

Article

A Mimic Active Defense Method Based on Multiple Encryption Structure Encodings

Sisi Shao ^{1,2,*}, Yuchen Shi ^{2,3}, Nan Fu ¹ , Wangjie Hu ¹, Shangdong Liu ^{2,3}  and Yimu Ji ^{2,3,*}

¹ School of Cyber Science and Engineering, Wuxi University, Wuxi 214105, China; funan1988@cw Xu.edu.cn (N.F.); 25385210@stu.cw Xu.edu.cn (W.H.)

² Institute of High Performance Computing and Bigdata, Nanjing University of Posts and Telecommunications, Nanjing 210003, China; 1222045907@njupt.edu.cn (Y.S.); lsd@njupt.edu.cn (S.L.)

³ School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing 210023, China

* Correspondence: shaosisi@cw Xu.edu.cn (S.S.); jiyim@njupt.edu.cn (Y.J.)

Abstract

To break the long-standing imbalance of “easy to attack, hard to defend” in cyberspace, cyberspace mimic defense (CMD) has been proposed. It provides active defense against known and unknown vulnerabilities and backdoors via a dynamic heterogeneous redundancy (DHR) mechanism. Although the DHR architecture can suppress disturbances from physical factors or endogenous faults, data processing and transmission inside the architecture are usually in plaintext, which poses severe threats to data privacy. To address this problem, we propose a mimic active defense method based on multiple encryption structure encodings. For heterogeneity, an encryption/decryption component set module is designed to achieve data-level heterogeneity among executors. For dynamics, an executor identifier encryption scheme based on the elliptic curve digital signature algorithm (ECDSA) is used to protect the executor selection process. Meanwhile, a dynamic scheduling algorithm based on historical confidence is applied to reduce the impact of faulty executors. Experimental results show that the proposed method has obvious advantages in data privacy protection in terms of average historical confidence, execution efficiency, and bit error rate.

Keywords: endogenous security; active defense; cyberspace mimic defense; dynamic heterogeneous redundancy architecture

1. Introduction

Over the past several decades, continuous advances in information and communication technologies (ICT) [1,2] have ushered human society into the information age, driving varying degrees of digital transformation across virtually all sectors. These advances have not only profoundly reshaped everyday lifestyles but have also rapidly promoted national economic development. In the process of digital transformation, network systems have been widely deployed across social domains and have become indispensable infrastructure of modern society. However, the massive access of terminal devices has significantly expanded the attack surface of network systems, while increasingly sophisticated network attack techniques pose severe threats to cyberspace security [3,4]. Malicious software and network attacks—such as worms, Trojans, denial-of-service attacks, and advanced persistent threat (APT) attacks—have emerged as major challenges to network security. Although existing defense mechanisms against network attacks have become relatively mature, the uncertainty and persistence of threats have led to the long-standing imbalance of “easy



Academic Editor: Zhengqiu Zhang

Received: 12 February 2026

Revised: 4 March 2026

Accepted: 6 March 2026

Published: 10 March 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and

conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

to attack, hard to defend” in networks [5–7]. Consequently, optimizing network defense systems and introducing active defense mechanisms as well as more flexible system architectures have become key research directions for enhancing network security capabilities. The theory of cyberspace mimic defense (CMD) [5], proposed by academician Wu J X, is forming a new paradigm for the development of network security and provides a potential solution to the long-standing global challenge of uncontrollable endogenous security issues in fields such as industrial control systems (ICS) and cyber-physical systems (CPS). By leveraging a dynamic heterogeneous redundancy (DHR) architecture, CMD enables multi-executor execution, majority voting, and multidimensional reconfiguration, using system uncertainty to counter the pervasive uncertainty of threats in network, thereby significantly enhancing system security and reliability [8–10].

It is worth noting that the conventional DHR architecture still exhibits shortcomings in terms of data privacy protection. Structural encoding [6], an innovative concept proposed by Academician Wu J X on the basis of DHR, serves as a critical complement to DHR architecture. By designing sets of structural and algorithmic encodings with identical functionality, structural encoding can mask or correct differential-mode manifestations of generalized uncertain disturbances within executors, thereby further extending the defensive depth of DHR. However, the core structural encodings in traditional DHR architectures are mostly implemented in plaintext. When third parties (e.g., cloud service providers) participate in data processing, accidental operations or malicious intent may lead to unauthorized data access, tampering, or leakage, thereby exacerbating data privacy risks. Even when data are encrypted, attackers may still exploit data correlations and metadata for analytical inference, indirectly obtaining sensitive information. This highlights the urgent need for a more fine-grained encrypted structural encoding scheme for DHR.

To address this issue, we integrate the DHR structural encoding architecture with multiple encryption structure encodings by designing a cryptographic component set module, an executor selection and matching module, and a dynamic scheduling module. Specifically, we propose a mimic active defense method based on multiple encryption structure encodings to enhance the data privacy capabilities of traditional DHR architectures. Specifically, encryption architecture encodings are first selected according to differences in encryption types, encryption principles, and key lengths, thereby introducing heterogeneity into the encryption structure encoding components. Subsequently, an executor identifier encryption algorithm based on the elliptic curve digital signature algorithm (ECDSA) is employed to encrypt identifier information during transmission, and a signature verification mechanism is designed to prevent identifier tampering. Finally, majority voting is applied to adjudicate output results, the historical confidence of each executor is computed, and dynamic scheduling of executors is performed based on credibility information. The main contributions of this paper are as follows.

1. A DHR structural encoding scheme based on multiple encryption algorithms is designed. We first develop a cryptographic component set module, which consists of a heterogeneous set of encryption/decryption structural encoding components and a public/private key repository. By incorporating multiple encrypted structural encodings under the principle of heterogeneity, this module provides fundamental support for executor diversity within the overall architecture. In addition, a dynamic scheduling algorithm based on historical confidence is employed to evaluate the historical performance of executors, perform online/offline switching and cleansing of executors that do not meet the requirements, ensure the reliability of the heterogeneous executor pool, and introduce dynamic adaptability into the entire architecture.
2. An executor identifier encryption algorithm based on ECDSA is proposed. Data security during the dynamic selection process in DHR is enhanced by encrypting the

executor identifiers. By incorporating elliptic curve cryptography into the overall architecture, the system is endowed with the capability to perform signature verification and encryption of identifier information. Furthermore, a bidirectional authentication and response mechanism between the encryption and decryption parties is established to cope with potential identifier tampering, thereby ensuring the secure transmission of identifier information.

3. Simulation experiments demonstrate the effectiveness of the proposed method. Under different redundancy levels, the average historical confidence remains above 91%. The execution efficiency outperforms the encryption structure encodings based on AES, SM4, Camellia, and RSA, and only slightly higher than those based on DES and SM2. This illustrates that the proposed method can achieve a favorable dynamic balance between security performance and execution overhead. In the constructed persistent attack and random attack scenarios, the average bit error rates are 0 and 0.002, respectively, which verifies that the proposed method possesses excellent attack resistance and security performance.

The structure of the rest of this paper is as follows. Section 2 reviews the related background on DHR architectures and encryption algorithms. Section 3 presents the proposed mimic active defense architecture based on multiple encryption structure encodings and the implementation procedure of the proposed method. Section 4 validates the superior performance of the proposed approach through experiments. Finally, Section 5 concludes the paper and outlines directions for future research.

2. Related Work

Since mimic defense theory was proposed, research on its practical applications has progressed rapidly. Researchers have continuously attempted to integrate mimic defense theory with existing defense mechanisms to develop innovative security technologies [11,12]. Yu X. S. et al. [13] proposed a microservice-based mimic defense approach and conducted security and performance evaluations, demonstrating the effectiveness and feasibility of mimic defense theory. Ma H. L. et al. [14] and Song K. et al. [15] addressed security threats arising from unknown vulnerabilities and backdoors in Ethernet switches and routers by proposing DHR-based endogenous security architectures, thereby enabling active defense for switching and routing systems. Xu M. X. et al. [16] investigated potential security issues in blockchain systems and proposed a mimic blockchain framework from the perspectives of dynamic heterogeneous consensus mechanisms and dynamic heterogeneous redundant signature algorithms. The experimental results showed that the mimic blockchain outperforms traditional blockchains in terms of security. Zhou Q. L. et al. [17] designed a mimic defense authentication method based on the DHR architecture, using mobile QR codes as interfaces and dynamic passwords as the core, effectively addressing the vulnerability of traditional physical access control authentication schemes. Wang Z. P. et al. [18] proposed a mimic DNS with intrusion tolerance to overcome the deployment difficulties of DNS security extensions, thereby ensuring DNS security. Yuan H. Y. et al. [19] introduced a DHR-based honeypot model to enhance protection against unknown vulnerabilities and backdoors, enabling active defense and improved attack tracing and entrapment capabilities. Pu L. et al. [20] proposed a mimic CloudSim system to address the lack of cloud service tools for evaluating newly proposed mimic defense mechanisms, facilitating modeling and simulation of MD mechanisms. Pu L. M. et al. [21] combined mimic defense theory with cloud service technologies to address vulnerabilities and backdoors at different layers or components of cloud services, proposing a lightweight mimic cloud service architecture that overcomes the limitations of static single-node service structures. Overall, the integration of DHR with various application domains has fully

demonstrated its core value in enabling endogenous security through multi-executor execution, multi-mode decision-making, and multi-dimensional reconstruction. However, most existing studies focus on deployment adaptation, functional validation, and performance optimization of DHR architectures, while research on the structural encoding—the fundamental component that supports efficient DHR operation—remains insufficient, leaving notable gaps and opportunities for further exploration.

With respect to structural encoding in DHR architectures, Shi Y. et al. [22] proposed a Turbo-based structural encoding DHR architecture, in which a dynamic scheduling algorithm driven by anomaly feedback information selects component combinations from interleaver and encoder/decoder component sets to form the runtime pool. The decoding tasks are then distributed in parallel to all component combinations in the pool, and after the iterative decoding process is completed, a multimode majority voting algorithm is applied to produce frame-level decisions, while anomaly feedback information is computed and returned to the scheduling module to dynamically adjust the component combinations in the runtime pool. This approach endows the traditional DHR architecture with an iterative endogenous effect in information transmission and processing, thereby enhancing its resistance to attacks. Zheng Y. L. et al. [23] proposed a Polar-code-based structural encoding DHR architecture by constructing reconfigurable heterogeneous decoding executors, decoding decision mechanisms, and policy scheduling strategies. Based on decision results, compromised reconfigurable executors are identified, dynamically taken offline, and replaced, thereby improving system tolerance to unknown attacks. However, in these approaches, the structural encoding modes of mimic DHR architectures are largely statically fixed, enabling attackers to infer encoding patterns through long-term observation and data analysis and to launch targeted attacks, which weakens the uncertainty-based defensive effectiveness of DHR architectures; moreover, data privacy issues are not explicitly considered.

Therefore, this paper proposes a mimic active defense method based on multiple encryption structure encodings, in which diverse encryption and decryption structural encodings are introduced into the DHR structural encoding architecture to enhance the defensive capability of DHR against attacks from the perspectives of dynamism, heterogeneity, and redundancy of cryptographic executors. Specifically, in terms of heterogeneity and redundancy, an encryption/decryption component set module is designed to achieve data-level heterogeneity among executors. In terms of dynamism, an ECDSA-based executor identifier encryption algorithm is employed to protect the executor selection process, and a history–confidence-based dynamic scheduling algorithm is further adopted to effectively schedule executors according to their historical confidence levels, thereby reducing the impact of failed executors on the final output while simultaneously improving the security of dynamic scheduling.

3. Mimic Active Defense Method Based on Multiple Encryption Structure Encodings

On the basis of the traditional DHR structural encoding framework, multiple commonly used encryption structure encodings are introduced, and the overall architecture is illustrated in Figure 1. The heterogeneity of the proposed architecture is embodied in the design of the encryption/decryption component set module, where heterogeneous encryption structural encodings are selected from the perspectives of encryption principles and key lengths, and diversified key schemes are adopted in the public/private key repositories to further enhance key heterogeneity. The dynamic is reflected in the executor selection and matching module and the dynamic scheduling module. Through their coordination, executors are reliably brought online or taken offline based on their historical confidence

values, ensuring that unreliable executors can be replaced in a timely manner under external attacks and thereby improving architectural stability. The redundancy is realized in the decision output module, where r executors perform encryption and decryption on the same message m to generate m_r , and a majority voting mechanism is applied to the m_r outputs from the decryption executors to obtain the final result m_v . This redundant executor design significantly enhances the overall robustness and interference resistance of the architecture.

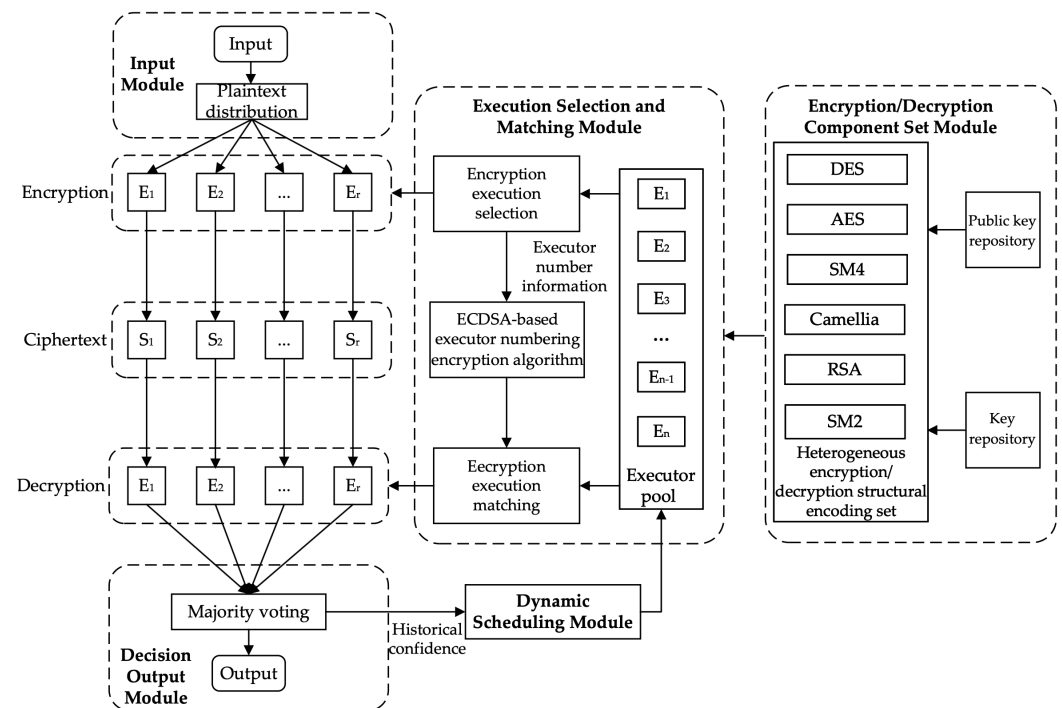


Figure 1. Mimic active defense architecture based on multiple encryption structure encodings.

The architecture consists of five main components: the input module, execution selection and matching module, encryption/decryption component set module, dynamic scheduling module, and decision output module. The functionality of each module is as follows.

1. **Input Module:** This module receives the plaintext to be encrypted. The plaintext distribution component duplicates and distributes the plaintext to each encryption executor.
2. **Execution Selection and Matching Module:** This module comprises encryption execution selection, an ECDSA-based executor numbering encryption algorithm, decryption execution matching, and a heterogeneous executor pool. The encryption execution selection is responsible for choosing executors from the waiting pool to come online. The ECDSA-based numbering encryption algorithm ensures secure communication of encryption executor numbers and corresponding key identifiers between encryption and decryption ends. Decryption execution matching aligns decryption units with the received encrypted executor numbers and provides matching keys for decryption. The heterogeneous executor pool stores executors created using different encrypted structural encodings, which can be selected for activation or scheduled for deactivation.
3. **Encryption/Decryption Component Set Module:** This module includes the heterogeneous encryption/decryption structural encoding set, a public key repository, and a key repository. The heterogeneous encoding set is composed of multiple asymmetric and symmetric encryption structural encodings, providing architectural heterogeneity. The public key repository stores the public keys for asymmetric encryption, while the

key repository stores private keys for asymmetric encryption and various lengths of symmetric encryption keys.

4. Dynamic Scheduling Module: Implemented via a dynamic scheduling algorithm based on historical confidence, this module deactivates unreliable executors based on historical confidence scores and notifies the encryption execution selection submodule to activate new executors. This process enhances the system's reliability under attacks targeting encryption executors.
5. Decision Output Module: This module comprises majority voting and the final output. The large-number decision process applies the "majority principle" to multiple decrypted plaintexts, producing a final output result that improves redundancy and fault tolerance.

The detailed execution process of the entire architecture is as follows.

Step 1: Initially, the information sender transmits the plaintext sequence m and the redundancy level r of the current transmission task to the input module.

Step 2: The input module instructs the encryption/decryption component set module to create n heterogeneous executors in the heterogeneous executor pool. Each executor is initialized with a historical confidence value δ , which will be used by both the encryption execution selection submodule and the decryption execution matching submodule.

Step 3: At time t , the encryption execution selection submodule selects r encryption executors based on their historical confidence values $T(t)$ and records the identifiers of the selected encryptors N_{Enc} .

Step 4: The identifiers N_{Enc} are encrypted using the ECDSA-based executor numbering encryption algorithm for secure transmission. At the decryption side, the encrypted identifiers are verified to ensure they have not been tampered with and then decrypted. The decryption execution matching submodule then matches the decrypted identifiers N_{Enc} to the corresponding decryption executors.

Step 5: The input module copies m into r copies and distributes it to the selected encryption executors for encryption. At the decryption side, the ciphertexts are input into the matched decryption executors, resulting in decrypted plaintexts m_r from each unit.

Step 6: The decrypted plaintexts m_r are input into the majority voting to obtain the final decided plaintext m_v . The historical confidence values $T(t)$ of each executor are then updated. For any executor with $T(t) < \zeta$, where ζ is the lower threshold for historical confidence, the dynamic scheduling module is invoked to offline the unreliable executors and replace them with other executors that satisfy $T(t) \geq \zeta$.

The mimic active defense method based on multiple encryption structure encodings endows the DHR architecture with encryption and decryption capabilities by designing a heterogeneous encryption/decryption component set, an ECDSA-based executor encoding encryption algorithm, and a dynamic scheduling algorithm based on historical confidence. This approach increases the representational uncertainty of the encryption structural codes, enhancing the defense capability and reliability of the mimic defense system against attacks targeting encrypted codes.

3.1. Heterogeneous Encryption/Decryption Component Set

The heterogeneity of our method is embodied in the encryption/decryption component set module. This module achieves architectural heterogeneity through the design of a heterogeneous encryption/decryption structural code component collection and a public/private key repository. The detailed design of the module is illustrated in Figure 2.

The heterogeneous encryption/decryption structure component set is selected according to three principles: encryption type, encryption mechanism diversity, and key length variation [24]. Encryption type enhances the heterogeneity of the component set by

combining symmetric and asymmetric encryption schemes. The encryption mechanism determines both the efficiency and security of the encryption structure, and using different mechanisms increases the heterogeneity of the component set, thereby raising the difficulty for attackers attempting to compromise the encryption executors. Key length is a critical indicator of encryption security, as varying key lengths significantly increase the complexity of brute-force attacks on the encryption executors. Based on these three principles, this work selects DES, AES, SM4, Camellia, RSA, and SM2 to form the heterogeneous encryption/decryption structure component set. Table 1 summarizes the differences among these components in terms of encryption mechanism and key length.

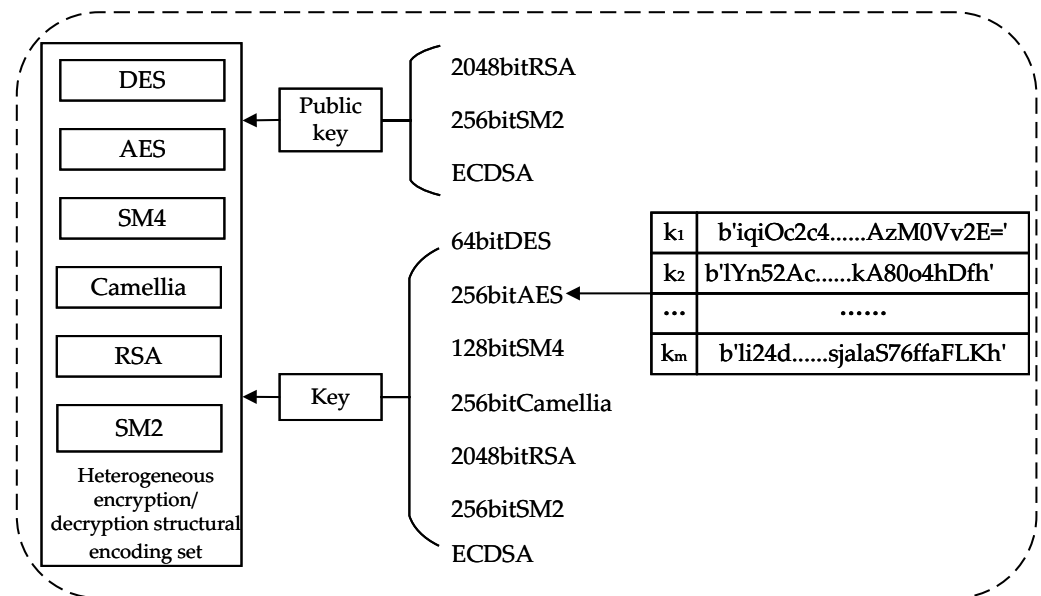


Figure 2. Encryption/decryption component set module.

Table 1. Differences between encryption/decryption structure components.

Name	Encryption Type	Encryption Mechanism	Key Length
DES	Symmetric	Permutation, S-box substitution, 16-round iteration	64 bit
AES	Symmetric	Key expansion, round key addition, 14-round iteration	256 bit
SM4	Symmetric	Key expansion, inverse transformation, 32-round iteration	128 bit
Camellia	Symmetric	Feistel network, FL function, 24-round iteration	256 bit
RSA	Asymmetric	Large prime factorization problem	2048 bit
SM2	Asymmetric	Elliptic curve cryptography (ECC)	256 bit

The selection of key lengths for each encryption/decryption structure component satisfies two conditions: ensuring the security of the encryption/decryption executors and maintaining key length diversity. Additionally, the key repository for each component stores pre-established keys agreed upon by both encryption and decryption parties. The number of keys is significantly greater than one, which reduces the risk of all keys being compromised and enhances the overall redundancy of the architecture. The selection of heterogeneous encryption algorithms inherently introduces performance disparities due to

their varying computational requirements. Asymmetric schemes such as RSA (2048-bit) typically incur higher computational overhead compared to symmetric ciphers like AES (256-bit) or SM4 (128-bit). To address this imbalance, the DHR architecture's inherent redundancy and majority adjudication mechanism play a crucial role. The system operates with a configurable redundancy level, where multiple executors process the same input in parallel, and the final output is determined by majority voting. This means the overall system performance is not bottlenecked by the slowest single executor, but rather by the aggregate performance of the selected set. Additionally, the heterogeneous executor pool allows the system to adaptively compose task sets, balancing the inclusion of lightweight and resource-intensive components to meet both security and performance requirements.

3.2. ECDSA-Based Executor Encoding Encryption Algorithm

The principle of DHR structural coding is to introduce multiple executors into the system, thereby enhancing the representational uncertainty of the architecture. However, when applying DHR structural coding in encryption and decryption scenarios, a challenge arises in matching executors between the encryption and decryption ends. The ECDSA-based executor encoding encryption algorithm addresses this issue by encrypting the executor identifiers using the ECDSA during transmission. This ensures that encryption executors and decryption executors are correctly paired at their respective ends, while effectively preventing attackers from exploiting mismatches between the two ends. As a result, the algorithm reduces inherent vulnerabilities within the architecture and enhances its resistance to attacks. We employ ECDSA for digital signature and identity authentication, which provides integrity and non-repudiation for messages in transit. Leveraging its elliptic curve properties, ECDSA enables efficient signature generation and verification with smaller key sizes, making it well-suited for validating the authenticity of signed messages prior to the transmission of any sensitive data. Concurrently, we utilize RSA for data confidentiality encryption, specifically to encrypt the signed messages and subsequent plaintext data. As a widely standardized and universally supported algorithm across nearly all execution environments, RSA minimizes compatibility risks and ensures seamless integration with diverse executor pools.

Based on Figure 3, which illustrates the successful signature verification flow for the ECDSA-based executor encoding encryption algorithm, all symbols are defined in the professional terminology reference table. Taking the encryption side A sending N_{Enc} to the decryption side B as an example, the algorithm execution flow is as follows.

Step 1: Initialize the algorithm parameters and create the transmission carrier. Encryption side A randomly or pseudo-randomly selects a private key k from the ECDSA private key library. On the elliptic curve $E(F_p)$, a base point G is selected with order n . The transmission carrier is defined by the algorithm; in this example, an RSA encryption executor is used.

Step 2: Generate an ECDSA signature. The N_{Enc} is input into the ECDSA signature algorithm. Through scalar multiplication and hash function computations, encryption side A calculates intermediate values r and s , resulting in the signature message (N_{Enc}, r, s) .

Step 3: RSA encryption. The signature message is input into the RSA encryption executor for encryption and sent it to decryption side B.

Step 4: Signature verification. Decryption side B decrypts the RSA message and uses the ECDSA verification algorithm to determine whether the signature has been tampered with. Based on the verification result, the system assesses whether the current plaintext transmission is secure. If the signature is verified successfully, decryption side B sends a verification success flag F_{Suc} back to encryption side A and proceeds to Step 5. If verification fails, as shown in Figure 4, decryption side B sends no response, and if encryption side A's

waiting time exceeds the predefined threshold, both sides terminate the task and notify the plaintext sender with a warning.

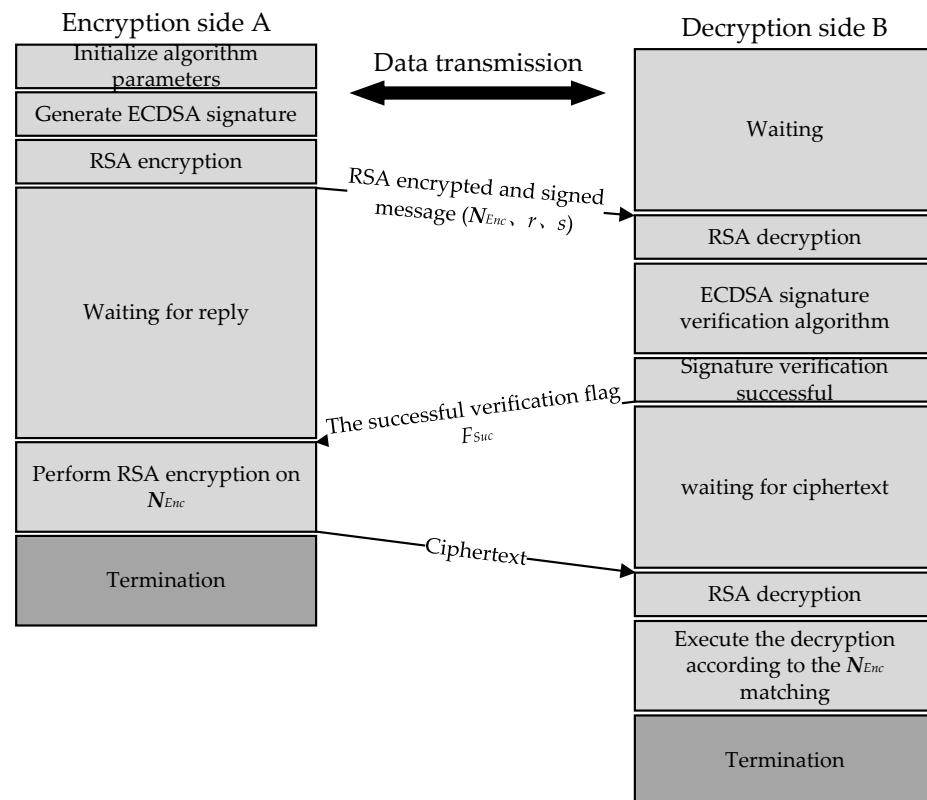


Figure 3. Execution flow of the ECDSA-based executor encoding encryption algorithm with successful signature verification.

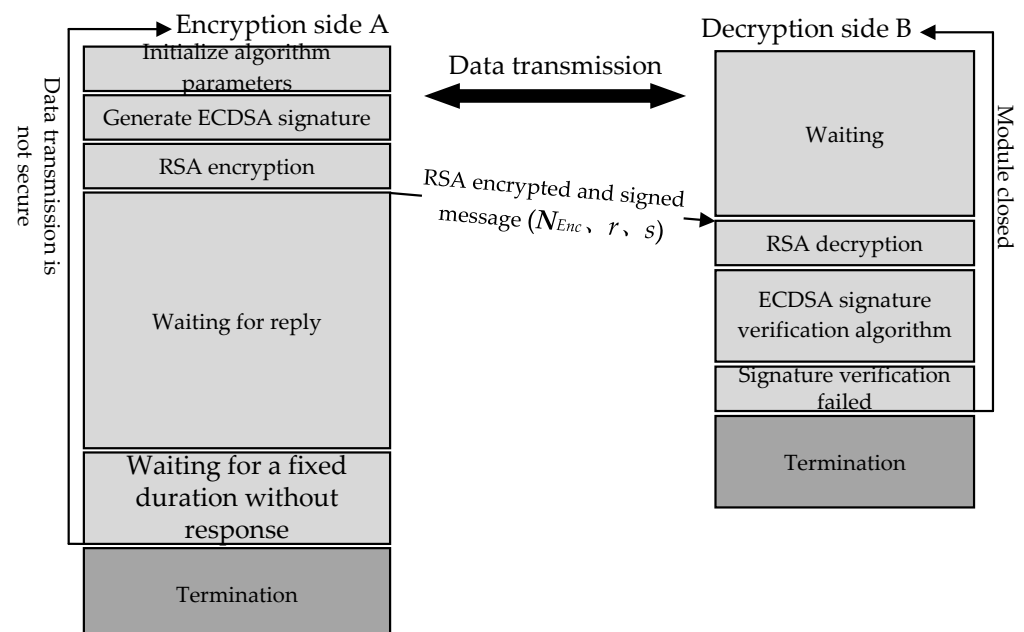


Figure 4. Execution flow of the ECDSA-based executor encoding encryption algorithm with failed signature verification.

Step 5: Encrypt and transmit N_{Enc} . After successful verification, upon receiving F_{Suc} , encryption side A immediately encrypts N_{Enc} using RSA and sends it to decryption side B.

Step 6: Decrypt N_{Enc} and match executors. Decryption side B successfully decrypts N_{Enc} and matches it with the corresponding decryption executor from the heterogeneous executor pool, preparing for the decryption task.

Step 7: Terminate algorithm module. Encryption side A and decryption side B both conclude the execution of this algorithm module.

This process ensures secure pairing of encryption and decryption executors, preventing mismatches and mitigating attacks targeting dual-end execution inconsistencies, thereby enhancing the system's robustness and security.

3.3. Dynamic Scheduling Algorithm Based on Historical Confidence

When the DHR structural coding is subjected to attacks, the unpredictable nature of attacker behavior increases the risk of compromise for online executors over time. The objective of improving the architecture is to provide a mechanism for evaluating the security of online executors. Therefore, the executor historical confidence $T(t)$ is introduced into the DHR scheduling algorithm as a basis for executor selection. Historical confidence reflects the trustworthiness of an executor; during its active period, it represents the probability that the executor remains uncompromised. For an online executor E_i , its historical confidence is calculated according to Equation (1), where count represents the number of plaintext transmissions.

$$T(t) = \frac{\sum_{j=1}^{count} t_{ij}(t)}{count} \quad (1)$$

For a redundancy level r in the proposed architecture, the average historical confidence is calculated according to Equation (2).

$$\bar{T}(t) = \frac{\sum_{i=1}^r T_i(t)}{r} \quad (2)$$

The online historical confidence is updated during the majority voting process in the decision output module. The algorithmic flow of the dynamic scheduling algorithm based on historical confidence is presented in Algorithm 1.

Algorithm 1 Dynamic scheduling algorithm based on historical confidence.

- 1: Input: Historical confidence of each executor $T_i(t)$, initial historical confidence δ , lower bound of historical confidence threshold ζ .
 - 2: Output: Scheduling scheme.
 - 3: for i in $1, 2, \dots, n$
 - 4: $T_i(t) = \delta$ // Initialize the historical confidence of each executor.
 - 5: end for
 - 6: for i in $1, 2, \dots, r$
 - 7: $T(t) = \frac{\sum_{j=1}^{count} t_{ij}(t)}{count}$ // Compute the historical confidence according to Equation (1).
 - 8: if $T_i(t) < \zeta$
 - 9: $Sort = [E_1, E_2, \dots, E_{n-r}]$ // Sort the executors in the waiting pool according to their historical confidence.
 - 10: $E_i = Sort[0]$
 - 11: end if
 - 12: end for
 - 13: update N_{Enc}
 - 14: input N_{Enc} to ECDSA // Transmit the updated executor identifier information to the decryption side.
 - 15: update E
-

4. Experiments and Analysis

4.1. Experimental Setup

We selected DES, AES, SM4, Camellia, RSA, and SM2 as the encryption/decryption structural encoding components in this study. Their specific characteristics and differences have been detailed in Section 3.1 (Heterogeneous Encryption/Decryption Component Set Module).

In the experiments, the number of executors in the heterogeneous executor pool is set to $n = 10$. When an executor is not selected as either an encryption or decryption executor, it remains in a standby state within the heterogeneous executor pool, and its system overhead is negligible and therefore ignored. To validate the effectiveness of the proposed method, the redundancy degree of the DHR structural encoding is configured as $r = 3$, $r = 4$, and $r = 5$. The size of the key pool for each encryption/decryption component is set to $s = 100$, and all keys are generated using a random key generation algorithm. The length of the plaintext sequence to be encrypted is set to $count = 100$, $count = 500$, and $count = 1000$, respectively. Each plaintext in the sequence m is a UTF-8-encoded string with a length of 128 bytes. In addition, the initial value of the historical confidence is set to $\delta = 0.5$, and the lower threshold of historical confidence is set to $\zeta = 0.8$. The initial confidence $\delta = 0.5$ represents an unbiased neutral state, as no historical performance data is available for executors at system startup. This setting ensures an objective and fair starting point for confidence updates, avoiding any subjective bias toward specific executors. The lower threshold $\zeta = 0.8$ is determined through repeated experiments under various attack scenarios. It represents a reasonable trade-off between system reliability and scheduling stability: values below 0.8 would retain poorly performing executors and reduce security, while values above 0.8 would trigger excessive scheduling and increase unnecessary overhead. All experimental results are obtained through 1000 Monte Carlo simulations to ensure statistical reliability and accuracy. The detailed hardware and software environment used for the simulations is summarized in Table 2.

Table 2. Hardware and software experimental environment.

Name	Hardware Model/Software Name
CPU	Intel(R) Core(TM) i7-9700 CPU @ 3.00 GHz
GPU	NVIDIA Quadro P620
Simulation Software	PyCharm 24.3
Programming Language	Python 3.6
Operating Platform	Windows10

The simulation experiments in this study evaluate the performance of our method based on multiple encrypted structural encodings from three perspectives: average historical confidence, execution efficiency, and bit error rate (BER). The proposed method is compared with DHR architectures employing single encryption structural encodings, including DES, AES, SM4, Camellia, RSA, and SM2, to verify its fundamental effectiveness as well as its reliability under attack scenarios.

4.2. Experimental Results and Analysis

4.2.1. Average Historical Confidence

The average historical confidence of the DHR encryption structural encoding is determined by the historical confidence of the selected executors, and its specific calculation is given in Equation (2). A higher average historical confidence indicates a higher overall reliability of the architecture [25]. Our method performs scheduling based on the historical confidence of executors, preferentially selecting those with higher historical confidence to

execute encryption tasks. When an executor is subjected to attacks or encounters unknown faults, the number of erroneous decision results increases significantly, leading to a decrease in its historical confidence. Once the historical confidence drops below the predefined lower threshold ζ , the corresponding execution entity is switched out. In this simulation, the variations in average historical confidence of the proposed architecture under non-attack conditions and continuous attack conditions are evaluated for redundancy levels $r = 3$, $r = 4$, and $r = 5$. The corresponding results are illustrated in Figures 5 and 6.

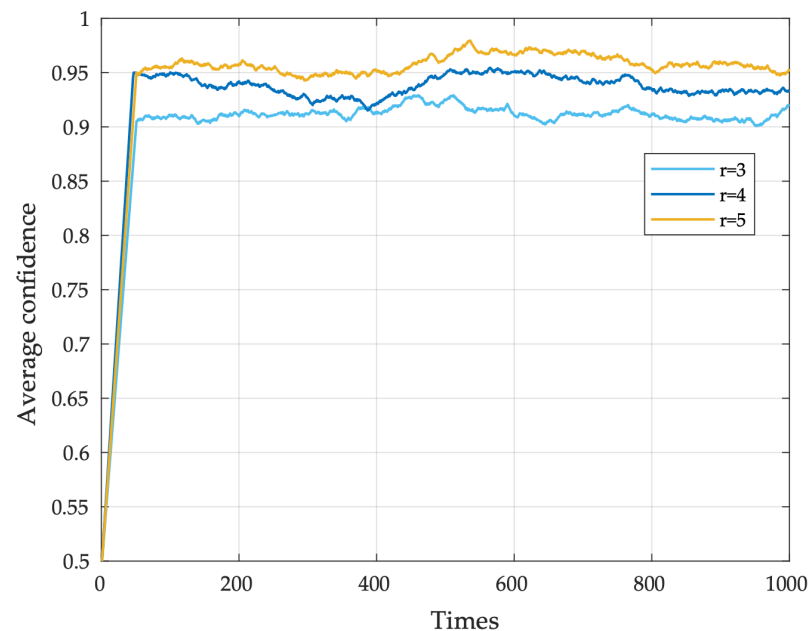


Figure 5. Comparison of the average historical confidence of the DHR architecture under different redundancy levels in the absence of attacks.

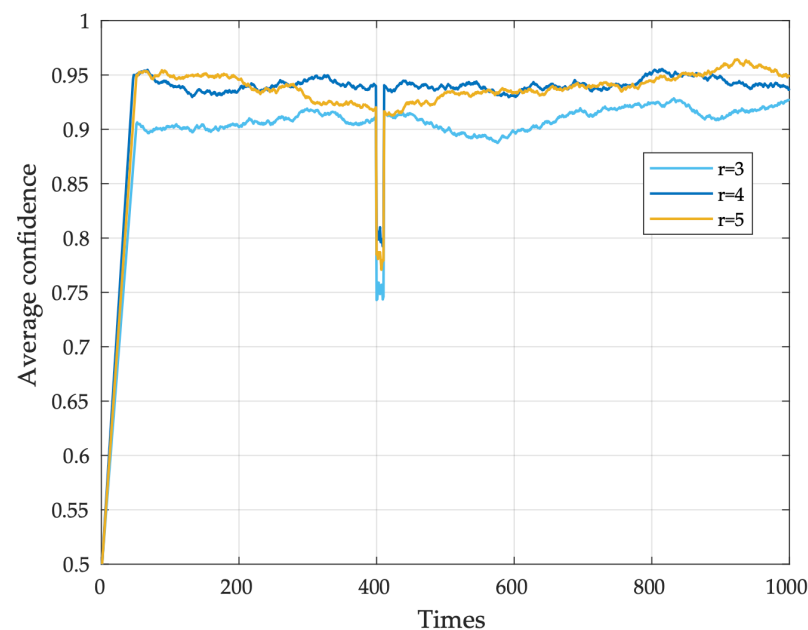


Figure 6. Comparison of the average historical confidence of the DHR architecture under different redundancy levels under continuous attacks.

From Figure 5, it can be observed that as the number of plaintext transmission tasks increases, the average historical confidence of our method under different redundancy levels continuously increases and gradually converges to a stable value. This behavior

arises because the historical confidence of encryption/decryption executors is dynamically updated during the majority voting process. Since no attacks occur in this scenario, the historical confidence of each executor steadily increases and eventually stabilizes. According to Equation (2), the average historical confidence therefore also exhibits a monotonic increase followed by convergence. Moreover, Figure 5 shows that a higher redundancy level leads to a higher average historical confidence. This is because, in the absence of attacks, greater redundancy implies a larger number of active encryption executors, which enhances the overall security of the architecture and consequently improves the average historical confidence.

In this simulation, a continuous attack model is adopted, in which a single encryption executor is persistently attacked. Specifically, the attack is launched during transmission rounds 400–410. As shown in Figure 6, during the early stage of the experiment, when the architecture is not subjected to continuous attacks, the average historical confidence increases steadily and converges, similar to the no-attack scenario. However, once the continuous attack is introduced, the historical confidence of the targeted executor drops significantly, leading to a noticeable decline in the overall average historical confidence of the architecture. When the historical confidence of an executor falls below the threshold ζ , the scheduling module immediately takes the executor offline and replaces it with the executor having the highest historical confidence from the heterogeneous executor pool. The compromised executor is then cleaned and subsequently reintroduced into the pool. Since the newly selected executor has not been attacked, the average historical confidence of the entire architecture recovers, resumes an upward trend, and eventually stabilizes again.

4.2.2. Execution Efficiency

The execution efficiency of a single encryption structure encoding is defined as the sum of the encryption and decryption time. In contrast, the execution efficiency of the DHR architecture based on multiple encryption structure encodings additionally includes the overhead introduced by scheduling heterogeneous executors. In this study, the execution efficiency is measured from the moment the plaintext to be encrypted is input until the final output is generated. To demonstrate that the number of transmission tasks does not affect the validity of the simulation results, the execution efficiency of our method under different redundancy levels is first compared for transmission counts of $count = 100$, $count = 500$, and $count = 1000$, as shown in Figure 7 and Table 3. Subsequently, both vertical and horizontal comparisons are conducted. The vertical comparison evaluates the execution efficiency of the proposed architecture under three redundancy settings ($r = 3$, $r = 4$, and $r = 5$) when $count = 1000$. The corresponding results are presented in Figure 8 and Table 4. The horizontal comparison evaluates the execution efficiency of our method against multiple single encryption structure encodings under the condition $r = 3$. In this experiment, six representative single encryption schemes—DES, AES, SM4, Camellia, RSA, and SM2—are selected for comparison. The experimental results are illustrated in Figure 9 and Table 5.

As shown in Figure 7 and Table 3, it can be observed that different transmission counts have negligible impact on the execution efficiency of our method. Under varying transmission counts, the changes in the average execution time of our method for $r = 3$, $r = 4$, and $r = 5$ are insignificant. As the redundancy level increases, the number of encryption executors also increases, which leads to a higher average execution time for architectures with higher redundancy compared to those with lower redundancy.

According to the comparative execution efficiency data of our method under $r = 3$, $r = 4$, and $r = 5$ shown in Figure 8 and Table 4, the execution time increases as the

redundancy level increases. The difference in average execution time between $r = 3$ and $r = 4$ is relatively small, whereas a significant increase is observed when $r = 5$. This is mainly because, under the condition $r = 5$, encryption/decryption executors based on the SM2 cryptographic structure are highly likely to be selected for execution. In this experiment, the SM2 executor is implemented using the gmssl library, whose computational overhead is relatively high, thereby resulting in lower execution efficiency of the DHR architecture based on multiple cryptographic structure encodings when $r = 5$. In mimic defense research, triple modular redundancy ($r = 3$) is the most commonly adopted configuration in the construction of DHR cryptographic structure encodings. As can also be observed from Figure 7, the execution efficiency reaches its highest level when $r = 3$. Under this redundancy level, our method achieves a dynamic balance between security performance and execution overhead.

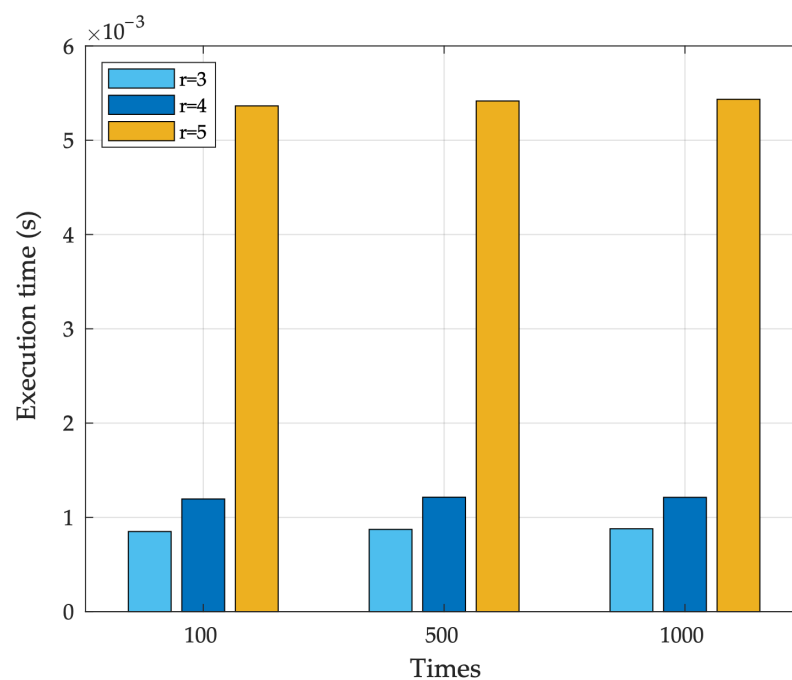


Figure 7. Comparison of the average execution time of our method under different transmission counts.

Table 3. Average execution time of our method under different transmission counts.

Type	Average Execution Time/s (count = 100)	Average Execution Time/s (count = 500)	Average Execution Time/s (count = 1000)
$r = 3$	0.00084983	0.00087254	0.000879297
$r = 4$	0.00119511	0.00121377	0.00132368
$r = 5$	0.00536428	0.00541633	0.00543421

According to the comparison results in Figure 9 and Table 5 for the case of $r = 3$, the execution efficiency of our method is lower than that of the AES, SM4, Camellia, and RSA cryptographic schemes, while it is higher than that of DES and SM2, placing it at an intermediate level overall. Considering the enhanced security guarantees provided by our method, the additional time overhead remains within an acceptable range. These results demonstrate the effectiveness and practical feasibility of the DHR architecture based on multiple cryptographic structure encodings in encryption scenarios.

Table 4. Execution efficiency of our method under different redundancy levels when *count* = 1000.

Type	Average Execution Time/s (<i>count</i> = 100)	Execution Efficiency
$r = 3$	0.00087927	100%
$r = 4$	0.00132368	66.43%
$r = 5$	0.00543421	16.18%

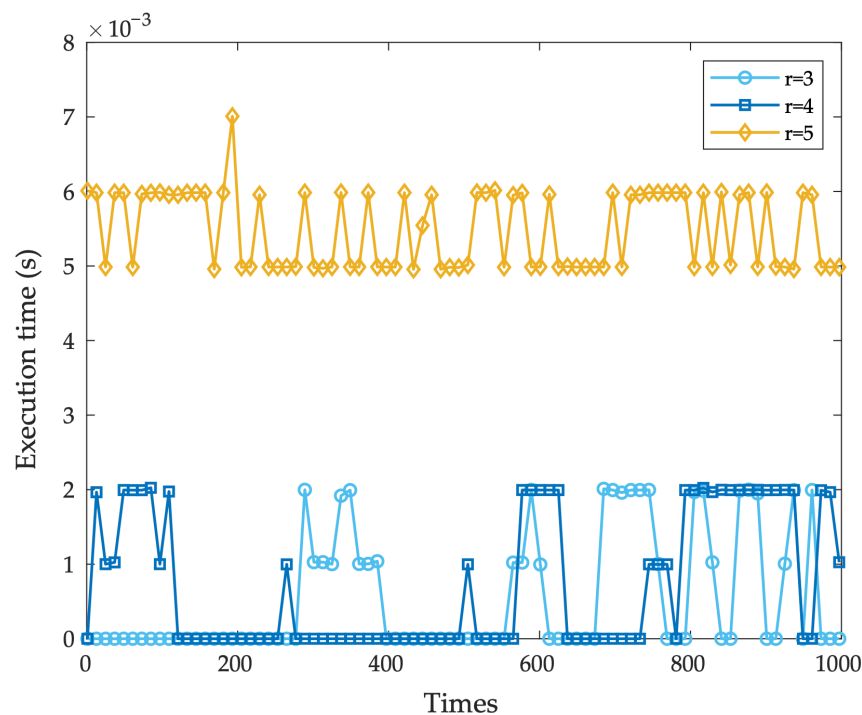
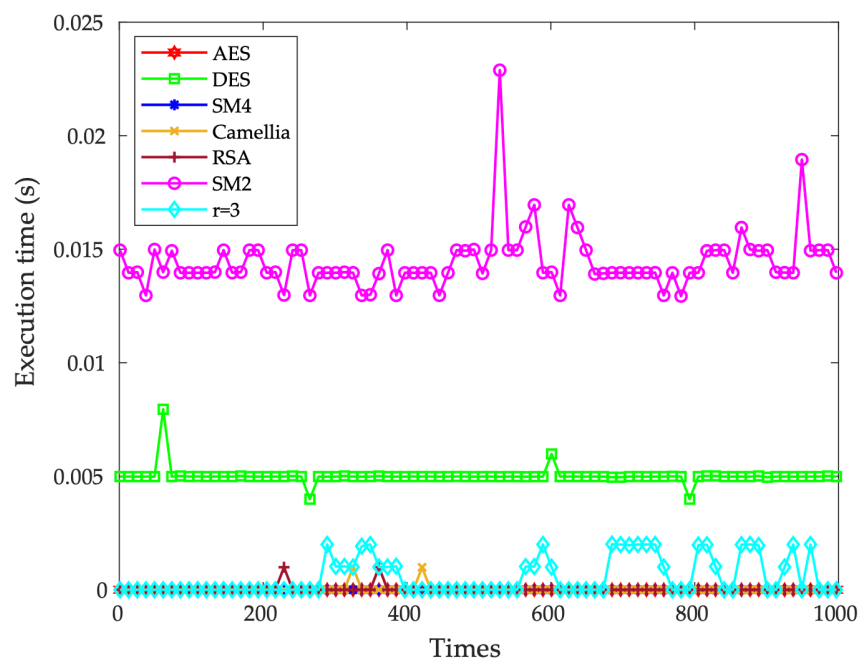
**Figure 8.** Comparison of execution efficiency of our method under different redundancy levels when *count* = 1000.**Figure 9.** Comparison of execution efficiency between the DHR architecture ($r = 3$) and other cryptographic components.

Table 5. Execution efficiency of multiple cryptographic structure encodings.

Type	Average Execution Time/s (count = 100)	Execution Efficiency
$r = 3$	0.00087927	6.012%
DES	0.00503452	34.42%
AES	0.00000393	0.027%
SM4	0.00000102	0.007%
Camellia	0.00002182	0.014%
RSA	0.00002008	0.014%
SM2	0.01462487	100%

4.2.3. Bit Error Rate

The bit error rate (BER) is defined as the ratio of incorrectly decrypted plaintext symbols to the total number of original plaintext symbols, and it is used to characterize the overall reliability of the encryption–decryption process of our method. In the simulations, both persistent attacks and random attacks are employed to evaluate the robustness of the architecture under adversarial conditions, thereby assessing its resistance to attacks.

It is assumed that the attacker launches attacks based on prior knowledge of the architecture. Without such prior information, the attacker cannot determine which encryption executor to target. If the prior information remains unchanged, the attack manifests as a persistent attack, in which the attacker concentrates on a specific encryption executor within our method and compromises plaintext integrity by tampering with the corresponding ciphertext. If the prior information changes, the attack manifests as a random attack, specifically targeting multiple encryption executors within the architecture of this paper, leading to the failure of majority voting and resulting in incorrect decisions, thereby compromising the decrypted plaintext information. These two attack models respectively simulate scenarios in which a single executor is attacked and multiple executors are attacked, and they are used to evaluate the anti-interference capability of the proposed architecture under both conditions.

In the simulation experiments presented in this paper, the persistent attack scenario adopts an information attack density of 50%. As shown in Figure 10, the RSA encryption executor is subjected to a persistent attack in this simulation, resulting in continuous fluctuations in its symbol error rate (SER). In contrast, the SERs of the SM2 and SM4 encryption executors remain zero, indicating that they are not affected by the persistent attack. In addition, the SER of the proposed architecture with redundancy $r = 3$ is also zero, demonstrating that our method can correctly decrypt ciphertexts under persistent attack conditions without being disturbed by such attacks. Compared to a single encryption structure encoding, our method exhibits superior attack resistance, which further highlights the effectiveness and superiority of the architectural design.

In the random attack scenario, an information attack density of 5% is adopted, and two encryption executors are attacked simultaneously. As illustrated in Figure 11, the AES and Camellia encryption executors are targeted in this random attack simulation, resulting in significant fluctuations in the SERs of both executors. In contrast, the overall SER curve of the proposed architecture with $r = 3$ exhibits much smaller fluctuations and remains approximately around $SER = 0$. The observed fluctuations are mainly caused by the fact that when more than $r/2$ encryption executors are attacked simultaneously, the majority voting algorithm has a certain probability of failure, which prevents the decision output module from fully guaranteeing the correctness of the output and leads to a temporary increase in the SER of our method. Nevertheless, as shown in Figure 11, the overall fluctuation amplitude of the SER curve of the proposed architecture is significantly smaller

than that of the attacked encryption executors. This result demonstrates that our method possesses strong robustness against random attacks and effectively enhances the stability and security of encryption structure encoding in encryption/decryption scenarios.

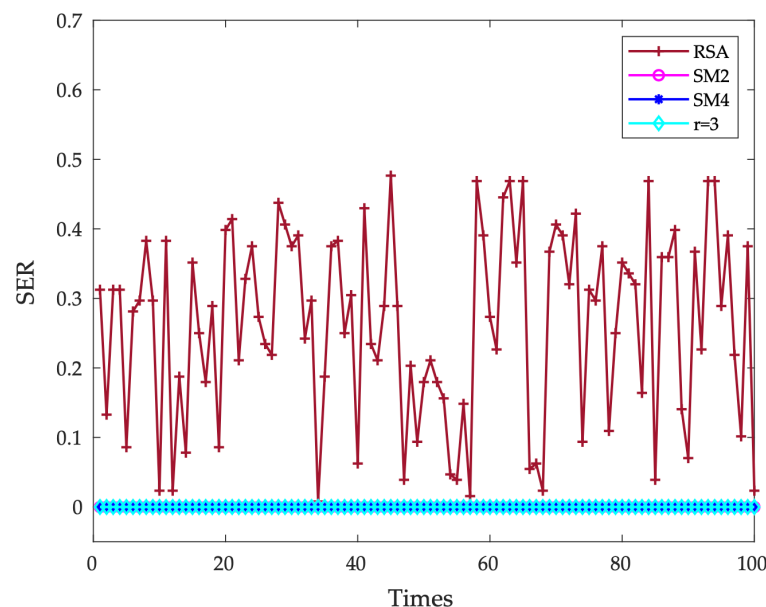


Figure 10. Comparison(there is no SM2 line in this figure, please check.) of the SER of our method and individual executors under persistent attacks.

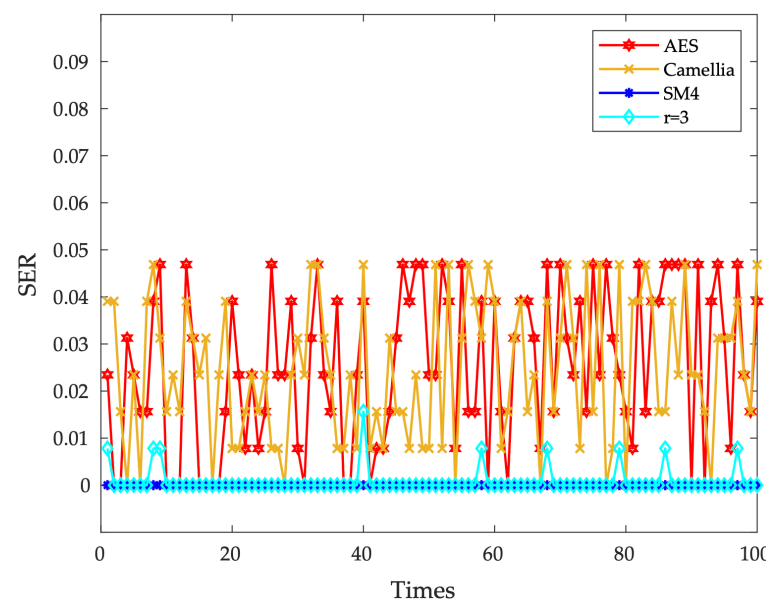


Figure 11. Comparison of the SER of our method and individual executors under random attacks.

5. Conclusions

To address the weak data protection capability of the traditional DHR architecture, this paper proposes a mimic active defense method based on multiple encryption structure encodings. By redesigning the encryption/decryption component set module, the executor selection and matching module, and the dynamic scheduling module, the structural encoding of the conventional DHR architecture is reconstructed. In the proposed architecture, executor identifier information is securely transmitted using an ECDSA-based executor encoding encryption algorithm to ensure correct matching between encryption and decryption executors. Subsequently, majority voting is performed on the outputs of decryption

executors, and the historical confidence of each executor is calculated and updated. Based on the historical confidence, executors are dynamically scheduled, thereby enhancing the data privacy protection capability of the traditional DHR architecture. Experimental evaluations were conducted from three perspectives: average historical trustworthiness, execution efficiency, and symbol error rate. The results demonstrate that, compared to the traditional DHR architecture, our method exhibits a clear advantage in terms of data privacy protection and reliability.

Nevertheless, although the proposed method enhances the data privacy security of the DHR architecture through the concept of structural encoding, the increasing complexity introduced by dynamic scheduling mechanisms raises new challenges. Issues related to architectural complexity, usability (practical deployability and configuration efficiency), and runtime performance of the DHR system warrant further investigation in future work. In particular, we will simplify the deployment and configuration of heterogeneous executors, optimize scheduling overhead, and improve the compatibility of the framework to enhance its practical application value.

Author Contributions: Conceptualization, S.S. and Y.S.; methodology, S.S.; software, Y.S.; validation, S.S. and Y.S.; formal analysis, N.F. and W.H.; resources, S.L. and Y.J.; writing—original draft preparation, S.S. and Y.S.; writing—review and editing, N.F. and Y.J.; supervision, Y.J.; funding, S.L. and Y.J.; acquisition, Y.J. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by the Jiangsu Key Development Planning Project (BE2023004-2), Wuxi University Research Start-up Fund for High-level Talents (2025r097). This work was supported by the Wuxi Key Laboratory of Artificial Intelligence and Security.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Acknowledgments: We would sincerely want to thank the peoples in the Nanjing Center of HPC China who are supported to do this work.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Liao, X.; Yang, S.; Ouyang, L. A deep reinforcement learning-based scheduling algorithm for mimic defense servers. *Symmetry* **2025**, *17*, 1960. [[CrossRef](#)]
2. Lv L.; Wu Z.; Zhang J. A VMD and LSTM based hybrid model of load forecasting for power grid security. *IEEE Trans. Ind. Inform.* **2021**, *18*, 6474–6482. [[CrossRef](#)]
3. Redhu A.; Choudhary P.; Srinivasan K. Deep learning-powered malware detection in cyberspace: A contemporary review. *Front. Phys.* **2024**, *12*, 6474–6482. [[CrossRef](#)]
4. Shi, Q.; Caleb, T.D.; Shao, S. Security of ADS-B and remote ID systems: Cyberattacks, detection techniques, and countermeasures. *Sensors* **2026**, *26*, 634. [[CrossRef](#)]
5. Wu J. Cyberspace endogenous safety and security. *Engineering* **2022**, *15*, 179–185. [[CrossRef](#)]
6. Wu J. Development paradigms of cyberspace endogenous safety and security. *Sci. China. Inf. Sci.* **2022**, *65*, 301–312. [[CrossRef](#)]
7. Sepczuk M.; Dynamic web application firewall detection supported by cyber mimic defense approach. *J. Netw. Comput. Appl.* **2023**, *213*, 103596. [[CrossRef](#)]
8. Wang, W.; Li, G.; Gai, K. Modelization and analysis of dynamic heterogeneous redundant system. *Concurr. Comput. Pract. Exp.* **2022**, *34*, 35–46. [[CrossRef](#)]
9. Ying, F.; Zhao, S.; Wang, J. A security information transmission method based on DHR for seafloor observation network. *Sensors* **2024**, *24*, 1147–1154. [[CrossRef](#)]
10. Hu, J.; Li, Y.; Li, Z. Unveiling the strategic defense mechanisms in dynamic heterogeneous redundancy architecture. *IEEE Trans. Netw. Serv. Manag.* **2024**, *35*, 154–163. [[CrossRef](#)]
11. Jia, H.Y.; Pan, Y.F.; Liu, W.H. Executive dynamic scheduling algorithm based on high-order heterogeneity. *J. Commun.* **2022**, *43*, 233–245.

12. Shao, S.S.; Gu, T.S.; Ji, Y.M. An active defense adjudication method based on adaptive anomaly sensing for mimic IoT. *IEEE Trans. Serv. Comput.* **2025**, *18*, 57–71. [[CrossRef](#)]
13. Yu, X.S.; Zhao, L.N.; Xu, Z.C. Mimic defense method for microservices based on dynamic heterogeneous redundancy. *Netw. Secur. Technol. Appl.* **2023**, *12*, 1–4.
14. Ma, H.L.; Yi, P.; Jiang, Y.M. Dynamic Heterogeneous Redundancy based Router Architecture with Mimic Defenses. *J. Cyber Secur.* **2017**, *2*, 29–42.
15. Song, K.; Liu, Q.R.; Wei, S. Endogenous security architecture of Ethernet switch based on mimic defense. *J. Commun.* **2020**, *41*, 18–26.
16. Xu, M.X.; Yuan, C.; Wang, Y.J. Mimic blockchain: A blockchain security solution. *J. Softw.* **2019**, *30*, 1681–1691.
17. Zhou, Q.L.; Ban, S.H.; Han, Y.J. Mimic defense authentication method for physical access control. *J. Commun.* **2020**, *41*, 80–87.
18. Wang, Z.P.; Hu, H.C.; Cheng, G.Z. Design of a DNS framework based on mimic security defense. *Acta Electron. Sin.* **2017**, *45*, 2705–2714.
19. Yuan, H.Y.; Guo, J.L.; Xu, M.Y. Research on honeypot technology based on endogenous security architecture. *Appl. Res. Comput.* **2023**, *40*, 1194–1201.
20. Pu, L.; Wu, J.; Ma, H. MimicCloudSim: An environment for modeling and simulation of mimic cloud service. *China Commun.* **2021**, *18*, 212–221. [[CrossRef](#)]
21. Pu, L.M. Research on Mimic Cloud Service Architecture and Key Technologies. Ph.D. Thesis, Strategic Support Force Information Engineering University, Zhengzhou, China, 2021.
22. Shi, Y.; Liu, S.; Shao, S. Mimic turbo compiled code structure for wireless communication systems. *IET Commun.* **2024**, *18*, 1089–1106. [[CrossRef](#)]
23. Zheng, Y.; Gu, T.; Liu, S. Highly reliable DHR-based polar compilation code communication method. *Comput. Netw.* **2024**, *252*, 110673. [[CrossRef](#)]
24. Pavani, K.; Sriramya, P. Reduction of complexity in asymmetric cryptography using RSA, RSA-CRT and novel N-prime RSA with different keys. In *Proceedings of the AIP Conference Proceedings*; AIP Publishing LLC: Melville, NY, USA, 2024; pp. 134–144.
25. Zheng, Y.; Li, Z.; Xu, X. Dynamic defenses in cyber security: Techniques, methods and challenges. *Digit. Commun. Netw.* **2022**, *8*, 422–435. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.