JIRBDAI
Journal of Institutional Research, Big Data Analytics and Innovation

An RNS-Enhanced DES Algorithm for Text Data Encryption in Legacy Industrial Control Systems

[1*]Aminat Omotayo Adebayo, [2]Tosho Abdulrauf

[1,2] Department of Computer Science, Faculty of Computing, Engineering and Technology, Al-Hikmah University, Ilorin, Kwara State, Nigeria.
[1*]ayakub@alhikmah.edu.ng, [2]autosho@alhikmah.edu.ng

ABSTRACT

Legacy Industrial Control Systems (ICS) continue to rely on lightweight cryptographic mechanisms due to hardware and computational constraints. Although the Data Encryption Standard (DES) remains attractive for such environments because of its simplicity. However, its susceptibility to modern cryptanalytic attacks affects its security effectiveness. This study presents the Data Encryption Residue Number System (DERNS) algorithm, an enhanced DES encryption scheme that integrates Residue Number System (RNS)-based key obfuscation together with pre-whitening and post-whitening operations into the DES encryption process. The development of the DERNS algorithm involved the analysis of the conventional DES structure, the integration of RNS-based key obfuscation techniques, the derivation of whitening masks from the obfuscated key, and the incorporation of pre-whitening and post-whitening operations into the DES encryption and decryption processes. The DERNS algorithm was implemented and evaluated against the standard DES algorithm using plaintext sizes ranging from 16 to 4096 bytes. Security performance was validated using entropy, avalanche effect, Strict Avalanche Criterion (SAC), Bit Independence Criterion (BIC), and key sensitivity, while the computational performance was evaluated using encryption time, decryption time, total execution time, throughput, and memory utilization. Experimental results show that DERNS maintained entropy values comparable to DES and close to the ideal value of 8 for larger plaintext sizes. The developed DERNS algorithm achieved improved avalanche characteristics, strong SAC and BIC compliance, and significantly enhanced key sensitivity. Performance evaluation further demonstrated superior scalability, with lower total execution times, higher throughput, and reduced memory utilization for moderate and large plaintext sizes compared with standard DES. The results indicate that the integration of RNS-based transformations enhances both the security and computational efficiency of DES without compromising its lightweight structure. The study concluded that DERNS provides a practical and scalable cryptographic enhancement suitable for legacy and resource-constrained industrial environments which requires secure and efficient data encryption.

KEYWORDS: Data Encryption Standard, Residue Number System, DERNS, Lightweight Cryptography, Industrial Control Systems.

I. INTRODUCTION

The rapid growth of interconnected digital systems has increased the importance of secure communication mechanisms across critical infrastructures, cloud systems, industrial automation platforms, and embedded computing environments (Bhamare et al., 2020; Nankya et al., 2023). Industrial Control Systems (ICS) continue to depend on lightweight and computationally efficient cryptographic solutions because many legacy infrastructures possess limited computational resources and cannot easily support computationally intensive modern encryption algorithms (Stouffer et al., 2023; Badawy et al., 2024). Despite the development of advanced cryptographic standards such as the Advanced Encryption Standard (AES), several

Volume 2, Issue 3, 2026
https://doi.org/10.5281/zenodo.21070162

legacy systems still rely on older encryption techniques, such as the Data Encryption Standard (DES) due to compatibility requirements, hardware limitations, and deployment constraints (Dhanda et al., 2020). The DES remains one of the most historically significant symmetric encryption algorithms due to its simplicity, structured Feistel design, and low computational complexity (Stallings, 2017; Al-Hazaimeh et al., 2023). However, DES suffers from several limitations including a relatively small key size and vulnerability to modern cryptanalytic attacks. These limitations have motivated several enhancement studies aimed at improving the security characteristics of DES (Shah et al., 2023; Alsuwaiedi & Rahma, 2023). At the same time, the Residue Number System (RNS) has attracted increasing attention in cryptographic research because of its parallel arithmetic capability, carry-free operations, modular representation properties, and suitability for high-speed computations (Omondi & Premkumar, 2007; Schoinianakis, 2020). RNS has been applied in various cryptographic and security-related applications which includes encryption systems, fault-tolerant systems, signal processing, and lightweight cryptographic architectures (Nykolaychuk et al., 2022; Yakymenko et al., 2024).

Existing studies have explored the integration of RNS with cryptographic systems for improved computational efficiency and enhanced security behaviour (Zawislak et al., 2022; Pujar et al., 2021). However, limited attention has been given to the integration of RNS-based transformations directly into the DES framework while simultaneously evaluating their impact on both security characteristics and computational performance in lightweight and legacy ICS environments. This study therefore designed an RNS-enhanced DES algorithm known as the Data Encryption Residue Number System (DERNS) algorithm. The designed DERNS algorithm integrated RNS-based key obfuscation together with pre-whitening and post-whitening operations into the conventional DES framework to improve both security and computational performance as well as preserves the underlying Feistel structure. The study presents the design and implementation of the DERNS algorithm and evaluated its effectiveness using security metrics including entropy, avalanche effect, Strict Avalanche Criterion (SAC), Bit Independence Criterion (BIC), and key sensitivity. It also evaluated performance metrics such as encryption time, decryption time, throughput, and memory utilization. The results provide insights into the suitability of the proposed approach for deployment in legacy and resource-constrained Industrial Control System environments.

II. RELATED WORKS

Several studies have explored different approaches for improving the security characteristics of conventional cryptographic algorithms, such as the DES algorithm and other lightweight symmetric encryption models. These studies have focused on enhancing diffusion properties, strengthening key dependency, improving computational efficiency, and adapting encryption systems for resource-constrained environments. A Novel Approach for Security Enhancement of DES proposed modifications to the internal DES structure, particularly the substitution process, to improve resistance against cryptanalytic attacks. The study demonstrated that enhancements to the S-box design can strengthen the security characteristics of DES (Shah et al., 2023). Similarly, A New Modified DES Algorithm Based on the Development of Binary Encryption Functions extended the conventional DES framework through the introduction of a new Xo operation to replace the traditional XOR function and employed multiple encryption keys instead of a single key. The results indicated improved security strength and increased encryption complexity, although at the expense of higher computational time (Alsuwaiedi & Rahma, 2023).

In another related work, Enhanced DES algorithm based on filtering and striding techniques proposed an improved DES model using filtering and striding operations to increase data security and encryption complexity. The study demonstrated that lightweight enhancement

Volume 2, Issue 3, 2026
https://doi.org/10.5281/zenodo.21070162

techniques can strengthen DES without introducing excessive computational overhead (Amorado et al., 2019). These studies collectively show that DES remains a flexible cryptographic structure capable of supporting additional enhancement layers despite its known limitations. Research efforts have also explored the application of the RNS in cryptography due to its inherent parallelism, modular arithmetic efficiency, and suitability for high-speed computation. Residue arithmetic systems in cryptography: a survey on modern security applications presented a comprehensive review of modern RNS applications in cryptographic systems and highlighted the computational advantages of modular arithmetic for secure data processing. The study emphasized that RNS-based techniques can improve arithmetic efficiency, fault tolerance, and parallel processing capabilities in encryption systems (Schoinianakis, 2020).

Similarly, Residue Number System Asymmetric Cryptoalgorithms explored the integration of RNS into asymmetric cryptographic models and demonstrated that residue-based transformations can improve computational efficiency while maintaining strong cryptographic properties (Nykolaychuk et al., 2022). In addition, Methods of crypto-stable symmetric encryption in the residual number system investigated symmetric encryption approaches based on residue number representations and showed that RNS can contribute to enhanced cryptographic stability and data transformation efficiency (Zawislak et al., 2022). More recently, Symmetric Encryption Algorithms in a Polynomial Residue Number System proposed residue-based symmetric encryption approaches aimed at improving encryption reliability and computational performance. Their work further demonstrated the suitability of RNS techniques for modern lightweight cryptographic applications (Yakymenko et al., 2024). These studies confirm that RNS possesses significant potential for enhancing encryption systems through modular arithmetic transformations and efficient parallel operations. The increasing demand for lightweight cryptographic systems has also motivated the development of encryption models optimized for resource-constrained environments such as IoT devices, embedded systems, and legacy ICS infrastructures. Lightweight cryptography: a solution to secure IoT discussed the growing importance of lightweight cryptography in low-resource environments and emphasized the need for encryption schemes that balance security strength with computational efficiency (Dhanda et al., 2020). Likewise, RBFK Cipher: A Randomized Butterfly Architecture-Based Lightweight Block Cipher proposed a lightweight block cipher architecture designed to improve security while minimizing computational overhead in edge computing environments (Rana et al., 2023).

Although previous studies have explored DES enhancement techniques, RNS-based cryptographic systems, and lightweight encryption architectures independently, limited attention has been given to integrating RNS-based transformations directly into the DES framework through lightweight operations such as key obfuscation and whitening mechanisms. Most existing RNS-based cryptographic studies focused on standalone encryption frameworks or asymmetric systems, while many DES enhancement approaches concentrated mainly on permutation or substitution modifications without leveraging modular arithmetic properties. Furthermore, limited studies have investigated the combined impact of RNS-based key transformations and whitening mechanisms on both the security characteristics and computational performance of DES in resource-constrained environments. This identified gap motivated the development of the DERNS algorithm proposed in this study, which integrated RNS-based key obfuscation together with pre-whitening and post-whitening operations into the traditional DES structure while preserving its lightweight Feistel structure.

Volume 2, Issue 3, 2026
https://doi.org/10.5281/zenodo.21070162

A. DES Architecture

DES is a symmetric block cipher that operates on 64-bit plaintext blocks using a 56-bit effective encryption key (Stallings, 2017). The algorithm performs 16 rounds of substitution and permutation operations by applying a different subkey in each round, generated from the main key using a key-scheduling algorithm (NIST FIPS Pub 81, 1980). To extend the usage of DES beyond a single 64-bit block, several modes of operation were standardized by NIST, including Electronic Codebook (ECB), Cipher Block Chaining (CBC), Cipher Feedback (CFB), Output Feedback (OFB), and Counter (CTR) (NIST FIPS Pub 81, 1980). Each mode has implications for error propagation, throughput, and parallelism, which makes the choice of mode critical in ICS environments where real-time communication and resilience are important. In all cases, plaintext is grouped into 64-bit blocks, and each block is enciphered using the secret key through initial permutation, 16 Feistel rounds, and a final permutation. Decryption follows the same structure, but applies the round keys in reverse order.

For any cipher, the most basic method of attack is brute force, which involves trying each possible key until the correct one is found. The key length determines the size of the key space and hence the feasibility of this type of attack. DES uses a 64-bit key field, but eight of those bits are used for parity checks, effectively limiting the key to 56 bits. This results in 256 possible keys (approximately 7.2×1016 candidates), which was considered large at the time of standardization but is now within reach of dedicated hardware and large distributed systems (Al-Hazaimeh et al., 2023). For the implementation and evaluation conducted in this study, DES was operated in ECB mode. ECB was selected to provide a deterministic baseline that enables direct comparison between conventional DES and the proposed DERNS without introducing additional state dependencies associated with feedback-based modes such as CBC, CFB, OFB, or CTR. This choice allows the observed security and performance differences to be attributed primarily to the proposed RNS-based enhancements rather than to the characteristics of a particular mode of operation. Nevertheless, ECB has well-known security limitations because identical plaintext blocks encrypted under the same key produce identical ciphertext blocks, potentially revealing patterns in the underlying data (NIST FIPS Pub 81, 1980). Consequently, ECB should be regarded in this work as an experimental baseline rather than a recommended mode for operational deployment. In practical ICS environments, more secure modes would generally be preferred to mitigate pattern leakage and enhance confidentiality. DES remains computationally efficient and suitable for resource-constrained environments but its limited key size and susceptibility to modern cryptanalytic attacks have motivated the development of enhancement techniques aimed at improving its security characteristics while preserving its lightweight structure.

Figure 1 illustrates the conventional DES encryption process. The plaintext undergoes an initial permutation before entering the 16-round Feistel network. In each round, the right half of the data block is processed through the DES round function using a generated subkey, after which the two halves are combined to produce the ciphertext through the final permutation. The proposed DERNS algorithm preserves the original DES Feistel architecture and key scheduling framework while introducing additional transformation layers based on RNS operations.

Volume 2, Issue 3, 2026
https://doi.org/10.5281/zenodo.21070162

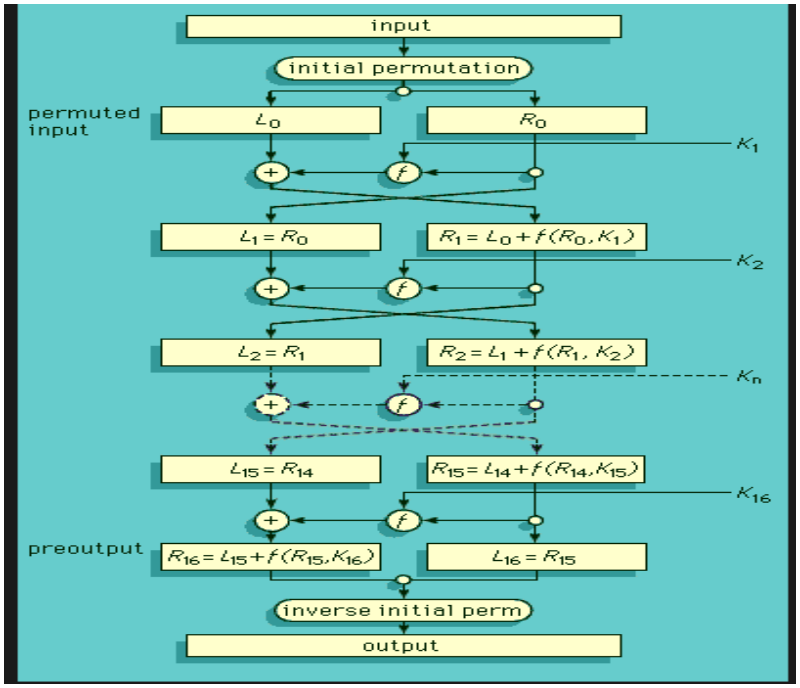

Figure 1: Flow Diagram for DES Algorithm (Source: Simmons, 2018. DES | Britannica, (2026).

B. RNS Fundamentals

The RNS is a non-weighted number representation system that expresses an integer using a set of pairwise coprime moduli (Babatunde et al., 2025). Due to its carry-free arithmetic properties, RNS supports parallel computations and efficient modular processing which makes it attractive for cryptographic applications (Zawislak et al., 2022). Let the selected moduli set be represented as:

$$M = \{m_1, m_2, m_3, ..., m_n\} \quad (1)$$

Where $\gcd(m_i, m_j) = 1$, for $i \neq j$. An integer X can be represented in residue form as:

$$X = (x_1, x_2, x_3, ..., x_n) \quad (2)$$

where

$$x_i = X \bmod m_i \quad (3)$$

The residue representation decomposes a large integer into smaller modular components that can be processed independently. In the developed DERNS algorithm, RNS is utilized for key obfuscation through modular decomposition of the DES encryption key. The resulting residue values are subsequently combined to generate whitening masks used during encryption and decryption operations. This approach introduces additional transformation complexity and also maintained computational efficiency. Furthermore, the key was reconstructed using the Chinese Remainder Theory (CRT). The CRT is used to determine a single integer from its remainders using a set of moduli. It has applications in various areas, such as digital signal processing and cryptography (Salifu, 2021; Singh & Agarwal, 2010). Therefore, the traditional CRT is defined as follows. For a modulus set $\{m_1, m_2, m_3, ..., m_n\}$ with the dynamic range $M = \prod_{i=1}^{n} m_i$, the residue number $(x_1, x_2, x_3, ..., x_n)$ can be converted into the decimal number X, as follows (Ibrahim & Gbolagade, 2019).

$$X = \sum_{i=1}^{n} \left| M_i \left| M_i^{-1} \right|_{m_i} x_i \right|_M \quad (4)$$

Volume 2, Issue 3, 2026
https://doi.org/10.5281/zenodo.21070162

Where

$M = \prod_{i=1}^{n} m_i$, $M_i = M/_{M\,i}$ and M_i^{-1} is the multiplicative inverse of M with respect to m_i (Salifu, 2021).

In this study, CRT was employed to carryout reverse conversion of residue to integer from a set of selected moduli (5, 7, 11).

III. METHODOLOGY

This study developed and evaluated the DERNS algorithm by incorporating the RNS-based key obfuscation together with pre-whitening and post-whitening operations into the traditional DES structure. The methodology involved analysing the conventional DES architecture, applying RNS principles to generate obfuscated key components, integrating the generated components into the DES encryption framework, and implementing the resulting algorithm using Python programming language. The developed DERNS algorithm was subsequently evaluated using selected security and performance metrics to determine its effectiveness in relation to the standard DES algorithm. The following subsections describe the design of the DERNS framework and algorithm, implementation of the DERNS algorithm, and the evaluation of the system.

A. The DERNS Architectural Framework

This framework integrates RNS operations into the standard DES algorithm in order to create a lightweight cryptographic system that is more resistant to differential and linear cryptanalysis which makes it suitable for resource-constrained ICS environments (Adebayo et al., 2024; NIST, 2022). The framework uses a block-based diagram, developed in Microsoft Visio (Microsoft 365). Visio was selected for its strength in producing clear, structured visuals of complex processes and interactions. The design process studied the DES encryption processes to see where RNS can be integrated and then mapped the core workflow (initial permutation, rounds, final permutation) of DES alongside RNS operations and also pinpoints the integration points in order to enhance encryption and decryption. Key visual elements include, curved rectangles for computational processes at each stage and directional arrows showing operational flow. This blueprint (illustrated in Figure 2) lays the groundwork for developing the DERNS algorithm.

B. Development of the DERNS Algorithm

The development of the DERNS algorithm followed a structured cryptographic algorithm design method. This method involves systematically modifying an existing encryption scheme by integrating additional security mechanism while preserving functional correctness and compatibility with the original cryptographic structure. The approach emphasizes logical decomposition of the encryption process into well-defined stages which ensures deterministic behaviour, reversibility, and functional correctness of the developed algorithm.

The method began with the analysis of the conventional DES algorithm to identify components suitable for enhancement. Based on this analysis, the RNS was incorporated as a complementary mechanism to strengthen key confidentiality and data protection. The DERNS algorithm was then developed through a sequence of logically defined stages. Each stage was designed to introduce a specific security improvement and also to preserve the operational integrity of the standard DES encryption framework. The framework therefore preserves the conventional DES Feistel network while introducing additional modular transformation layers.

Volume 2, Issue 3, 2026
https://doi.org/10.5281/zenodo.21070162

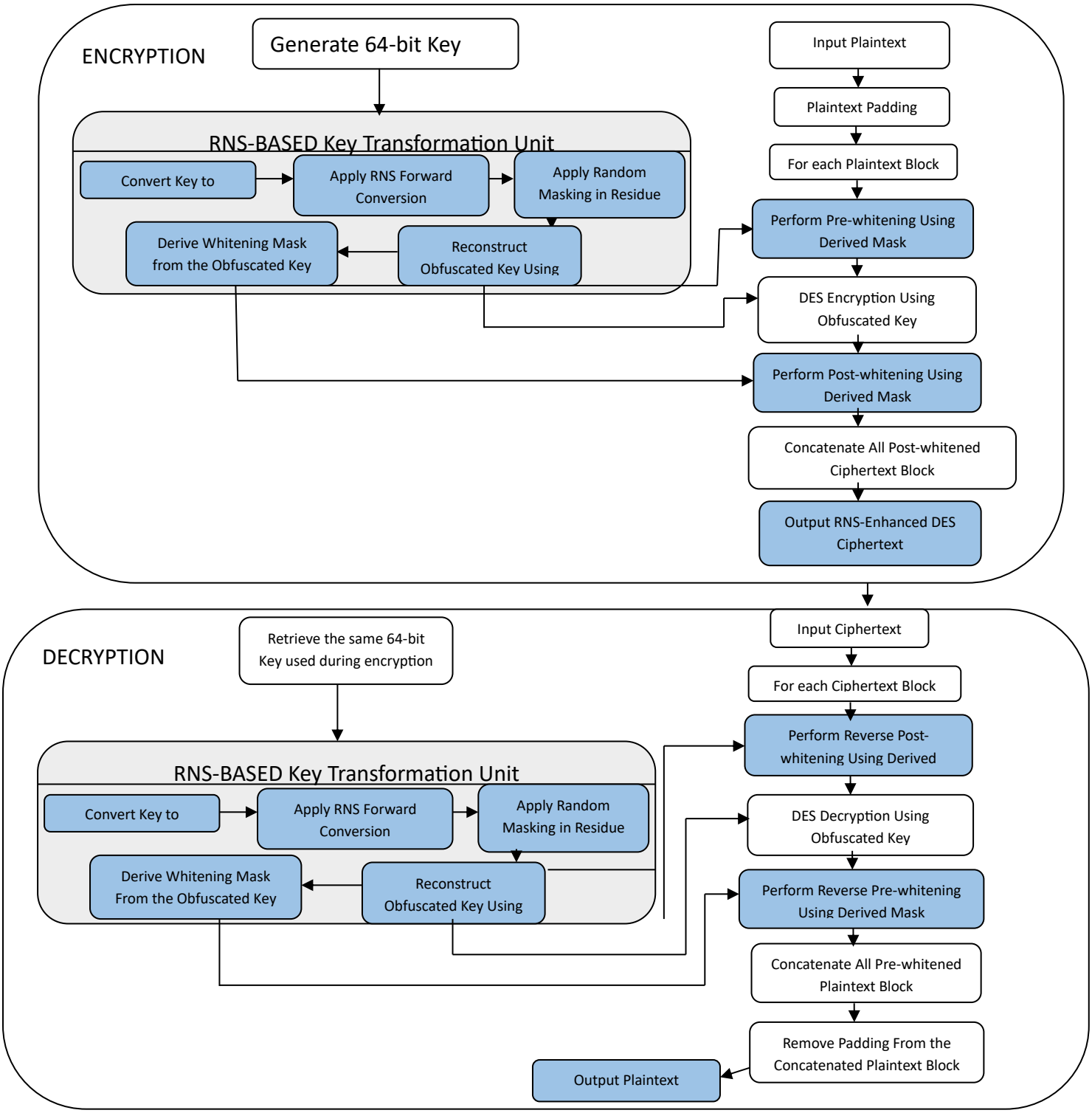

Figure 2: Framework of the Proposed DERNS Algorithm

The DERNS framework combines the conventional DES framework with RNS-derived transformation stages. The framework introduces four major enhancement layers: RNS-based key obfuscation, whitening mask generation, pre-whitening transformation and post-whitening transformation. The overall DERNS framework is illustrated in Figure 2.

The workflow begins with plaintext input and DES key generation. The encryption key is transformed through the RNS module to generate modular residue representations which are combined to form the obfuscated whitening key. A whitening mask was generated from the

Volume 2, Issue 3, 2026
https://doi.org/10.5281/zenodo.21070162

obfuscated key produced during the key obfuscation stage. This mask served as an additional layer of protection for plaintext data. Before encryption, the plaintext was pre-whitened by computing the exclusive OR (XOR) between the plaintext block and the generated whitening mask. This operation was applied as an additional pre-processing step before the standard DES encryption. After the standard DES encryption rounds are completed, the ciphertext undergoes a final post-whitening transformation before the final ciphertext output is produced.

Logical Developmental Stages of the DERNS Algorithm

(a). Stage 1: Analysis of the Conventional DES Structure

The first stage involved a detailed analysis of the conventional DES encryption structure to identify opportunities for enhancement. Particular attention was given to the handling of cryptographic keys and the role of Initial Permutation (IP) and Final Permutation (FP) operations. These operations were observed to primarily perform fixed bit rearrangements and do not significantly contribute to the cryptographic strength of the algorithm compared to the substitution and the key-mixing processes. Based on this, the study identified the need to introduce additional protective mechanism capable of improving confidentiality without altering the fundamental Feistel structure of the DES.

(b). Stage 2: Design of RNS-Based Key Obfuscation Mechanism

The second stage focused on strengthening key confidentiality through the application of RNS-based key obfuscation. In this stage, the original 64 bit DES key was converted to its RNS representation using a set of pairwise coprime moduli. Specifically, the moduli set (5,7,11) was chosen to decompose the original key into smaller residue components that could be independently manipulated without exposing the direct binary representation of the key.

$$RNS(K) = \{K \bmod m_1, K \bmod m_2, \ldots, K \bmod m_n\} \quad (5)$$

Where K is the integer representation of the encryption key, $m_1, m_2, \ldots, m_n$ is pairwise coprime moduli, K mod m_i is the residue of K with respect to modulus m_i, for i = 5, 7, 11 as used in this study. RNS (K) is the residue vector corresponding to the key K in the RNS domain.

Each residue components was subsequently masked using modular randomness to generate an obfuscated representation of the key. Let r_i denote the residue of the original DES key under modulus m_i. To obscure the residue representation, a masking value delta i (δ_i) was generated using a pseudo-random number generator as $\delta_i = \text{rand}(1, m_i - 1)$ (6)

where m_i is the corresponding modulus, and added to each residue and reduced modulo m_i. The resulting masked residue is given by:

$$r_i' = (r_i + \delta_i) \bmod m_i \quad (7)$$

where r_i' is the transformed (obfuscated) residue corresponding to the i^{th} modulus, r_i is the original residue of the key under modulus m_i. δ_i is the random masking value added to the residue r_i; $\delta_i \in \{1, \ldots, m_i - 1\}$, m_i is the i^{th} modulus in the pairwise coprime moduli set.

$(r_i + \delta_i) \bmod m_i$ computes the remainder obtained when the sum of the original residue and masked value is divided by m_i. i = 1, 2, …, n, where n is the number of moduli in the RNS moduli set.

The masked value, δ_i, is generated randomly to conceal the original residue and enhance key security against direct residue-based attacks. This process ensured that the binary representation of the key was modified and also maintained compatibility with the conventional DES algorithm. The selection of the coprime moduli set (5,7,11) was guided by considerations of computational efficiency and suitability for resource-constrained environment such as legacy ICS. The chosen moduli are relatively small integers that support efficient residue computation and reconstruction using the CRT with minimal arithmetic overhead. The dynamic range of the moduli set, defined as the product of the individual moduli provides sufficient

Volume 2, Issue 3, 2026
https://doi.org/10.5281/zenodo.21070162

capacity to represent the intermediate values generated during the key transformation process and also maintains low computational complexity. The RNS obfuscated key was reconstructed using

$K' = CRT(r_1', r_2', \ldots, r_n')$ (8)

In contrast, generalized moduli configurations such as sets of the form (2n -1, 2n, 2n + 1) are typically designed for high-speed arithmetic operations in specialized hardware architectures. Although, such generalized sets can offer advantages in large scale numerical computations, they may also introduce additional implementation complexity and increased computational overhead that are unnecessary for the key obfuscation objective of the proposed algorithm. Therefore, the moduli set (5,7,11) was selected as a lightweight and computationally efficient configuration that aligns with the design objective of preserving low resource requirements and also enhancing key confidentiality.

Following reconstruction of the obfuscated key, the standard DES key scheduling procedure was applied to generate the round subkeys required for encryption and decryption operations. The reconstructed 64-bit key was first subjected to the conventional DES permutation and splitting process which produced 28-bit halves. These halves were then cyclically shifted according to the predefined DES shift schedule for each encryption round. At each round of the encryption process, the shifted key halves were combined and compressed using the DES key permutation function to produce 48-bit round subkeys. This procedure was repeated across sixteen rounds and resulted in sixteen distinct subkeys that were sequentially applied during the Feistel encryption process. The use of the standard DES key scheduling mechanism ensured compatibility with the original DES structure and also allowed the security enhancement introduced by the RNS-based key obfuscation to be consistently propagated throughout the encryption rounds.

Through this structured design process, the RNS-based key obfuscation mechanism provided an additional layer of protection for the cryptographic key and also maintained the deterministic behaviour, reversibility, and operational compatibility with the conventional DES encryption framework.

(c). Stage 3: Integration of RNS-Derived Whitening Mechanism

In this stage, an RNS-derived whitening mechanism was introduced to enhance data protection prior to encryption. A whitening mask was generated using the formula:

$M_i = (B + i \cdot r_i \bmod n + r_i^2 \bmod n) \bmod 256$ (9)

Where:

$$B = \sum_{j=1}^{n} r_j \quad (10)$$

where M_i is the ith byte of the whitening mask, B is sum of all RNS residues, r_j is the residue under modulus m_j , n is the number of residues (3 in this study: moduli 5,7,11), i is the byte position of the mask (0 to 7), mod 256 constrains each mask value to the range 0 - 255, which corresponds to a single byte.

Before encryption, the plaintext was pre-whitened by computing the exclusive OR (XOR) between the plaintext block and the generated whitening mask as follows:

$P_i' = P_i \oplus M_i \oplus ((31i) \bmod 256)$ (11)

Where P_i' is the transformed (masked) plaintext byte at position I, P_i is the original plaintext byte at position I, M_i is the i^{th} dynamically generated mask byte, $\oplus$ is the bitwise Exclusive-OR (XOR) operation, i is the byte position index within the plaintext block, (31i) mod 256 is a position-dependent offset that generates a byte value in the range 0 - 255, which introduced additional diffusion and variability into the masking process. mod 256 ensures that the

Volume 2, Issue 3, 2026
https://doi.org/10.5281/zenodo.21070162

computed offset remains within the valid byte range. P_i' represents the final masked plaintext byte that is forwarded to the subsequent encryption stage.

The combination of the dynamic mask M_i and the position-dependent term (31i) mod 256 enhances confusion and diffusion by making identical plaintext bytes produce different masked outputs at different positions. This operation was applied as an additional preprocessing step before the standard DES encryption. The deterministic derivation of the whitening mask ensured consistent and repeatable behaviour between encryption and decryption processes.

(d). Stage 4: Design of the Modified Encryption Process

The fourth stage involved defining the modified encryption sequence using pre-whitened plaintext and the reconstructed obfuscated key. In this stage, the core encryption operation retained the standard DES structure which includes the 16-round Feistel network responsible for substitution, permutation and key mixing operations. This was computed using:

$$C = DES\ K'(P') \quad (12)$$

Where C is the resulting ciphertext produced after encryption, DES K′( ) is the DES encryption operation using the transformed key K′, K′ is the transformed encryption key generated through the DERNS key processing stage and P′ is the transformed (masked) plaintext obtained after the dynamic masking process.

The transformed plaintext P′ was encrypted using DES with the transformed key K′, which produces the ciphertext C. This additional preprocessing stage increased the complexity of the encryption process and enhances resistance against cryptanalytic attacks and also preserves the backward compatibility of the system that still rely on DES such as legacy ICS.

(e). Stage 5: Design of the Post-Whitening Mechanism

Following the encryption process, a post-whitening mechanism was incorporated to provide additional protection for the generated ciphertext. This stage involved computing the XOR between the ciphertext and the same whitening mask used during the pre-whitening stage. The post-whitening operation was computed using

$$C_i' = C_i \oplus M_i \oplus ((17_i) \bmod 256) \quad (13)$$

Where C_i' is the final transformed ciphertext byte at position I, C_i is the original ciphertext byte produced by the DES encryption process, M_i is the dynamically generated mask byte corresponding to position I, $\oplus$ is the bitwise Exclusive-OR (XOR) operation, i is the byte index within the ciphertext block or stream and (17i) mod 256 is a deterministic position-dependent byte value used to introduce additional diffusion and randomness across ciphertext positions. mod 256 ensures the value remains within the valid 8-bit range [0, 255] while C_i' represents the final obfuscated ciphertext byte intended for transmission or storage. This post-whitening operation was applied as an additional output masking step after DES encryption. By introducing this extra layer at the output stage, the design improved resistance to pattern analysis and also maintained computational efficiency and structural simplicity.

(f). Stage 6: Design of the Decryption Process

The final stage involved designing the decryption procedure as the reverse sequence of the encryption process to ensure accurate recovery of the original plaintext. The ciphertext was first subjected to reversal of the post-whitening operation using the same whitening mask. The resulting intermediate data was then decrypted using the DES decryption mechanism with the reconstructed obfuscated key. Finally, the original plaintext was recovered by reversing the pre-whitening operation through another XOR computation with the identical mask. The deterministic generation of the whitening mask ensured symmetric behaviour between encryption and decryption operations which thereby preserves data integrity and correctness of the overall system. The Decryption equations are:

Volume 2, Issue 3, 2026
https://doi.org/10.5281/zenodo.21070162

(a). Remove post-whitening:

$$C = C' \oplus M \oplus ((17i) \bmod 256) \quad (14)$$

(b). DES Decrypt:

$$P' = DES^{-1}_{K'}(C) \quad (15)$$

(c). Remove pre-whitening:

$$P = P' \oplus M \oplus ((31i) \bmod 256) \quad (16)$$

Following the structured development of the DERNS algorithm through the defined design stages, the final encryption and decryption procedure were formalized into algorithmic representations. These algorithms describe the logical sequence of operations required to perform secure text data encryption and decryption using the developed DERNS framework.

Presentation of the DERNS Algorithm

The encryption phase for the DERNS algorithm considered integrating RNS into the DES operation as stated in Algorithm 1.

Table 1: DERNS Encryption Algorithm

Algorithm 1: DERNS Encryption Algorithm	
Step 1:	*Start*
Step 2:	*Input Plaintext.*
Step 3:	*Accept or generate a 64-bit DES secret key and convert it to its integer representation.*
Step 4:	*Transform the key using RNS with the moduli set [5,7,11] to produce a vector of residue values.*
Step 5:	*Reconstruct the obfuscated key by applying the CRT to the generated residues.*
Step 6:	*Derive an (8 bytes) whitening mask from the obfuscated key to be used for XOR-based whitening operations.*
Step 7:	*Apply PKCS#5 padding to the plaintext to ensure that its length is a multiple of 8 bytes (64 bits).*
Step 8:	*For each 64-bit (8-byte) plaintext block:* *8.1. Perform pre-whitening by XORing the padded plaintext block with the whitening mask.* *8.2. Encrypt the whitened block using standard DES encryption function operating in ECB mode.* *8.3. Perform post-whitening by XORing the resulting ciphertext with the same whitening mask.*
Step 9:	*Concatenate all post-whitened ciphertext blocks to form the complete ciphertext.*
Step 10:	*Return the final DERNS ciphertext in hexadecimal format.*
Step 11:	*Stop.*

The decryption phase for the DERNS algorithm considered integrating RNS into the DES operation as stated in Algorithm 2.

Table 2: DERNS Decryption Algorithm

Algorithm 2: DERNS Decryption Algorithm	
Step 1:	*Start*
Step 2:	*Input the ciphertext in hexadecimal format.*
Step 3:	*Convert the ciphertext from hexadecimal representation into binary data (bytes)*
Step 4:	*Retrieve the same 64-bit DES secret key used during encryption and convert it to an integer representation.*
Step 5:	*Transform the key using RNS with the same moduli set [5,7,11], to generate the corresponding residue values.*
Step 6:	*Reconstruct the obfuscated key by applying the CRT.*
Step 7:	*Derive the same 8-byte whitening mask from the obfuscated key for reversing the whitening operations.*
Step 8:	*For each 64-bit (8-byte) ciphertext block:* *8.1. Perform reverse post-whitening by XORing the ciphertext block with the whitening mask.*

Volume 2, Issue 3, 2026
https://doi.org/10.5281/zenodo.21070162

8.2. Decrypt the resulting block using the standard DES decryption function operating in ECB mode.
8.3. Perform reverse pre-whitening by XORing the decrypted block with the whitening mask.
Step 9: *Concatenate all resulting plaintext blocks.*
Step 10: *Remove the PKCS#5 padding from the reconstructed plaintext.*
Step 11: *Return the final plaintext data.*
Step 12: *Stop*

The whitening operations are applied externally to the DES core, ensuring that the internal structure of DES remains unchanged.

A. Experimental Setup for Implementation

The algorithm was implemented using Python (Version 3.12) on a Windows 10 Professional 64-bit platform. The experiments were conducted using an Intel Core i3-7020U CPU @ 2.30 GHz with 8 GB of RAM. Although, this test hardware is significantly more powerful than typical legacy ICS field devices (which often operate with 10–100 MHz CPUs and only 4–64 KB of RAM) (Sorescu et al., 2025), this setup was chosen to ensure precise, reproducible measurement of the performance metrics of the algorithm. The key is not that the test machine matches legacy hardware, but that the results, specifically the measured execution time, memory footprint, and throughput, allow us to determine whether the DERNS algorithm is lightweight enough to function within the strict resource constraint of legacy PLCs and RTUs (Sorescu et al., 2025). In other words, a powerful machine was used to get accurate data, and to check if that data shows that the algorithm is small and fast enough for constrained devices. For the purposes of experimental evaluation, DES was implemented using ECB mode. The selection of ECB was intended to provide a simple and deterministic baseline for assessing the cryptographic behaviour and performance characteristics of the proposed DERNS framework without introducing additional dependencies associated with feedback-based modes of operation. However, ECB is known to reveal structural patterns in plaintext data because identical plaintext blocks produce identical ciphertext blocks under the same key. Consequently, ECB is generally not recommended for practical deployments involving sensitive information. The objective of this study was therefore to evaluate the effectiveness of the proposed DERNS enhancements relative to a conventional DES implementation under controlled conditions rather than to recommend ECB for operational use.

B. System Evaluation

The evaluation of the developed DERNS algorithm was carried out to assess its cryptographic strength and computational efficiency under controlled experimental conditions. Performance evaluation is a foundational phase in cryptographic system development because the security of an encryption algorithm is determined not only by its architectural design but also by its statistical behaviour and operational efficiency (Korobeinikova & Kopach, 2024; Thabit et al., 2021). Metrics such as the avalanche effect, entropy, SAC, BIC, key sensitivity, execution time, throughput and memory usage were used to evaluate the DERNS algorithm because they are established standard in the literature to determine the effectiveness and feasibility of cryptographic algorithms in resource-constrained environments, such as legacy ICS devices (Sorescu et al., 2025; Thabit et al., 2021). The evaluation of the DERNS algorithm was divided into two main categories: security evaluation and performance evaluation. Under security evaluation, the metrics were used are avalanche effect, entropy, Strict Avalanche Criterion (SAC), and Bit Independence Criterion (BIC) while under performance evaluation, the metrics considered are encryption time, decryption time, throughput, and memory utilisation.

Volume 2, Issue 3, 2026
https://doi.org/10.5281/zenodo.21070162

IV. RESULTS AND DISCUSSION

A. Security Analysis of DERNS and DES Algorithm

The security performance of the proposed DERNS algorithm was evaluated against the conventional DES algorithm using entropy, avalanche effect, Strict Avalanche Criterion (SAC), Bit Independence Criterion (BIC), and key sensitivity across plaintext sizes ranging from 16 to 4096 bytes. The results are presented in Table 1.

Table 1: Security Metrics Comparison of DERNS and DES Across Varying Plaintext Sizes.

Plaintext Size (B)	DERNS Entropy	DES Entropy	DERNS Avalanche (%)	DES Avalanche (%)	DERNS SAC (%)	DES SAC (%)	DERNS BIC (%)	DES BIC (%)	DERNS Key Sensitivity (%)	DES Key Sensitivity (%)
16	4.50	4.50	48.44	34.38	51.54	51.20	98.43	98.43	51.56	0.00
64	5.89	5.90	57.81	54.69	49.10	49.76	98.02	98.30	48.44	0.00
256	7.17	7.28	62.50	46.88	49.88	50.71	98.13	98.34	49.57	0.00
512	7.63	7.58	45.31	43.75	51.20	50.42	98.27	98.15	49.50	0.00
1024	7.80	7.86	53.12	56.25	49.76	49.90	98.08	98.20	49.65	0.00
2048	7.89	7.93	46.88	43.75	49.76	49.37	98.26	98.48	50.09	0.00
4096	7.95	7.95	46.88	46.88	49.93	51.29	98.15	98.41	49.56	0.00

As shown in Table 1, both algorithms exhibited increasing entropy as plaintext size increased, reaching near-ideal values at larger plaintext sizes. DERNS achieved entropy values between 7.80 and 7.95 for plaintext sizes of 1024 to 4096 bytes, while DES achieved comparable values between 7.86 and 7.95. This suggests that the proposed modifications do not degrade ciphertext randomness and maintain the statistical characteristics expected of a secure block cipher. The diffusion characteristics of DERNS were evaluated using the avalanche effect, SAC, and BIC metrics. Avalanche values remained within the acceptable range for secure encryption and were generally higher for DERNS at smaller and moderate plaintext sizes. For example, at 256 bytes, DERNS achieved an avalanche effect of 62.50% compared to 46.88% for DES. The enhanced diffusion can be attributed to the additional transformations introduced by the whitening stage, which increase the propagation of plaintext variations throughout the encryption process. Similarly, SAC values for both algorithms remained close to the theoretical ideal of 50%, ranging from 49.10% to 51.54% for DERNS and 49.37% to 51.29% for DES. The consistency of these values demonstrates that the proposed modifications preserved balanced bit propagation and do not introduce observable bias into the ciphertext generation process. BIC values also remained consistently high, exceeding 98% for both algorithms across all plaintext sizes, indicating strong independence among ciphertext bits and resistance to statistical dependencies that may facilitate cryptanalysis. In contrast, DES produced no measurable variation under the adopted evaluation approach. This behaviour suggests that the RNS-based obfuscation and whitening mechanisms strengthen the dependence of ciphertext generation on the encryption key which thereby improved resistance to key-related attacks. Finally, the security evaluation demonstrates that DERNS maintains the desirable statistical properties of DES, provides enhanced key sensitivity and competitive diffusion characteristics.

B. Performance Analysis of DERNS and DES Algorithm

The computational performance of both algorithms was evaluated using encryption time, decryption time, total execution time, throughput, and memory consumption. The results obtained under identical experimental conditions are presented in Table 2.

Volume 2, Issue 3, 2026
https://doi.org/10.5281/zenodo.21070162

Table 2: Performance Comparison of DERNS and DES Across Varying Plaintext Sizes

Plaintext Size (B)	DERNS Encryption Time (ms)	DES Encryption Time (ms)	DERNS Decryption Time (ms)	DES Decryption Time (ms)	DERNS Total Time (ms)	DES Total Time (ms)	DERNS Throughput (KB/s)	DES Throughput (KB/s)	DERNS Memory (KB)	DES Memory (KB)
16	0.29	0.15	0.12	0.12	0.41	0.27	38.46	57.85	1.79	2.49
64	0.17	0.30	0.11	0.28	0.28	0.58	225.71	108.43	1.83	2.75
256	0.28	0.95	0.20	1.01	0.48	1.96	522.58	127.48	3.87	6.15
512	0.42	1.80	0.34	1.79	0.76	3.59	657.29	139.36	6.77	11.22
1024	0.69	9.17	0.62	8.28	1.31	17.45	763.71	57.32	13.30	21.34
2048	1.93	8.37	1.20	6.96	3.13	15.33	639.08	130.48	26.45	41.40
4096	2.40	13.98	2.26	15.48	4.66	29.46	858.61	135.76	48.93	81.64

The results in Table 2 indicate for very small plaintext sizes (16 bytes), DES exhibited lower execution times due to its simpler structure and absence of additional preprocessing operations. However, as plaintext size increased, DERNS demonstrated superior performance across most performance metrics. At 4096 bytes, DERNS achieved encryption and decryption times of 2.40 ms and 2.26 ms, respectively, compared to 13.98 ms and 15.48 ms for DES. Consequently, the total execution time of DERNS remained significantly lower than that of DES for moderate and large plaintext sizes. The throughput results further reinforce this observation. DES achieved a higher throughput at 16 bytes, while DERNS outperformed DES for all larger plaintext sizes. At 4096 bytes, DERNS achieved a throughput of 858.61 KB/s compared to 135.76 KB/s for DES which represents a substantial improvement in data processing efficiency. This indicates that the DERNS approach scales effectively as data volume increases. Memory consumption results reveal a similar trend. DERNS consistently required less memory than DES across all evaluated plaintext sizes. For example, at 4096 bytes, DERNS consumed 48.93 KB compared to 81.64 KB for DES. The reduced memory requirement revealed an improved resource utilization, which is particularly advantageous for deployment in resource-constrained environments. The combined performance results indicate that although DERNS introduces a small overhead for very small plaintext sizes, it achieved superior scalability and resource efficiency as the amount of processed data increases.

C. Comparative Assessment and Security-Performance Trade-off

The comparative evaluation demonstrates that DERNS successfully balances security enhancement with computational efficiency. Security metrics such as entropy, SAC, and BIC remained comparable to those of DES which confirms that the integration of RNS-based operations does not compromise the fundamental cryptographic characteristics of the underlying cipher. At the same time, DERNS exhibited stronger key sensitivity and competitive avalanche behaviour which indicates an improved responsiveness to both plaintext and key modifications. From a performance perspective, DES maintained a slight advantage only for very small plaintext sizes. However, this advantage diminished rapidly as plaintext size increased. DERNS consistently achieved lower execution times, higher throughput, and reduced memory consumption for moderate and large datasets which demonstrates superior scalability. These findings suggest that the proposed DERNS algorithm is particularly suitable for legacy ICS, where secure communication must be achieved under constrained computational and memory resources. By preserving the security characteristics of DES while improving key sensitivity and resource efficiency, DERNS provides a practical enhancement to conventional DES-based protection mechanisms.

V. CONCLUSION

Based on the results obtained from both security and performance evaluations, it can be concluded that the DERNS algorithm successfully enhances the traditional DES framework

Volume 2, Issue 3, 2026
https://doi.org/10.5281/zenodo.21070162

without compromising its core cryptographic structure. The integration of RNS into DES introduces a more dynamic and flexible transformation process which improves the responsiveness of the algorithm to both plaintext and key variations. This is evident in the improved avalanche characteristics, strong SAC and BIC performance, and consistent key sensitivity observed in the results. From a performance perspective, although the inclusion of RNS-based operations introduces additional computational steps, the overall impact is well managed. As the data size increases, DERNS demonstrated better scalability, improved throughput and reduced execution time compared to standard DES. Additionally, its relatively lower memory consumption, combined with competitive execution time at moderate and large data sizes, indicates that the developed algorithm exhibits lightweight characteristics which makes it suitable for deployment in environments with limited computational resources. Finally, the study establishes that DERNS offers a practical balance between enhanced security features and acceptable performance overhead. This makes it a viable alternative to standard DES, particularly in legacy systems where lightweight yet effective cryptographic solutions are required.

Future work can explore the parallel implementation of the DERNS algorithm by leveraging on the inherent parallelism of the RNS to further improve computational efficiency, particularly through hardware-based approaches. This will be particularly beneficial in industrial and embedded systems where performance and power efficiency are critical. Also, future work can investigate the integration of DERNS with more secure block cipher modes, including CBC, CFB, OFB, and CTR, to evaluate its performance and security characteristics under deployment-oriented conditions. In addition, further investigation into the optimization of RNS parameters, especially the selection and configuration of moduli set, can enhance both the performance and security characteristics of the model. The application of the RNS-based enhancement can also be extended to more modern cryptographic algorithms beyond DES to evaluate its effectiveness in contemporary encryption systems. Moreover, the robustness of the DERNS algorithm can be examined against advanced cryptanalytic attacks, including differential, linear, and side-channel attacks in order to provide deeper insight into its security strength. Practical deployment considerations can be addressed through real-world system integration and testing, particularly in legacy ICS environments in order to evaluate operational performance under realistic conditions. Further refinement of key sensitivity evaluation models is also recommended to provide broader validation of ciphertext responsiveness under different measurement frameworks. Additionally, hardware-based implementation of the DERNS algorithm on platforms such as Field-Programmable Gate Array (FPGA) or embedded systems can be explored to assess real-time performance, power consumption, and deployment feasibility. Finally, future research can explore applying the DERNS enhancement approach to more recent lightweight cryptographic algorithms. This will help determine whether the RNS-based techniques used in this study can also improve the security strength and computational efficiency of newer encryption methods, beyond the traditional DES framework.

REFERENCES

Adebayo, A., Adeniyi, A. E., Ajao, J., Isiaka, R., Gbolagade, K., & Abdulsalam, S. (2024). Development of a Multi-Level Data Encryption Standard with Residue Number System for Data Security. In *Conference Organising Committee* (Vol. 14).

Alsuwaiedi, H. K. A., & Rahma, A. M. S. (2023). A new modified DES algorithm based on the development of binary encryption functions. *Journal of King Saud University - Computer and Information Sciences*, *35*(8). https://doi.org/10.1016/j.jksuci.2023.101716

Al-Hazaimeh, O. M., Al-Shannaq, M. A., Bawaneh, M. J., & Nahar, K. M. O. (2023). Analytical approach for the data encryption standard algorithm. International Journal of Interactive Mobile Technologies.

https://doi.org/10.5281/zenodo.21070162

Amorado, R. V., Sison, A. M., & Medina, R. P. (2019, March). Enhanced data encryption standard (DES) algorithm based on filtering and striding techniques. In *Proceedings of the 2nd International Conference on Information Science and Systems* (pp. 252-256).

Babatunde A N, Balogun B F, Lawal A O, Akuji I I, Kolawole O F, Isiaka O S, Oke, A. A., Kadri A F, Shuaib B M, Sulaiman, H. B., 10, A. T., & 11. (2025). Application of Residue Number System in Information Security: A Systematic Review. In *Special Issue on Computing & Advances in Information Technology* (Vol. 11, Issue 1). https://doi.org/10.56471/

Badawy, M., Sherief, N. H., & Abdel-Hamid, A. A. (2024). Legacy ICS cybersecurity assessment using hybrid threat modelling-An oil and gas sector case study. Applied Sciences, 14(18), 8398.

Bhamare, D., Zolanvari, M., Erbad, A., Jain, R., Khan, K., & Meskin, N. (2020). Cybersecurity for industrial control systems: A survey. Computers and Security.

Buchanan, W. J., & Maglaras, L. (2023, June). Review of the nist light-weight cryptography finalists. In *2023 19th International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT)* (pp. 469-474). IEEE.

Dhanda, S. S., Singh, B., & Jindal, P. (2020). Lightweight cryptography: A solution to secure IoT. Wireless Personal Communications.

Ibrahim, M. B., & Gbolagade, K. A. (2019). Performance comparison of Huffman coding and Lempel-Ziv-Welch text compression algorithms with Chinese remainder theorem. University of Pitesti Scientific Bulletin Series: Electronics and Computer Science, 19(2), 7–12.

Korobeinikova, T., & Kopach, A. (2024, May). Analysis of Modern Encryption Algorithms and their Impact on the Security and Effectiveness of Information Protection. In *Sworld-Us Conference proceedings* (No. usc24-00, pp. 35-40).

Nankya, M., Chataut, R., & Akl, R. (2023). Securing Industrial Control Systems: Components, Cyber Threats, and Machine Learning-Driven Defense Strategies. *Sensors (Basel, Switzerland)*, *23*(21), 8840. https://doi.org/10.3390/s23218840

Nguyen, H. P., & Chen, Y. (2024). Lightweight, post-quantum secure cryptography based on Ascon. Electronics, 13(22), 4550. https://doi.org/10.3390/electronics13224550

NIST. (1999). Data Encryption Standard (DES). Federal Information Processing Standards Publication.

NIST, (2022). Towards a New Lightweight Cryptography Standard. *ETSI Security Conference, 2022*.https://csrc.nist.gov/csrc/media/Presentations/2022/towards-a-new-lightweight-cryptography-standard/images-media/Talk-Etsi-2022-Meltem-Oct2022.pdf

Nykolaychuk, Y. M., Yakymenko, I. Z., Vozna, N. Y., & Kasianchuk, M. M. (2022). Residue number system asymmetric cryptoalgorithms. Cybernetics and Systems Analysis.

Omondi, A., & Premkumar, B. (2007). Residue Number Systems: Theory and Implementation.

Pujar, S. R., Poonja, A. R., & Aithal, G. (2019, May). Exponential Cipher Based on Residue Number System and Its Application to Image Security. In *International Conference on Artificial Intelligence and Data Engineering* (pp. 529-541). Singapore: Springer Nature Singapore.

Radhakrishnan, I., Jadon, S., & Honnavalli, P. B. (2024). Efficiency and security evaluation of lightweight cryptographic algorithms for resource-constrained IoT devices. *Sensors*, *24*(12), 4008.

Rana, S., Mondal, M. R. H., & Kamruzzaman, J. (2023). RBFK cipher: a randomized butterfly architecture-based lightweight block cipher for IoT devices in the edge computing environment. *Cybersecurity*, *6*(1), 3

Salifu, A. M. (2021). New reverse conversion for four-moduli set and five-moduli set. *Journal of Computer and Communications*, *9*(4), 57-66.

Schoinianakis, D. (2020). Residue arithmetic systems in cryptography: A survey on modern security applications. Journal of Cryptographic Engineering.

Shah, D., Shah, T., Jamal, S. S., Hazzazi, M. M., Aljaedi, A., & Alharbi, A. R. (2023). A novel approach for security enhancement of data encryption standard. Computers, Materials and Continua.

Simmons, G. J. (2018). Data encryption standard. Encyclopedia Britannica Inc. Retrieved from https://www. britannica. com/topic/Data-Encryption-Standard. Last visited January, 26, 2018.

Singh, S., & Agarwal, G. (2010). Use of the Chinese remainder theorem to generate random numbers for cryptography. International Journal of Applied Engineering Research, 1(1), 165–174.

Sorescu, T. G., Chiriac, V. M., Stoica, M. A., Comsa, C. R., Soroaga, I. G., & Contac, A. (2025). Comparative Performance Analysis of Lightweight Cryptographic Algorithms on Resource-Constrained IoT Platforms. *Sensors*, *25*(18), 5887.

Stallings, W. (2017). Cryptography and network security (7th ed.). Pearson.

Stouffer, K., Pease, M., Tang, C., Zimmerman, T., Pillitteri, V., Lightman, S., & Thompson, M. (2023). Guide to operational technology security. National Institute of Standards and Technology.

Thabit, F., Alhomdy, S., & Jagtap, S. (2021). Security analysis and performance evaluation of a new lightweight cryptographic algorithm for cloud computing. *Global Transitions Proceedings*, *2*(1), 100-110.

Wiemers, A., & Mittmann, J. (2023). Improving recent side-channel attacks against the DES key schedule. *Journal of Cryptographic Engineering*, *13*(1), 1–17. https://doi.org/10.1007/S13389-021-00279-2/TABLES/5

Yakymenko, I., Karpinski, M., Shevchuk, R., & Kasianchuk, M. (2024). Symmetric encryption algorithms in a polynomial residue number system. *Journal of Applied Mathematics*, *2024*(1), 4894415.

Zawislak, S., Kasianchuk, M., Iakymenko, I., & Jancarczyk, D. (2022). Methods of crypto-stable symmetric encryption in the residual number system. Procedia Computer Science.

https://doi.org/10.5281/zenodo.21070162