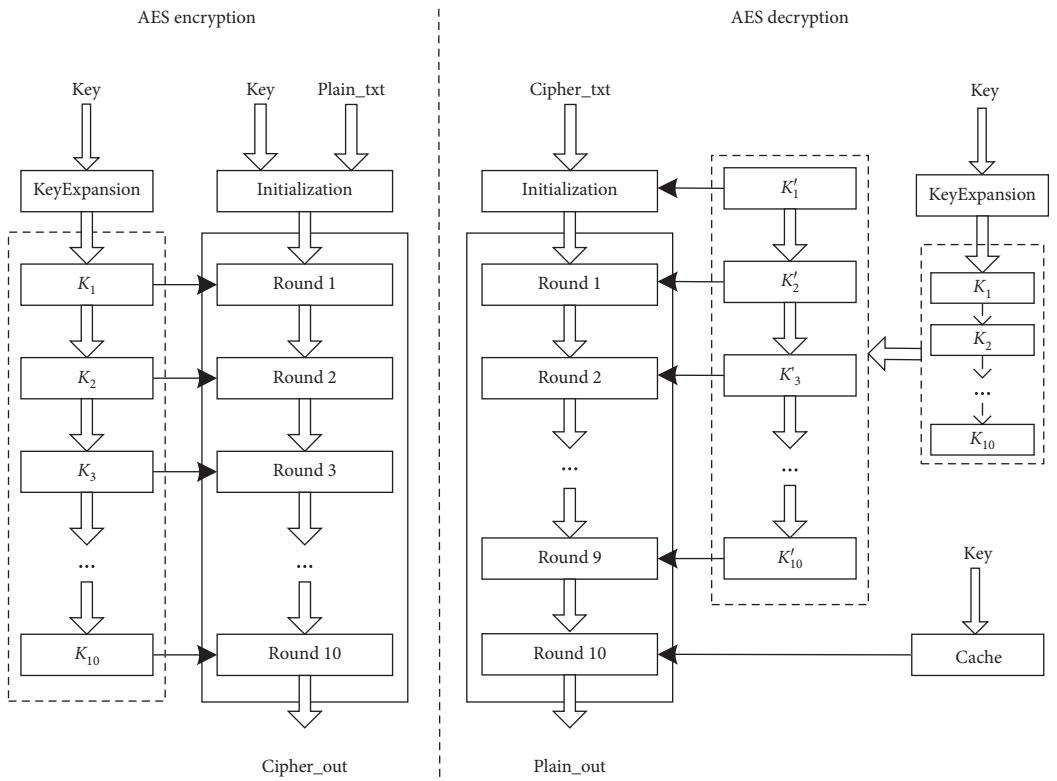

Figure 6: AES encryption and decryption structure.

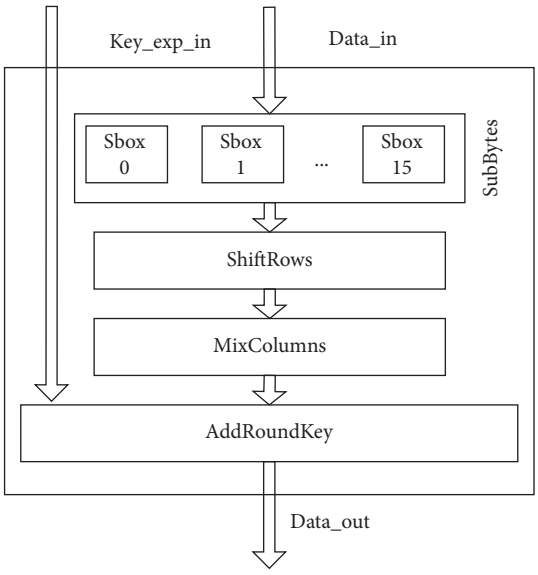

Figure 7: AES single-round encryption structure.

operation of a base point P on the curve with an integer k, that is, the addition of P for k times:

$$kP = \sum_{1}^{k} P = P + P+, \ldots, +P. \quad (4)$$

The point multiplication process includes double point and point addition operations; therefore, the optimization of double point and point addition is an important way to improve the efficiency of elliptic curve calculation.

Here, the Karatsuba–Ofman algorithm (KOA), fast modular reduction, radix-4 modular inversion, Montgomery point multiplication, and other optimization methods are combined to achieve an energy-efficient and antiattack ECC algorithm. Through the bottom-up design method, the most basic operations are realized with modular addition and subtraction, modular reduction, modular inversion, and modular multiplication, and then, the point multiplication operation is optimized by point addition and double point. Finally, the functions of elliptic curve signature, verification, encryption and decryption, and key agreement are realized. The overall structure is shown in Figure 8.

In the point multiplication calculation, the point addition and double point operations will be called many times, while the coordinate transformation is only calculated once, i.e., at the last time. Therefore, the point addition and double point operations are optimized by performance optimization, while the coordinate transformation operation is optimized by resource optimization. Second, the master state machine is used to schedule and manage the point multiplication module in order to meet the calculation requirements of different functions. Finally, the reuse of resources is realized through the coordinate transformation operation's shared modular addition and subtraction modules, and to reduce the consumption of FPGA resources, the data transmission is completed through an asynchronous FIFO method.

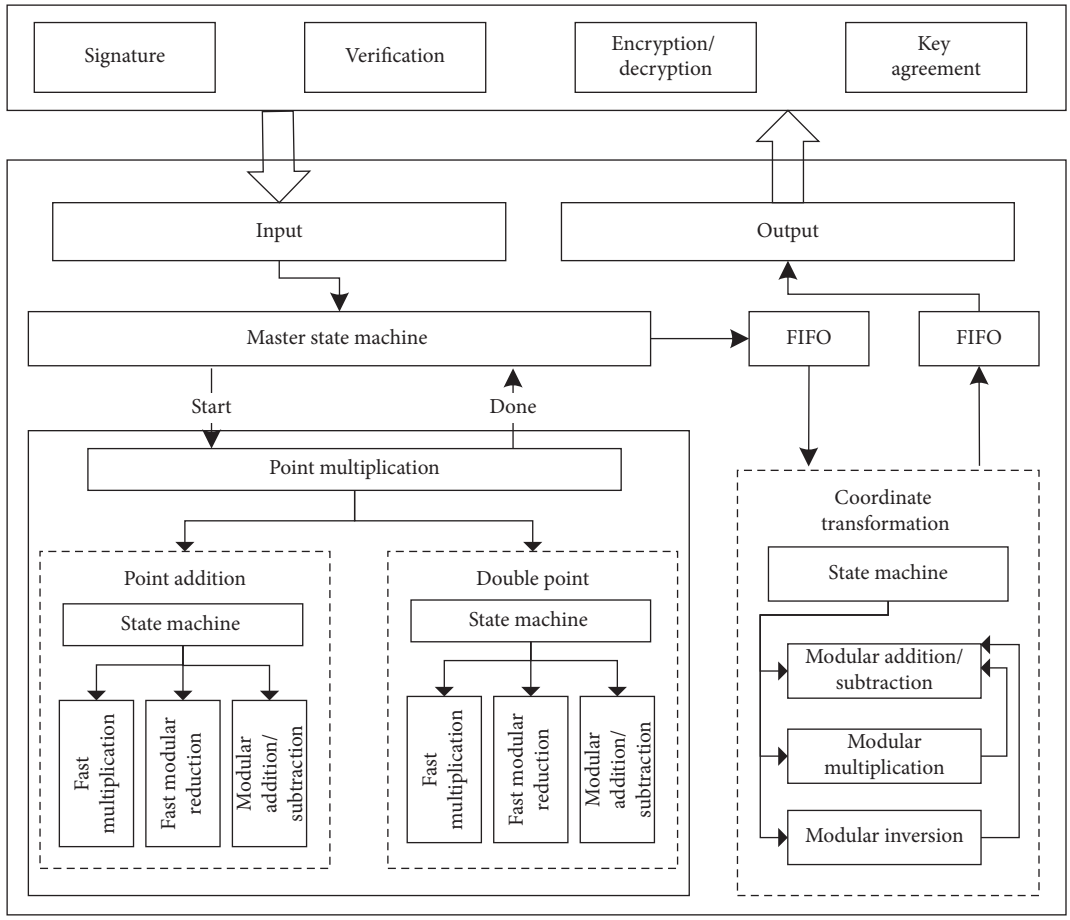

Figure 8: ECC algorithm overall architecture.

(1) KOA Fast Multiplication. The core idea of KOA [42] is "divide and conquer," i.e., a calculation approach in which a complex multiplication operation is decomposed into multiple simple multiplication operations by recursion. It is faster and more efficient than traditional calculation. For two n-bit numbers, if they are directly multiplied, the complexity is $O(n^2)$, and the complexity can be reduced to $O(n^{\log_2 3})$ by using KOA.

For the n-bit A, this can be expressed as $A = (\underbrace{\alpha_{n-1}, \alpha_{n-2}, \ldots, \alpha_{n/2}}_{A^H}, \underbrace{\alpha_{n/2-1}, \ldots, \alpha_0}_{A^L})$, where $\alpha_i \in \{0, 1\}$, $0 \le i < n$.

Then, A and B can be expressed equivalently as follows:

$$\begin{aligned} A &= A^H \times 2^{n/2} + A^L, \\ B &= B^H \times 2^{n/2} + B^L. \end{aligned} \tag{5}$$

Therefore, $C = A \times B$ can be calculated by the following formula:

$$\begin{aligned} C &= \left(A^H \times 2^{n/2} + A^L\right) \times \left(B^H \times 2^{n/2} + B^L\right) \\ &= z_2 \times 2^n + z_1 \times 2^{n/2} + z_0, \end{aligned} \tag{6}$$

where $z_2 = A^H \times B^H$, $z_1 = A^H \times B^L + A^L \times B^H$, and $z_0 = A^L \times B^L$.

The ECC parameter is 256 bits, while FPGA DSP supports multiplication operations with a maximum bitwidth of 64 bits. Then, 256 bits can be divided into 128 bits, and then, 128 bits can be divided into 64 bits. After two recursive operations, the result is obtained, as shown in Algorithm 1.

(2) Fast Modular Reduction. For fast modular reduction, the ECC special parameters can be optimized. Because of the particularity of prime number p, several addition and subtraction operations can be used to obtain the result of modular reduction. Compared with the current universal Montgomery algorithm, the fast modular reduction algorithm can save several 256-bit multiplication operations and significantly improve the performance.

When $p = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$, for a large number A: $A = A_{15} \times 2^{480} + A_{14} \times 2^{448} + \ldots + A_1 \times 2^{32} + A_0$, each A_i is a 32 bit integer; then, A can be expressed as follows: $A = (A_{15} \| A_{14} \| \ldots \| A_1 \| A_0)$.

Then, $B = A \bmod p = (T + 2S_1 + 2S_2 + S_3 + S_4 - D_1 - D_2 - D_3 - D_4) \bmod p$, where each 256 bit operand is represented as shown in Table 1.

(3) Extended Euclidean Modular Inversion. The extended Euclidean algorithm uses the toss and turns division method to obtain the modular inverse, and according to the nature of

Input: A, B, $n//n$ is the bit width
Output: C
(1) if $(n == 64)$ return $C = A \times B$;
(2) $A = A^H \times 2^{n/2} + A^L$;
(3) $B = B^H \times 2^{n/2} + B^L$;
(4) $C_1 = \text{KOA}\ (A^H, B^H, n/2)$;
(5) $C_2 = \text{KOA}\ (A^L, B^L, n/2)$;
(6) $C_3 = \text{KOA}\ (A^H + A^L, B^H + B^L, n/2)$;
(7) $C = C_1 << n + (C_3 - C_2 - C_1) << (n/2) + C_2$;

Algorithm 1: KOA multiplication.

Input: a, b, p
Output: $c = b/a \bmod p$
(1) $u = a$; $v = p$; $x_1 = b$; $x_2 = 0$;
(2) while $(v > 0)$
(3) if $(u[1:0] == 2'\text{b00})$
(4) $u = u \gg 2$; $x_1 = x_1/4 \bmod p$;
(5) else if $(v[1:0] == 2'\text{b00})$
(6) $v = v \gg 2$; $x_2 = x_2/4 \bmod p$;
(7) else if $(u[1:0] == v[1:0])$
(8) if $(u > v)$ $u = (u - v) \gg 2$;
(9) $x_1 = (x_1 - x_2)/4 \bmod p$;
(10) else $v = (v - u) \gg 2$;
(11) $x_2 = (x_2 - x_1)/4 \bmod p$;
(12) else if $(u[1:0] == 2'\text{b10})$
(13) if $((u >> 1) > v)$ $u = ((u \gg 1) - v) \gg 1$;
(14) $x_1 = (x_1/2 - x_2)/2 \bmod p$;
(15) else $u = u \gg 1$; $x_1 = x_1/2 \bmod p$;
(16) $v = (v - (u \gg 1)) \gg 1$;
(17) $x_2 = (x_2 - x_1/2)/2 \bmod p$;
(18) else if $(v[1:0] == 2'\text{b10})$
(19) if $(u > (v >> 1))$ $u = (u - (v \gg 1)) \gg 1$;
(20) $x_1 = (x_1 - x_2/2)/2 \bmod p$;
(21) $v = v \gg 1$; $x_2 = x_2/2 \bmod p$;
(22) else $v = ((v \gg 1) - u) \gg 1$;
(23) $x_2 = (x_2/2 - x_1)/2 \bmod p$;
(24) else if $(u \geq v)$
(25) $u = (u - v) \gg 1$; $x_1 = (x_1 - x_2)/2 \bmod p$;
(26) else
(27) $v = (v - u) \gg 1$; $x_2 = (x_2 - x_1)/2 \bmod p$;
(28) endwhile
(29) return $c = x_1$;

Algorithm 2: Extended Euclidean modular inversion.

Input: $k = (k_{l-1}, \ldots, k_0)$, point G
Output: $Q = kG$
(1) Initialize $R_0 = G$; $R_1 = 2G$; $i = l - 2$;
(2) while $(i \geq 0)$
(3) if $(k_i == 0)$ $R_1 = R_0 + R_1$; $R_0 = 2R_0$;
(4) else if $(k_i == 1)$ $R_0 = R_0 + R_1$; $R_1 = 2R_1$;
(5) $i = i - 1$;
(6) endwhile
(7) $Q = R_0$;

Algorithm 3: Montgomery point multiplication.

Table 1: Fast modulus reduction parameter representation.

	255–224	223–192	191–160	159–128	127–96	95–64	63–32	31–0
T	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
S_1	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	0	0	0
S_2	0	A_{15}	A_{14}	A_{13}	A_{12}	0	0	0
S_3	A_{15}	A_{14}	0	0	0	A_{10}	A_9	A_8
S_4	A_8	A_{13}	A_{15}	A_{14}	A_{13}	A_{11}	A_{10}	A_9
D_1	A_{10}	A_8	0	0	0	A_{13}	A_{12}	A_{11}
D_2	A_{11}	A_9	0	0	A_{15}	A_{14}	A_{13}	A_{12}
D_3	A_{12}	0	A_{10}	A_9	A_8	A_{15}	A_{14}	A_{13}
D_4	A_{13}	0	A_{11}	A_{10}	A_9	0	A_{15}	A_{14}

the common divisor, all divisions can be changed to addition and subtraction operations, and the division by 2 operation is completed by binary shift, which is beneficial to hardware implementation. Here, it is expressed in a quaternary system; state machine cycle control is used to optimize the extended Euclidean algorithm, and the value of $b/a \bmod p$ can be directly obtained. The specific process is shown in Algorithm 2.

In Algorithm 2, the first line is the initialization state; the second line is the loop judgment state; the lines 3–27 are various judgments and calculations; the last line is the output. Second, the calculation of u and v always ensures that the result value is less than p, and no additional processing is required. However, the addition and subtraction of $x1$, $x2$ may be greater than p or overflow, and division by 2 and division by 4 require additional judgment. The specific calculation formula is as follows:

$$\frac{x}{2} \bmod p = \begin{cases} x \gg 1, & \text{if } x[0] == 1'b0, \\ x \gg 1 + p \gg 1 + 1, & \text{if } x[0] == 1'b1, \end{cases}$$

$$\frac{x}{4} \bmod p = \begin{cases} x \gg 2, & \text{if } x[1:0] == 2'b00, \\ (x \gg 1 + p \gg 1 + 1) \gg 1, & \text{if } x[1:0] == 2'b01, \\ x \gg 2 + p \gg 1 + 1, & \text{if } x[1:0] == 2'b10, \\ (x \gg 1 + p \gg 1 + 1) \gg 1 + p \gg 1 + 1, & \text{if } x[1:0] == 2'b11. \end{cases} \tag{7}$$

(4) Point Multiplication Optimization. At present, the Montgomery point multiplication algorithm is the most efficient and widely used algorithm [43, 44]. The specific operation process is shown in Algorithm 3.

It can be seen from Algorithm 3 that regardless of the value of k_i, the point addition and double point will be calculated during each cycle; in addition, the two are independent of each other and can be executed in parallel. At the same time, due to the simultaneous calculation of point addition and double point, the power consumption information leaked during the point multiplication operation is unruly, which can effectively resist the simple power consumption attack (SPA).

In the standard projective coordinate, given point $P(X_1, Y_1, Z_1)$ and point $Q(X_2, Y_2, Z_2)$, the calculation formula of point addition and double point is as follows.

The point addition formula is as follows:

$$\begin{cases} X(P+Q) = (X_1X_2 - aZ_1Z_2)^2 - 4bZ_1Z_2(X_1Z_2 + X_2Z_1), \\ Z(P+Q) = x_G(X_1Z_2 - X_2Z_1)^2. \end{cases} \tag{8}$$

The double point formula is as follows:

$$\begin{cases} X(2P) = \left(X_1^2 - aZ_1^2\right)^2 - 8bX_1Z_1^3, \\ Z(2P) = 4Z_1\left(X_1^3 + aX_1Z_1^2 + bZ_1^3\right). \end{cases} \tag{9}$$

Finally, the result is converted to an affine coordinate as follows:

$$x_1 = \frac{X_1}{Z_1}, \tag{10}$$

$$x_2 = \frac{X_2}{Z_2}, \tag{11}$$

$$y_1 = \frac{2b + (a + x_Gx_1)(x_G + x_1) - x_2(x_G - x_1)^2}{2y_G}. \tag{12}$$

Then, (x_1, y_1) is the result. Among them, (x_G, y_G) is the coordinate of the base point G.

It can be seen that, under standard projection coordinates, the projection point and the affine point will be mapped one by one. The affine coordinates will be

transformed into the projection coordinates at the beginning and will be mapped back to the affine coordinates when the operation is finished. Therefore, in the entire calculation process, only one modular inversion operation is used at the last time, and there is no modular inversion participation in the intermediate iteration process.

Finally, in order to further optimize the calculation efficiency of point addition and double point, the data stream is deeply optimized to complete the calculation in the shortest time. Fast modular multiplication consists of two modules, KOA multiplication and fast modular reduction, and the results can be calculated within one clock. Therefore, the part of the calculation process of point addition and double point is adjusted, and the KOA multiplication and fast modular reduction modules are alternately called to give full play to the calculation efficiency. After optimization, the point addition and double point calculation can finally be completed within 12 clocks, which have a very high efficiency.

3.4. Key Generation and Management. After the initial working key is obtained through the key management center, a pseudorandom number generator is used to generate the first group of keys to encrypt the first batch of packets. Then, the first batch of packets is used to generate the second group of keys, the second batch of packets is used to generate the third group of keys, and so on in order to generate all encryption keys, as shown in Figure 9.

3.4.1. Random Generation of Keys. A pseudorandom number generator [45] is widely used in information encryption. The selection of pseudorandom numbers starts from random seeds. Therefore, in order to ensure that the pseudorandom numbers obtained each time are sufficiently "random," the selection of random seeds is very important. If the random seeds are the same, the random numbers generated by the same random number generator will also be the same.

The most common method of generating pseudorandom numbers is to utilize a feedback shift register, which consists of two parts: the shift register and the feedback function. When the feedback function is linear, the feedback shift register is a linear feedback shift register (LFSR), as shown in Figure 10.

In the LFSR structure, f_n is the feedback coefficient, 1 means connection, and 0 means no connection.

Obviously, the output sequence of the LFSR is periodic, and an n-level LFSR provides up to $2n-1$ states (excluding all 0 states). According to the different feedback modes, the characteristic polynomial of the LFSR can be defined as follows:

$$p(x) = \sum_{i=0}^{n} f_i x^i = f_n x^n + f_{n-1} x^{n-1} + \ldots + f_1 x + 1. \quad (13)$$

In this paper, a 128 bit random number *randum* is defined, and the initial working key is used as a random seed to randomly generate the first group of keys. The polynomial used is $p(2) = 2^{128} + 2^{126} + 2^{101} + 2^{99} + 1$.

Similarly, the polynomial $p(2) = 2^{32} + 2^{26} + 2^{23} + 2^{22} + 2^{16} + 2^{12} + 2^{11} + 2^{10} + 2^8 + 2^7 + 2^5 + 2^4 + 2^2 + 2 + 1$ is used to complete the calculation of CRC32. Its initialization parameter is 0xFFFFFFFF, which uses big-endian alignment, and the end is not enough to be filled with 0x00. To meet the calculation requirements, the parallel method of the use of a lookup table is employed to realize the CRC32 calculation, which supports the direct calculation of 64-bit and 128-bit data and meets the calculation requirements of a 1G/10G network.

3.4.2. Key Storage and Update. During encryption and decryption, the key involved in the operation comes from the hash value of the last packet. To ensure the key synchronization of encryption and decryption, it is necessary to maintain a key storage (KS) module on the encryption and decryption channel. The depth of KS must be greater than the maximum number of hash iterations to ensure that the key is generated in advance before the next round of use. Here, dual-port RAM is used to store keys with a depth of 256 bits and a width of 128 bits, as shown in Figure 11.

In Figure 11, the initial key of KS is generated by the random number generator and, then, is continuously updated by the hash value of the packet. In the encryption direction, every time a packet that needs to be encrypted is received, the HKey of the corresponding entry is first taken from KS by using the PID as the key. Then, the hash value of the first 64 bytes of the encrypted packet updates KS. For the KS in the decryption direction, first, the HKey of the corresponding PID is taken from KS, and the encrypted packet is decrypted; at the same time, the hash value of the first 64 bytes of the current packet is fed back to the KS.

For the working key, it is not only used to initialize the random number generator but also the key for signature and verification. Moreover, the working key needs to be kept during the working period of the mimic encryption box, and there should be a backup and recovery mechanism. Therefore, the working key requires a higher security storage and update strategy. Here, flash memory is used to store the working key, and only the FPGA has read and write permissions. In addition, the key management center establishes a secure tunnel with the mimic encryption box and regularly updates the working key online.

3.5. Data Packet Processing

3.5.1. Rule Filtering. To monitor network data and prevent illegal access, for rule filtering, five tuples are used to form an access control list (ACL), including the protocol type, source and destination IP address, and source and destination port. At the same time, FPGA reconfigurability is used to form a pipeline for processing each step in order to meet the demand of high-speed data sending and receiving. The specific structure is shown in Figure 12.

Figure 12 shows that rule filtering is mainly composed of 4 functional modules: Parse, Key, Match, and Action.

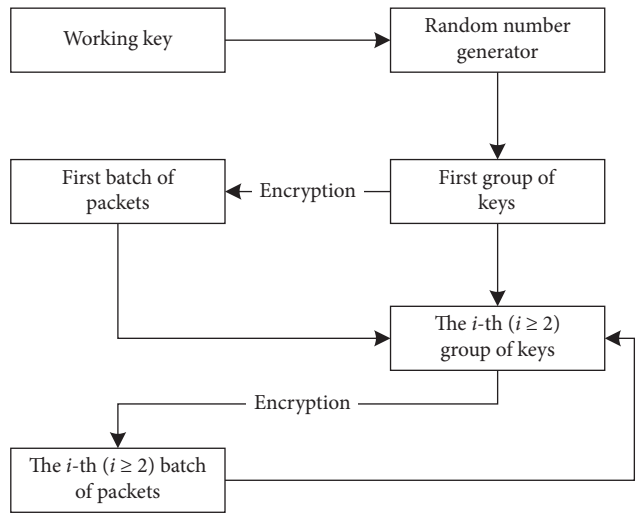

Figure 9: Key generation process.

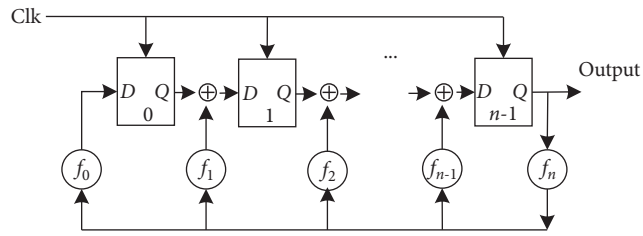

Figure 10: LFSR structure.

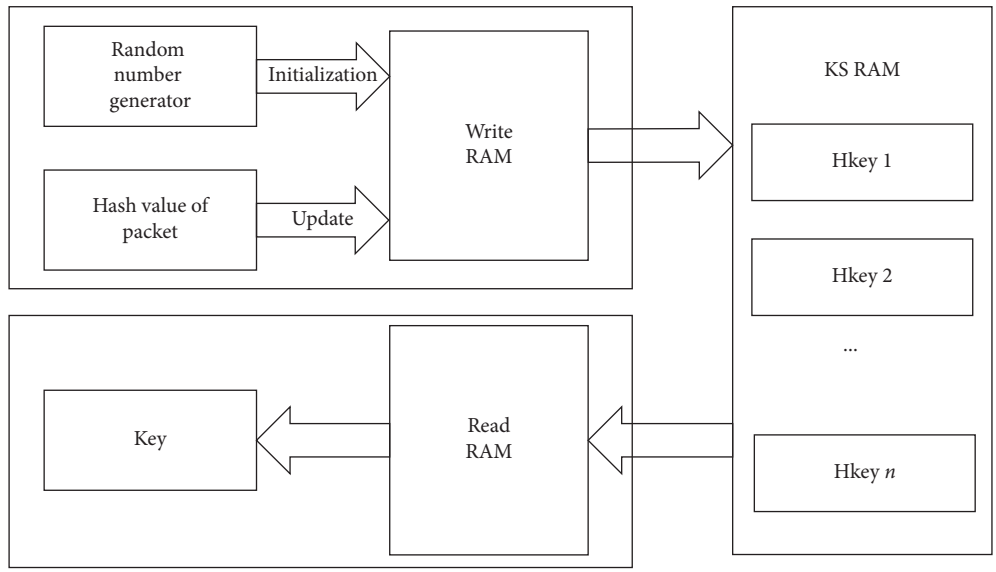

Figure 11: Storage and update for the key.

Among them, Parse represents the analysis of packet content; Key represents the extraction of keywords; Match represents matching, in which the matching is conducted by the rules of the ACL; Action represents action execution, indicating the processing of packets; and ACL represents the access control list and consists of five tuples. While the current packet is being parsed, FIFO is used for buffering and output according to the matching result.

Here, mainly IPv4 packets are analyzed. Multiple values for the Key module, ACL, and Action module are defined simultaneously, i.e., Key = $\{key_1, key_2, ..., key_{n1}\}$, ACL = $\{acl_1, acl_2, ..., acl_{n2}\}$, and Action = $\{action_1, action_2, ..., action_{n3}\}$. In

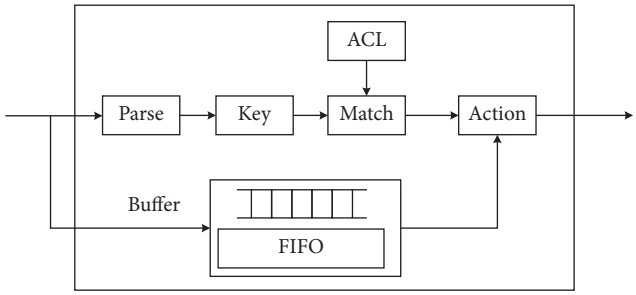

Figure 12: Rule filtering of packets.

this way, the packets are stripped layer by layer and extracted to a number of keywords, and then, according to the rules in a lookup table, the plaintext path, ciphertext path, discarding, encapsulation, decapsulation, and other data packets' processing is achieved.

3.5.2. DDR Cache Storage. To effectively deal with the impact of network blocking, prevent the mass dropping of packets, and ensure the smooth transmission of packets in each link, the packets are cached in memory. The FPGA provides the memory operation interface, but for each read and write, multiple signals need to be judged, which is quite tedious. To simplify the memory operation and improve the read and write efficiency, the FPGA memory interface was further encapsulated and optimized, as shown in Figure 13.

In the user FPGA logic module, four sets of FIFO are defined and are used to store the data and address for reading and writing memory. Then, the memory read-write control signals, such as app_addr, app_cmd, and app_en, are associated with FIFO read-write signals. In this way, as long as there are data and addresses in the four FIFOs, the state machine control will automatically read, arbitrate, and complete the memory read and write operations. Second, the memory read and write logic is independent, and the two do not affect each other, thereby improving the efficiency of the memory operations.

In addition, because the memory interface width is 256 bits and the MAC layer transmits 8/64 bits, bitwidth conversion is required for data written to memory. Similarly, data read from memory is converted from 256 bits to 8/64 bits. Finally, due to the need to complete reading and writing memory data according to the address and form a one-to-one mapping between data and address, the size of the address list has a direct impact on the space for reading and writing memory data. To store more data, the depth of the FIFO read and write memory address is set to 8192. In this way, multiple data packets can be cached, and the impact of network blocking can be effectively alleviated.

3.6. Security Mechanism

3.6.1. Autonegotiation Network Transformation. IP address hopping, port number hopping, protocol camouflage, channel transformation, and other technologies can hide the service mark and confuse the attacker, achieving covert communication. To further increase the attack difficulty and cost of the attacker, an autonegotiation transformation network is established. Through automatic negotiation, the communication parties form a security policy combination mechanism by frequently changing their IP addresses, ports, protocols, and channels in order to improve their defense capability. The specific process is shown in Figure 14.

As seen from Figure 14, the transformation process of the self-transforming network is as follows:

(1) The initiator will combine the transformed IP address, port, protocol, and channel to generate information *A*. The public key of the responder is used to encrypt and sign the data, generate the encrypted data *EncA* and the signed data *SigA*, and send it to the responder.

(2) After receiving the *EncA* and *SigA* from the initiator, the responder uses the private key to decrypt the information *A*, signs to generate the signature data *SigATmp*, and verifies the signature data *SigA* and *SigATmp*. If the verification is correct, the process continues to the next step.

(3) Similarly, the responder will transform the IP address, port, protocol, and channel combination to generate information *B*, encrypt *A* and *B* with the public key of initiator to generate encrypted data *EncAB*, and generate signature data *SigB* for *B*. Then, the responder will send *EncAB* and *SigB* to the initiator.

(4) The initiator receives *EncAB* and *SigB* from the responder and decrypts *A* and *B* of *EncAB* with the private key. The initiator, then, signs *B* to generate signature data *SigBTmp* and verifies the signature data *SigB* and *SigBTmp*. If the verification is correct, the process continues to the next step.

(5) The communication parties will update the data packet filtering rules by information *A* and *B* and use the new IP address, port, protocol, and channel for data encryption transmission.

The communication parties themselves can negotiate the time interval, which can be configured by ARM, or the initiator can define a timer to automatically negotiate according to certain time rules. Alternatively, the key management center may notify both parties to complete the network transformation.

3.6.2. Feedback Reconstruction Mechanism. The dynamic configuration of system parameters is mainly realized by the ARM, and the data are written into the inRAM of FPGA through memory address mapping. Moreover, the FPGA writes its state information, including controlling responses, number of incoming and outgoing packages, packet loss rate, decryption failure, and checksum error, into outRAM and sends it to the ARM. This structure is shown in Figure 15.

In this way, the collected information is analyzed by the ARM, and if an abnormality is found, the feedback

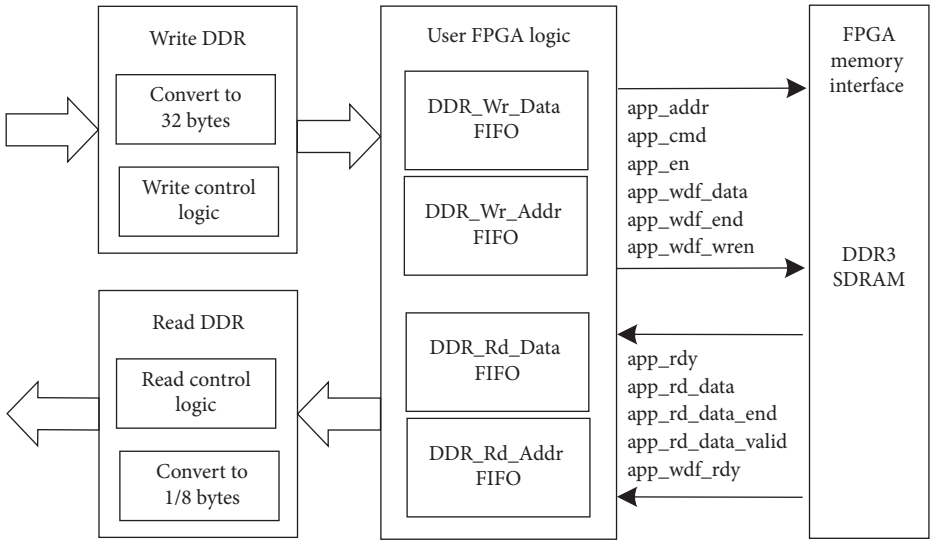

Figure 13: Memory read and write control optimization.

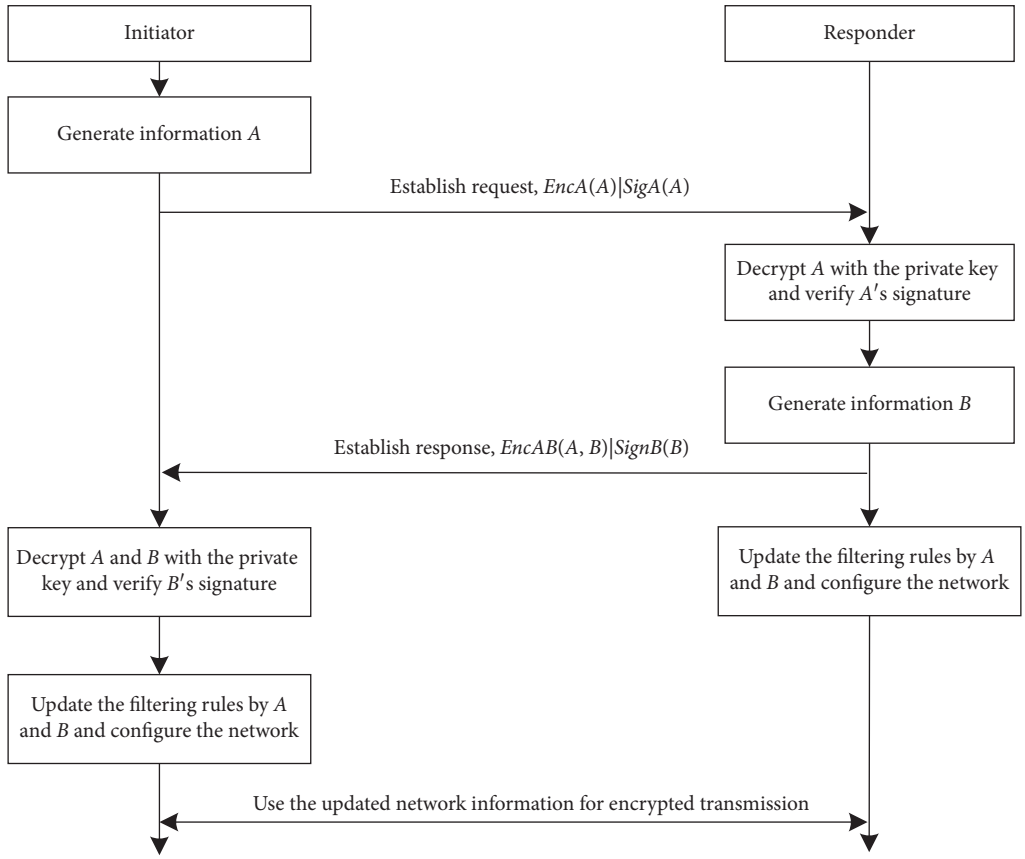

Figure 14: Self-transforming network negotiation process.

reconstruction mechanism is activated. First, a decision tree is built on the ARM side. In this decision tree, each branch node represents the choice between multiple alternatives, and each leaf node represents a decision. Then, whenever an abnormality is detected, the depth-first algorithm is used to traverse down from the root node, and if the judgment conditions of the current node are met, the execution is performed sequentially. Third, a heartbeat mechanism is added to the judgment condition, and the ARM sends a heartbeat packet to confirm whether the current network is under attack. If no response is received for a long time, the key management center will be notified and try to select another network for communication. Finally, according to the results of the decision tree judgment, the FPGA is

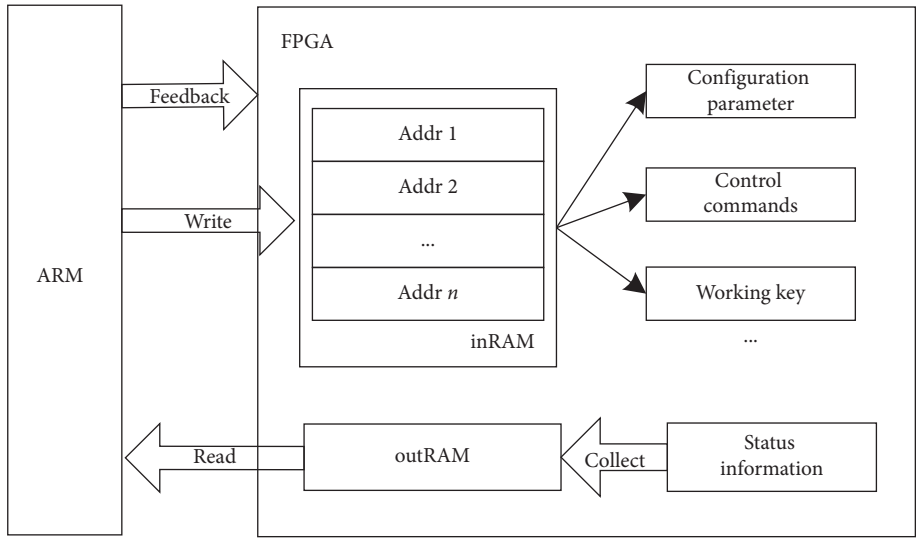

Figure 15: FPGA management structure.

"cleaned" through various operations, such as resetting all data paths and memory, closing the current network interface, and changing the network configuration.

For example, when the FPGA receives a large number of packets that do not comply with the rules in a short timeframe, it can be judged that the current network has been illegally attacked. Then, the current network interface can be closed, and the key management center can be notified to reselect a new network with which to connect. For another example, when no response from the other party is received for a long time, another network port is selected and a heartbeat packet is sent to verify the working status of the other party. In addition, when an attack causes the FPGA to work abnormally, the ARM can first perform a reset operation. If it still does not work normally, it can directly load other bitstreams to reconstruct the FPGA.

4. Experimental Results and Analysis

4.1. System Implementation. The server, FPGA, and switch used in this paper are shown in Table 2. The development environment is Vivado v2019.2 (64 bit).

The FPGA is composed of dual-core ARM processors and a Kintex-7 programmable chip, which can communicate by memory mapping. Its structure is shown in Figure 16. Among the FPGA's components, the ARM side has a 1G Ethernet port; the FPGA side has two 10G Ethernet ports and two 1G Ethernet ports; the FPGA is connected with DDR3 memory and flash. By writing bitstream into flash, the automatic loading and reconstruction of FPGA can be completed.

4.1.1. Implementation of Each Module. The following are implemented on the FPGA: 10G Ethernet interfaces; 1G Ethernet interfaces; packet parsing and encapsulation modules; and key management and update modules. The realization of each functional module is shown in Table 3.

To prevent packet overflow, each module is interconnected through an asynchronous FIFO and a set *prog_full* flag. If the FIFO is about to be full, the MAC layer frame flow control function will be triggered immediately and the data transmission will be suspended. In addition, 1G networks and 10G networks have the same processing flow, except for the fact that they work at different frequencies and the data width is 1 byte and 8 bytes, respectively. Therefore, bitwidth conversion is needed before packet encryption and decryption can be performed.

Second, the mimic encryption box in this paper encapsulates the AES128 and the 3DES algorithms to realize encryption and decryption and encapsulates the SHA1 and the SHA256 pipeline algorithms to form the hash algorithm. The implementation of each algorithm is shown in Table 4.

Finally, for the ECC algorithm, the specific situation of clock frequency, resource consumption, and calculation cycle of each module is shown in Table 5.

Table 5 shows that when the point multiplication frequency is 25 MHz, the calculation can be completed after 3064 clocks, which is a very high computational speed.

When the FPGA is configured as a dual 1G network, AES128 and SHA1 are used to complete the encryption. The occupancy ratio of LUTs is 60.17%, REGs is 32.15%, and DSP is 48%. When configured as a dual 10G network, 3DES and SHA256 are used to complete the encryption. The occupancy ratio of LUTs is 73.88%, REGs is 37.63%, and DSP is 48%. Therefore, the design requirements of the mimicry encryption box are fully met.

4.2. Performance Analysis

4.2.1. Encryption and Decryption Throughput. The throughput calculation formula is as follows:

$$T = \frac{B \times f_{\max} \times N}{d}, \tag{14}$$

Table 2: Configuration information of each component.

Component	Name	Configuration information
Server	IBM X3650 M3	CPU type: X5650 2.66 GHz; memory: 24 GB
FPGA	Xilinx XC7Z035	ARM: 800 MHz; memory: 1 GB; flash: 256 Mb; 2 SFP+ 10G Ethernet ports; 3 1G Ethernet ports; logic cells: 270K; number of DSP: 900
Switch	H3C 24-port 10G switch	24 1/10G SFP + Ethernet ports; 4 10/100/1000 M electrical ports

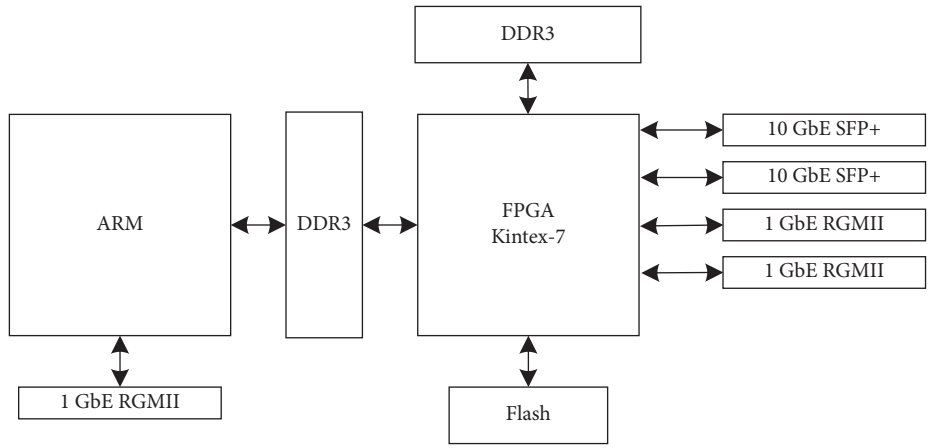

Figure 16: Hardware structure of the FPGA.

Table 3: Implementation of FPGA each functional module.

Functional module	Description	Frequency (MHz)	LUTs	REGs
ARM	ARM communication module	100	6487	8296
SFP_10GE_MAC	10G Ethernet port	156.25	4816	5457
Tri_Mode_Ethernet_MAC	1G Ethernet port	125	637	879
Memory_WR	Memory and read-write interfaces	666.67	9321	7549
Packet_Parse	Packet parsing	156.25	900	2336
Packet_Filter	Packet filtering	156.25	1026	2673
Packet_Package	Packet encapsulation	156.25	274	476
Parm_Mng	Parameter and key management	100	979	1832
Key_Update	Key update	156.25	517	1016

Table 4: Implementation of each encryption algorithm and hash algorithm.

Main algorithm	Algorithm structure	Highest frequency (MHz)	LUTs	REGs
AES128	Serial, 23 clocks	350	1060	402
3DES	48-stage pipeline	410	4436	5661
SHA1	80-stage pipeline	400	12336	23860
SHA256	64-stage pipeline	280	21472	24374
CRC32_64	64-bit lookup table	450	266	100
CRC32_128	128-bit lookup table	320	445	164
LFSR	Lookup tables, state machine	310	124	186

where T is the throughput, B is the data block size, $f_{\max}$ is the maximum clock frequency of each scheme, N is the pipeline stages, and d is the calculation delay.

To ensure the performance of encryption and decryption, AES128, 3DES, and LFSR all work at 200 MHz, while due to the use of pipeline technology, SHA1 and SHA256 work at 125 MHz or 156.25 MHz. In addition, for a 10G network, multiple encryption algorithm modules are designed to work in parallel in order to further improve the efficiency of execution. The throughput of each algorithm is calculated by formula (14), and the result is shown in Figure 17.

In Figure 17, in order to meet the demand of 10G encryption, 10 AES modules are designed to execute in parallel

Table 5: Implementation of each ECC module.

Module	Frequency (MHz)	LUTs	REGs	DSP	Calculation period
Point addition	25	8625	4380	144	12
Double point	25	9685	3620	144	12
Point multiplication control	25	1489	7217	0	3064
Coordinate transformation	25	17820	4401	144	225
Master state machine	50	252	1294	0	—

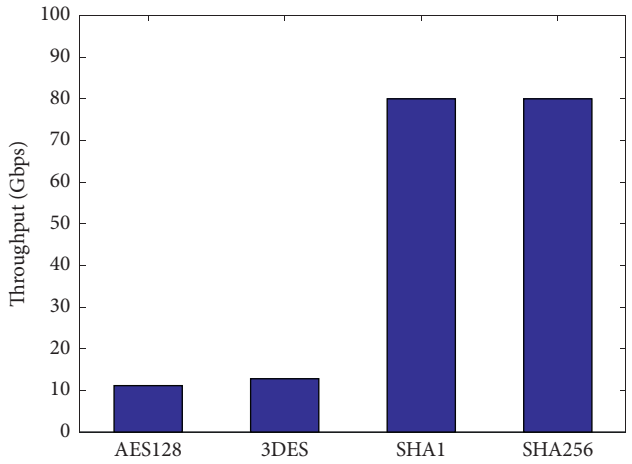

Figure 17: Throughput of each encryption algorithm.

and can work in four modes: ECB, CBC, CFB, and OFB. However, 3DES only works in the ECB mode. As seen from Figure 17, the algorithm throughputs achieved in this paper are all above 10 Gbps and can, thereby, fully meet the computing requirements of encryption and decryption of a 1G/10G network.

In the process of ECC signature/verification, encryption, and decryption of the key exchange, point multiplication will be called many times, and hash operation and coordinate transformation are also needed. Based on a frequency of 25 MHz, the corresponding speed of the ECC applications is shown in Table 6.

Table 6 reveals that the ECC can complete signature/verification, key encryption, and decryption, at least, 3000 times per second, which is a considerable execution speed.

4.2.2. Network Performance. Under 1G/10G networks, the maximum transmission unit is configured as 1495 bytes, and files of different sizes are encrypted and transmitted to the opposite end. After the peer receives the file, it decrypts the file. The CPU and mimic encryption box were used for testing. As the file size changes, the communication time between the two is shown in Figure 18.

Figure 18 shows that the processing time of the mimic encryption box encryption and decryption is significantly lower than that of the CPU. This is mainly because the mimic encryption box completes encryption and decryption while transmitting data. The CPU needs to encrypt data before transmitting and to decrypt the data after receiving it, which is time consuming. Especially under the 10G network, the encryption time occupies more than 88% of the total time;

Table 6: Calculation speed of the ECC applications.

Application	Clock number	Speed (p/s)
Signature	4068	6145
Verification	7629	3276
Encryption	7439	3360
Decryption	3908	6397

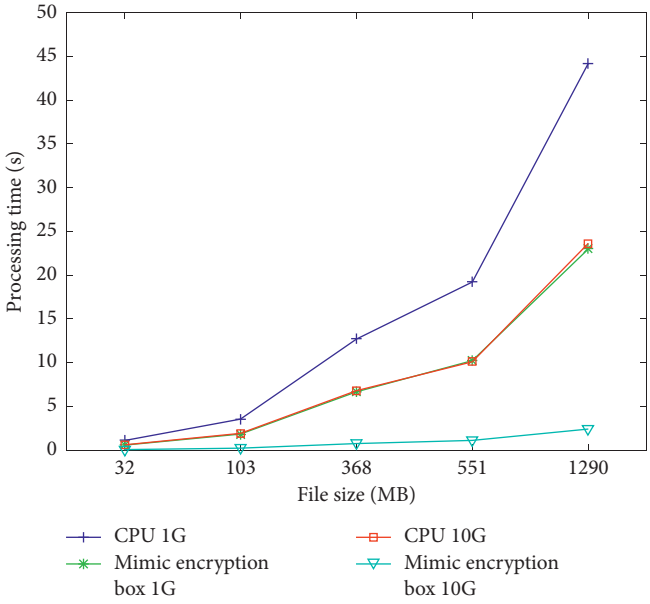

Figure 18: Comparison of the file encryption and decryption transmission performance between the CPU and the mimic encryption box.

therefore, the encryption speed of the CPU becomes the bottleneck.

In the 1G/10G networks, 100,000 packets are sent each time. As the packet length increases, the processing time of the CPU and the mimic encryption box changes, as shown in Figure 19.

Figure 19 shows that the mimic encryption box takes significantly less time to process encrypted packets than the CPU does. This is mainly because the FPGA omits system scheduling and speeds up network transmission and encrypted data processing.

In the 1G network, one byte is transmitted per clock at 125 MHz. However, AES128 needs to input 16 bytes each time to participate in the operation, and generates the result after 23 clocks. For continuous data streams, AES working at 200 MHz can meet the computing needs of the 1G network. At this time, the DDR cache is mainly used to avoid data loss

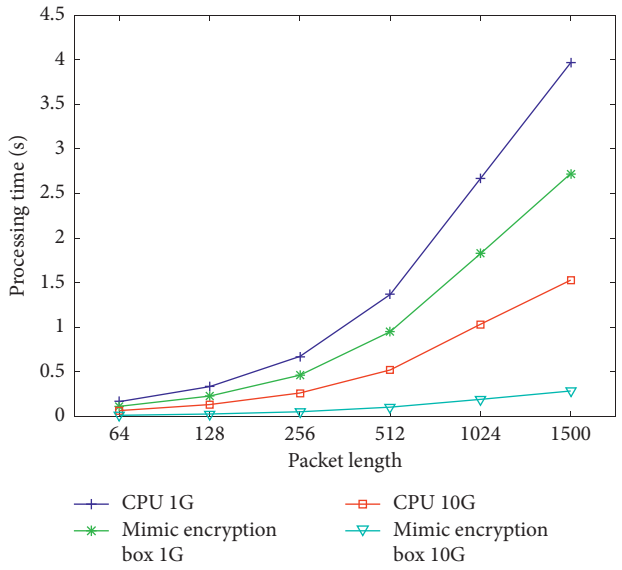

FIGURE 19: Comparison of the processing time of a large number of packets of different lengths between the CPU and the mimic encryption box.

when switching network interfaces. In the 10G network, 8 bytes are transmitted per clock at 156.25 MHz, and 10 AESs are required to work at 200 MHz under the continuous data stream. To ensure that the packets are first in, first out, the packets need to be distribution and collected. Also, it is necessary to wait for 10 groups of packets to be encrypted before sending out. As a result, the data throughput of the network side is greater than that of the encryption side. In addition, once the FIFO of the cached packet is about to be full, the flow control mechanism will be triggered to inform the other party to stop sending data. Obviously, if the DDR cache is not used, flow control frames will be sent frequently, resulting in a decrease in data throughput. After testing, the maximum throughput of the mimic encryption box under continuous data flow is as shown in Table 7.

It can be seen from Table 7 that using DDR cache improves data throughput to a certain extent. Moreover, the pause time of the flow control can be increased by the DDR cache, which can reduce the frequent transmission of flow control frames.

4.3. Security Analysis

4.3.1. Mimic Security Analysis. If the mimic encryption box is represented by the symbol Ω, it can be described by a 6-tuple as follows: $\Omega = \{\mathrm{Ec}, \mathrm{Key}, \mathrm{IP}, \mathrm{Port}, \mathrm{NPT}, \mathrm{NI}\}$, where Ec represents the encryption algorithm, Key represents the key, IP represents the IP address, Port represents the port number, NPT represents the network protocol type, and NI represents the network interface. The multiple stages of the system have multiple different encryption combination schemes. If a state vector $\Omega(t) = \{\mathrm{Ec}(t), \mathrm{Key}(t), \mathrm{IP}(t), \mathrm{Port}(t), \mathrm{NPT}(t), \mathrm{NI}(t)\}$ is used to represent a state at a certain moment, a set of reachable finite states can be used to represent all the different states of the system; that is,

$$\Omega = \begin{Bmatrix} \Omega(t_1) \\ \Omega(t_2) \\ \cdots \\ \Omega(t_l) \end{Bmatrix} = \begin{Bmatrix} \mathrm{Ec}(t_1), \mathrm{Key}(t_1), \mathrm{IP}(t_1), \mathrm{Port}(t_1), \mathrm{NPT}(t_1), \mathrm{NI}(t_1) \\ \mathrm{Ec}(t_2), \mathrm{Key}(t_2), \mathrm{IP}(t_2), \mathrm{Port}(t_2), \mathrm{NPT}(t_2), \mathrm{NI}(t_2) \\ \cdots \\ \mathrm{Ec}(t_l), \mathrm{Key}(t_l), \mathrm{IP}(t_l), \mathrm{Port}(t_l), \mathrm{NPT}(t_l), \mathrm{NI}(t_l) \end{Bmatrix}. \quad (15)$$

Among them, the components of the vector represent the changes of the system encryption algorithm, key, IP address, port number, protocol type, and network interface channel.

The traditional encryption system's component vectors do not change during operation; therefore, $\Omega(t_1) = \Omega(t_2) = \ldots = \Omega(t_l)$; that is, the traditional encryption system is static and deterministic. The characteristics of the mimic encryption box are its dynamics, diversity, and randomness; that is, at the time ti, the state of the system changes, so $\Omega(t_1) \neq \Omega(t_2) \neq \ldots \neq \Omega(t_l)$.

Ω can be described by information entropy: $H(x) = -\sum_{j=1}^{l} p_j \log(p_j)$, where p_j represents the probability of occurrence of each component vector $\Omega(t_j)$. Therefore, the external uncertainty of Ω can be transformed into the size of the information entropy; that is, the maximum information entropy can be determined as follows: $H_{\max}(X) = \max\left\{-\sum_{j=1}^{l} p_j \log(p_j)\right\}$.

Obviously, when the probability of occurrence of $l\Omega(t_j)$ is the same and equal to $1/l$, $H(X)$ reaches the maximum value, namely, $H_{\max}(X) = \log l$. Therefore, the greater the change state of Ω is, the greater the information entropy and the greater the external uncertainty are. As a result, this paper uses multiple encryption algorithms and multiple groups of keys, combined with IP address hopping, port number hopping, protocol camouflage, and network interface selection, to jointly realize a number of different element changes and combinations, which have a high degree of uncertainty.

4.3.2. Encryption Security Analysis. The mimic encryption box encrypts the plaintext load of the TCP layer and changes the IP address, port, and protocol so that the attacker cannot obtain user-related information, increasing the difficulty of the attack. The selected AES and 3DES encryption

Table 7: Maximum throughput of the mimic encryption box.

	1G FIFO cache	1G DDR cache	10G FIFO cache	10G DDR cache
Continuous data stream	110 MB/s		812 MB/s	855 MB/s

algorithms have high security and can provide high-quality data protection. At the same time, the encryption algorithm provides four working modes: ECB, CBC, CFB, and OFB. Among them, under CBC, data blocks are interrelated during encryption and are not easy to be actively attacked.

Second, the 256-bit ECC algorithm is used to complete the negotiation between the communication parties. The ECC algorithm is based on the intractable problem of discrete logarithms, which has higher security and resistance to attacks. Although, according to the published $E(F_p(a, b))$, base point G and order n, 2G, 3G, ..., nG can be calculated, and $n\mathrm{G} = O$. In addition, when a large number k is given, $P = kG$ can be easily calculated. However, given P and G, it is very difficult to reversely infer k.

Finally, the hash operation is irreversible, and the hash value of the packet is used as the key of the next round in order to prevent the attacker from pushing back the key according to the content. If the output value of the hash is uniformly distributed and the bits of message digest are m bits, then there are $n = 2^m$ possible outputs. If k $(k \le n)$ random inputs are selected, the probability of, at least, one collision is as follows:

$$\begin{aligned} P(n,k) &= 1 - \frac{n!}{(n-k)! \times n^k} \\ &= 1 - \left[\left(1 - \frac{1}{n}\right)\left(1 - \frac{2}{n}\right), ..., \left(1 - \frac{k-1}{n}\right)\right] \\ &= 1 - \prod_{i=1}^{k-1}\left(1 - \frac{i}{n}\right) \approx 1 - \prod_{i=1}^{k-1} e^{-i/n} = 1 - e^{-k(k-1)/2n}. \end{aligned} \tag{16}$$

To make $P(n,k) > 0.5$, that is, to achieve $1/2 = 1 - e^{-k(k-1)/2n}$, we have the following: $\ln 2 \approx k^2/2n$; then, $k \approx \sqrt{n}$.

According to the abovementioned calculation, if the hash function has m-bit output digests, then the probability of a collision occurring with only $k = 2^{m/2}$ attempts is, at least, 50%. In this way, any change in the input information will result in a significant change in the hash result, thereby ensuring that the key is greatly different. Table 8 shows the collision threshold of the hash function.

4.3.3. Antiattack Analysis. A test environment is set up through network cameras, mimic encryption boxes, and switches, and then, a simulation of attackers launching attacks on intermediate switches is conducted in order to verify the security of the mimic encryption boxes, as shown in Figure 20.

Table 8: Hash function collision threshold.

Hash function	Collision threshold
SHA1	$2\text{^}80 \approx 1.2 \times 10\text{^}24$
SHA256	$2\text{^}128 \approx 3.4 \times 10\text{^}38$

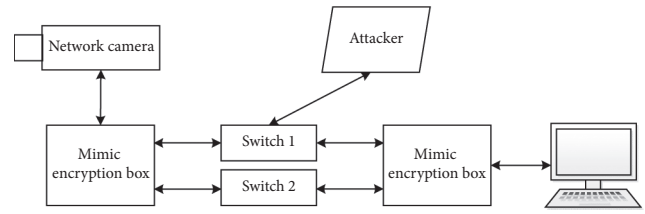

Figure 20: Security test environment of the mimic encryption box.

(1) Network Sniffing. The mimic encryption box uses dynamic network information hopping and pseudorandom encryption to make the system appear in a changing state. The IP address and port hopping mechanism essentially increase the difficulty of scanning by randomly and unpredictably migrating service entries. To obtain accurate target information, the attacker will increase the number and frequency of scanning and detection, which will significantly increase the cost of the attack. Assuming that the time of an attack is t, the number of IP addresses that can be hopped is n, the number of ports is m, and then, the time required for the attacker to successfully break a service is $T = t \times (1 + \sum_{k=1}^{n\times m-1} k \times C_{n\times m-1}^{k} / (C_{n\times m}^{k} \times C_{n\times m-k}^{1}))$, which is simplified as $T = t \times (1 + (n \times m - 1)/2)$. Then, taking 251 IP addresses 192.168.0.3–192.168.0.253 as an example, 64510 ports in 1025–65534 are detected. These IP addresses and ports are combined to form 251×64510 service entries. If the attacker scans every 5 milliseconds, it will take, at least, 40480 seconds. Therefore, the time of IP address and port number hopping can be set to 3600 seconds to reduce the scanning success rate. Second, if the scanning detection frequency is too high, the illegal data can be found quickly based on the statistical information of the filtering rules. In addition, even if the detection is successful, if the mimic encryption box subsequently switches the network, such as switching from switch 1 to switch 2, the attacker needs to sniff again.

(2) Tampering Attack. For tampered encrypted packets, if the attacker tampered with the PID, it will cause the decryption to fail. However, because it is decrypted first and, then, the CRC32 is checked, after decryption fails, the CRC32 cannot be passed, so the packet is discarded. If the first 64 bytes of the packet are tampered with, the key update and decryption will fail, and the packet will be discarded. If the data behind the 64 bytes of the packet are tampered with, the decryption will fail and the packet will be discarded. Obviously, as long as the attacker tampered with any byte of the encrypted packet, it will cause the decryption to fail and cause the packet to be discarded. At this point, the ARM can monitor the behavior and quantity of packet discards in real time. If a large number of packets are found to be discarded, it will

consider that the system has been attacked by an attacker, and then, communication is stopped or the network channel is changed.

(3) Replay Attack. When an attacker intercepts a packet and implements a replay attack, the system will follow the normal process. However, because PIDs increase in sequence, through the collection of statistics on the same PID packets, if it is found that the number of packets of a certain PID is significantly higher than other packets, it can be judged that it has received a replay attack. Second, because the TCP/IP protocol is used to transmit data, the repeated sending of the same packet will cause the system to fail to receive the packet with the corresponding sequence number. At this time, the system will not receive valid packets and will send new requests repeatedly. Finally, the sequence number field of the TCP header is encrypted; therefore, it is difficult for an attacker to forge the correct sequence number. In this way, through user monitoring, replay attacks in this situation can be found.

(4) Ciphertext-Only Attack. The mimic encryption box can effectively resist key exhaustive attacks, ciphertext-only attacks, and differential attacks. With the constantly updated key and the dynamically changing encryption algorithm, that is, the "one packet, one key," it effectively prevents attackers from using brute force attacks and ciphertext to reverse the key, ensuring that the attacker cannot decrypt all data normally. At the same time, the hash value of the first 64 bytes of the encrypted packet is used as the key for the next round. Since the hash value is basically irregular symbol data, it is difficult for an attacker to perform inference analysis. The mimic encryption box effectively resists the method of using differential attacks to decipher the ciphertext and prevents unauthorized leakage and undetected modification of data.

(5) DDos Attack. A DDos attack will cause the mimic encryption box to have a key update failure and a decryption failure; the attack will, then, affect normal system operation. If a large number of illegal interference packets are detected in a short period of time, the feedback reconstruction mechanism can be used to select other networks for communication. For example, when the attacker attacks the network where switch 1 is located, at this time, the FPGA can be reconstructed and the network where switch 2 is located can be selected. The transformation of the network mitigated the DDos attack to a certain extent.

(6) Vulnerability Attack. The mimic encryption box adopts the ARM + FPGA software and hardware cooperation method. All packets must be "reviewed" by the FPGA, enabling the prevention of unauthorized data access or network attacks, and the FPGA's security is much higher than that of software security products. Using the FPGA network interface for parameter configuration can realize one-way transmission control, protocol filtering, and the content filtering of packets. It can effectively prevent the use of the vulnerabilities or the weaknesses of multimedia equipment in order to carry out targeted intrusion and destruction. At the same time, by using FPGA hardware encryption, only the decrypted data can be transmitted to the upper-layer application; therefore, the packets containing malicious content are displayed as garbled after decryption, and normal attacks on the software system cannot be carried out.

4.3.4. Comparison with Other Schemes. Considering defense features such as dynamic, diversity, intrusion detection, encrypted transmission, virtualization, and hardware protection, a comparison of the defense mechanisms of different schemes is shown in Table 9.

Table 9 reveals that offering more advantages than other solutions, the mimic encryption box not only combines the randomness, dynamics, and diversity defense characteristics of MTD and CMD but also integrates hardware protection and reconfigurable encryption technology. In addition, most of the schemes in Table 9 adopt the software implementation and are built on the operating system. If the operating system itself has vulnerabilities, someone can bypass the protection mechanism and launch attacks. The mimic encryption box implements a security protection mechanism on the FPGA, does not rely on the operating system, and uses the FPGA's high anti-interference ability to filter some system attacks. Second, the mimic encryption box strictly controls the illegal traffic of the network to the front-end equipment with rule filtering. Third, the mimic encryption box uses FPGA encryption to increase the uncertainty of the system without reducing system performance, thereby increasing the difficulty of attack. However, some existing solutions use virtualization technology. Although layer-by-layer virtualization brings a certain degree of security, it loses performance. Fourth, the mimic encryption box uses a dynamically variable encryption algorithm and uses a different key to encrypt each packet. Even if an attacker intercepts a large number of packets, it is difficult to successfully implement the ciphertext-only attack. Finally, in view of the diversity and mobility of network multimedia data and equipment, the mimic encryption box can be easily connected to the original network. Compared with other solutions, it not only has higher security but also has portability and scalability for deployment.

5. Scope and Limitations

The mimic encryption box uses reconfigurable hardware to realize dynamic random encryption, network structure transformation, and data security filtering, which improves the security of sensitive data transmission and has good reliability. It can be deployed at the edge of the network and bound with terminal equipment to enhance the security of network data transmission, such as that associated with network cameras, video terminal equipment, data collection equipment, and self-service terminals. Second, the mimic encryption box can also be deployed in the Ethernet to complete end-to-end data encryption transmission. For the internal communications of the military and government

Table 9: Comparison between this scheme and other schemes.

Scheme	Dynamic	Diversity	Intrusion detection	Encrypted transmission	Virtualization	Hardware protection	Defensive effect
Aydeger et al. [9]	√	√	√				Defend against crossfire attacks
Aydeger et al. [10]	√	√	√		√		Resist crossfire attacks and provide network forensics
Wang et al. [12]	√	√					Increase the scanning space
Zhao et al. [15]	√	√					Defend against sniffer attack
Tang et al. [26]	√	√		√			Provide dynamic encryption; resist both exhaustive and analysis attacks
Lin et al. [34]			√			√	Detect the security of packets
Our system	√	√	√	√		√	Provide dynamic network and encryption and access control filtering; resist ciphertext-only and key exhaustion attacks

confidential departments, it can provide high-security services from the inside to outside. Finally, the mimic encryption box can also be applied to blockchain, cloud security, data security, and other fields. While using hardware to improve security, it can also be used to accelerate application calculations, such as signatures, verification, and data integrity verification.

The mimic encryption box provides the management of reconfigurable cryptographic algorithms and keys, enabling multimedia devices to conveniently call cryptographic services and to complete application layer encryption and authentication of video and voice data. It also provides fine-grained security and authentication services. Second, the mimic encryption box uses ACL to filter packets, strictly controls the access of illegal traffic, and has a certain anti-infiltration function. Finally, the mimic encryption box uses a "one packet, one key" encryption mechanism and has a certain self-cleaning function with a self-transforming network capability and a feedback mechanism. After being attacked, it can be "online" again in a reconstructed way.

At present, the mimic encryption box mainly completes the secure encryption of network multimedia data. In the future, we will consider using secure tunnel technology to provide better antireplay and antitampering functions with time factor and integrity verification. In addition, limited by FPGA resources, the current mimic encryption box has a mediocre performance in a 10G network. Therefore, how to use the idea of mimic defense to design a mimic encryption box or gateway with higher performance and higher security and apply it in 10 G/100G networks still needs further research.

6. Conclusions and Future Work

The mimic encryption box proposed in this paper optimizes the implementation of the reconfigurable hash algorithm, symmetric encryption algorithm, and elliptic curve algorithm, thereby improving the processing efficiency of data encryption and decryption. It uses a pseudorandom number generator to generate the initial key and updates the key with the hash value of the packet in order to realize the random encryption mode of "one packet, one key." Dynamic network information is realized by IP address hopping, port hopping, and protocol camouflage. The firewall function is realized by the access control list formed by five tuples, which limits the illegal access of the attacker. At the same time, the FPGA is managed through the ARM, and under abnormal conditions, the feedback information is used to realize the "cleaning" and reconstruction of the FPGA so that the mimic encryption box can work again. The experimental results and the analysis show that the mimic encryption box not only has higher encryption and decryption throughput but also has higher security. It can effectively prevent data leakage and tampering, disrupt attackers, and weaken network sniffing and vulnerability attacks. In addition, it can resist key exhaustive attacks and ciphertext-only attacks. It is suitable for applications with high security requirements.

Further study is suggested for the analysis of a mimic encryption gateway with higher performance, ways to improve its processing performance in 10G/100G networks, and the expansion of the use of the mimic encryption box across a range of system applications. At the same time, further study is suggested on ways to combine the mimic encryption box with SDN, as well as the use of the mimic encryption box to achieve a better mimic defense with a time-varying network, through changing the routing and composition of the network and forming a combination of multilayer changes, thereby effectively resisting network attacks.

Data Availability

All data generated or analyzed during this study are included in this article.

Conflicts of Interest

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

References

[1] A. Nauman, Y. A. Qadri, M. Amjad, Y. B. Zikria, M. K. Afzal, and S. W. Kim, "Multimedia Internet of Things: a comprehensive survey," *IEEE Access*, vol. 8, pp. 8202–8250, 2020.

[2] P. Chauhan, A. Choudhary, and A. K. Gupt, "Multimedia big data security,," in *Proceedings of the 2018 International Conference on Recent Innovations in Electrical, Electronics & Communication Engineering (ICRIEECE)*, pp. 112–117, Bhubaneswar, India, 2018.

[3] V. Nikhila and C. Rupa, "Intensifying multimedia Information security using comprehensive cipher," in *Proceedings of the 2019 Innovations in Power and Advanced Computing Technologies (I-PACT)*, pp. 1–4, Vellore, India, 2019.

[4] X. Li, J. Xu, H.-N. Dai, Q. Zhao, C. F. Cheang, and Q. Wang, "On modeling eavesdropping attacks in wireless networks," *Journal of Computational Science*, vol. 11, pp. 196–204, 2015.

[5] S. Ma, T. Zhang, A. Wu, and X. Zhao, "Lightweight and privacy-preserving data aggregation for mobile multimedia security," *IEEE Access*, vol. 7, pp. 114131–114140, 2019.

[6] H. Zhou, Y. Ma, L. Yan et al., "A Time-based self-cleaning control mechanism in moving target defense," in *Proceedings of the IEEE 4th Information Technology and Mechatronics Engineering Conference (ITOEC)*, pp. 983–986, Chongqing, China, 2018.

[7] X. Zhou, Y. Lu, Y. Wang et al., "Overview on moving target network defense," in *Proceedings of the IEEE 3rd International Conference on Image, Vision and Computing (ICIVC)*, pp. 821–827, Chongqing, China, 2018.

[8] B. Zhang, X. Chang, and J. Li, "A generalized information security model SOCMD for CMD systems," *Chinese Journal of Electronics*, vol. 29, no. 3, pp. 417–426, 2020.

[9] A. Aydeger, N. Saputro, K. Akkaya et al., "Mitigating crossfire attacks using SDN-based moving target defense," in *Proceedings of the IEEE 41st Conference on Local Computer Networks (LCN)*, pp. 627–630, Dubai, China, 2016.

[10] A. Aydeger, N. Saputro, K. Akkaya et al., "A moving target defense and network forensics framework for ISP networks using SDN and NFV," *Future Generation Computer Systems*, vol. 94, pp. 496–509, 2019.

[11] K. Zeitz, M. Cantrell, R. Marchany et al., "Changing the game: a micro moving target IPv6 defense for the Internet of Things," *IEEE Wireless Communications Letters*, vol. 7, no. 4, pp. 5778–6581, 2018.

[12] K. Wang, X. Chen, and Y. Zhu, "Random domain name and address mutation (RDAM) for thwarting reconnaissance attacks," *PLoS One*, vol. 12, no. 5, pp. 1–22, 2017.

[13] J. Zheng and A. S. Namin, "The Impact of address changes and host diversity on the effectiveness of moving target defense strategy," in *Proceedings of the IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, pp. 553–558, Atlanta, GA, USA, 2016.

[14] J. B. Hong and D. S. Kim, "Assessing the effectiveness of moving target defenses using security models," *IEEE Transactions on Dependable & Secure Computing*, vol. 13, no. 2, pp. 163–177, 2015.

[15] Z. Zhao, D. Gong, B. Lu, F. Liu, and C. Zhang, "SDN-based double hopping communication against sniffer attack," *Mathematical Problems in Engineering*, vol. 2016, Article ID 8927169, 13 pages, 2016.

[16] J. H. Jafarian, E. Al-Shaer, and Q. Duan, "An effective address mutation approach for disrupting reconnaissance attacks," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 12, pp. 2562–2577, 2015.

[17] J. H. Jafarian, E. Al-Shaer, and Q. Duan, "Adversary-aware IP address randomization for proactive agility against sophisticated attackers," in *Proceedings of the Computer Communications (INFOCOM)*, pp. 738–746, Kowloon, Hong Kong, 2015.

[18] H. Hu, Z. Wang, G. Cheng, and J. Wu, "MNOS: a mimic network operating system for software defined networks," *IET Information Security*, vol. 11, no. 6, pp. 345–355, 2017.

[19] H. Hu, J. Wu, Z. Wang, and G. Cheng, "Mimic defense: a designed-in cybersecurity defense framework," *IET Information Security*, vol. 12, no. 3, pp. 226–237, 2018.

[20] B. Ma and Z. Zhang, "Security research of redundancy in mimic defense system," in *Proceedings of the 3rd IEEE International Conference on Computer and Communications (ICCC)*, pp. 2910–2914, Chengdu, China, 2017.

[21] C. Qi, J. Wu, G. Cheng, J. Ai, and S. Zhao, "An aware-scheduling security architecture with priority-equal multi-controller for SDN," *China Communications*, vol. 14, no. 9, pp. 144–154, 2017.

[22] W. Liu, F. Chen, H. Hu et al., "A novel framework for zero-day attacks detection and response with cyberspace mimic defense architecture," in *Proceedings of the International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, pp. 50–53, Nanjing, China, 2017.

[23] H. Li, J. Hu, H. Ma et al., "The architecture of distributed storage system under mimic defense theory," in *Proceedings of the IEEE International Conference on Big Data (Big Data)*, pp. 2658–2663, Boston, MA, USA, 2017.

[24] A. Abusukhon, Z. Mohammad, and M. Talib, "A novel network security algorithm based on encrypting text into a white-page image," in *Proceedings of the World Congress on Engineering and Computer Science*, pp. 1–5, San Francisco, CA, USA, 2016.

[25] A. Abusukhon, M. N. Anwar, Z. Mohammad, and B. Alghannam, "A hybrid network security algorithm based on Diffie Hellman and text-to-image encryption algorithm," *Journal of Discrete Mathematical Sciences and Cryptography*, vol. 22, no. 1, pp. 65–81, 2019.

[26] H. Tang, Q. T. Sun, X. Yang, and K. Long, "A network coding and DES based dynamic encryption scheme for moving target defense," *IEEE Access*, vol. 6, pp. 26059–26068, 2018.

[27] A. Khan, Q. T. Sun, Z. Mahmood, and A. U. Ghafoor, "Energy efficient partial permutation encryption on network coded MANETs," *Journal of Electrical and Computer Engineering*, vol. 2017, pp. 1–10, 2017.

[28] W. Jiang, H. Xu, H. Dong, H. Jin, and X. Liao, "An improved security framework for web service-based resources," *Turkish Journal of Electrical Engineering & Computer Sciences*, vol. 24, no. 3, pp. 774–792, 2016.

[29] J. Li, Y. Zhang, X. Chen, and Y. Xiang, "Secure attribute-based data sharing for resource-limited users in cloud computing," *Computers & Security*, vol. 72, pp. 1–12, 2018.

[30] X. Jia, D. He, N. Kumar, and K.-K. R. Choo, "A provably secure and efficient identity-based anonymous authentication scheme for mobile edge computing," *IEEE Systems Journal*, vol. 14, no. 1, pp. 560–571, 2020.

[31] N. Indira, S. R. Devi, and A. V. Kalpana, "R2R-CSES: proactive security data process using random round crypto security encryption standard in cloud environment," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–12, 2020.

[32] L. A. Maciel, M. A. Souza, H. C. Freitas et al., "Reconfigurable FPGA-based K-means/K-modes architecture for network

Intrusion detection," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 8, pp. 1459–1463, 2020.

[33] J. A. Joseph, R. Korah, and S. Salivahanan, "Efficient string matching FPGA for speed up network Intrusion detection," *Applied Mathematics & Information Sciences*, vol. 12, no. 2, pp. 397–404, 2018.

[34] S. Lin, D. Zhang, Y. Fu et al., "A design of the ethernet firewall Based on FPGA," in *Proceedings of the 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, pp. 1–5, Shanghai, China, 2017.

[35] S. M. Keni and S. Mande, "Packet filtering for IPV4 protocol using FPGA," in *Proceedings of the Second International Conference on Intelligent Computing and Control Systems (ICICCS)*, pp. 400–403, Madurai, India, 2018.

[36] R. Ricart-Sanchez, P. Malagon, J. M. Alcaraz-Calero et al., "Net FPGA-based firewall solution for 5G multi-tenant architectures," in *Proceedings of the IEEE International Conference on Edge Computing (EDGE)*, pp. 132–136, Milan, Italy, 2019.

[37] H. E. Michail, G. S. Athanasiou, V. I. Kelefouras et al., "Area-throughput trade-offs for SHA-1 and SHA-256 hash functions' pipelined designs," *Journal of Circuits, Systems and Computers*, vol. 25, no. 4, pp. 1–27, 2016.

[38] S. Suhaili and T. Watanabe, "High throughput evaluation of SHA-1 implementation using unfolding transformation," *ARPN Journal of Engineering and Applied Sciences*, vol. 11, no. 5, pp. 3350–3355, 2016.

[39] M. M. Wong, M. L. D. Wong, C. Zhang et al., "Circuit and system design for optimal lightweight AES encryption on FPGA," *IAENG International Journal of Computer Science*, vol. 45, no. 1, pp. 52–62, 2018.

[40] A. Hafsa, A. Sghaier, M. Machhout et al., "A new security approach to support the operations of ECC and AES algorithms on FPGA," in *Proceedings of the 19th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*, pp. 95–100, Sousse, Tunisia, 2019.

[41] M. S. Hossain, Y. Kong, E. Saeedi et al., "High-performance elliptic curve cryptography processor over NIST prime fields," *IET Computers & Digital Techniques*, vol. 11, no. 1, pp. 33–42, 2016.

[42] S. Khan, K. Javeed, and Y. A. Shah, "High-speed FPGA implementation of full-word Montgomery multiplier for ECC applications," *Microprocessors and Microsystems*, vol. 62, pp. 91–101, 2018.

[43] W. Yu, K. Wang, B. Li et al., "Montgomery algorithm over a prime field," *Chinese Journal of Electronics*, vol. 28, no. 1, pp. 43–48, 2019.

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

[45] K. N. Devika and R. Bhakthavatchalu, "Design of reconfigurable LFSR for VLSI IC testing in ASIC and FPGA," in *Proceedings of the International Conference on Communication and Signal Processing (ICCSP)*, pp. 928–932, Chennai, China, 2017.