© AUG 2026 | IRE Journals | Volume 10 Issue 2 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV10I2-1722723

Development of a Secure Triple Data Encryption Algorithm (TDESA) Using a Key-Dependent Double XOR Transformation

S. F. ILO[1], NNAMDI H. I.[2], NWANEBU O. T.[3]
[1, 2, 3]*Department of Computer Engineering, Michael Okpara University of Agriculture, Umudike, Abia State, Nigeria*

Abstract- ***The rapid expansion of digital communication infrastructure has intensified the demand for encryption algorithms that are simultaneously efficient and resistant to modern cryptanalytic techniques. Conventional symmetric cryptographic systems, including the Data Encryption Standard (DES) and its triple-application successor (3DES), continue to face challenges related to computational overhead and diminishing resistance against brute-force and statistical attacks as computing power increases. This study presents an improved Secure Triple Data Encryption Algorithm (TDESA) that embeds a key-dependent double transformation layer, comprising a bitwise Exclusive-OR (XOR) operation followed by a key-controlled cyclic rotation and a second XOR operation, between two full encryption stages. Unlike a naive double-XOR construction, in which two successive XOR operations with fixed keys collapse algebraically into a single XOR and therefore add no security, the rotation-dependent design proposed here breaks this linearity and ensures that the intermediate transformation cannot be reduced to an equivalent single-key operation. The mathematical structure, algorithmic procedure, and system architecture of the proposed model are presented in detail, together with a security justification of the redesigned transformation layer. The algorithm was implemented and evaluated in MATLAB R2023b using plaintext samples of 1 KB, 10 KB, 100 KB, 500 KB, and 1 MB, with encryption time, decryption time, memory utilisation, and avalanche effect measured against DES, 3DES, and AES-128 baselines. Results indicate that the proposed TDESA achieves an average encryption time of approximately 34.0 milliseconds for a 1 MB file, positioning it between AES-128 (29.1 ms) and 3DES (60.2 ms), while exhibiting an avalanche effect of approximately 51.2 percent, comparable to AES. The proposed model is suitable for secure communication systems, cloud computing environments, and resource-constrained Internet-of-Things (IoT) applications where a balance between confidentiality strength and computational economy is required.***

Keywords: symmetric cryptography; triple data encryption standard; key-dependent xor transformation; cyclic rotation; cryptographic diffusion; internet of things security.

I. INTRODUCTION

The continuous growth of digital communication systems has resulted in an unprecedented volume of sensitive information being transmitted across public and private networks on a daily basis. Financial transactions, medical records, government communications, and proprietary organisational data are routinely exchanged through digital platforms, making the confidentiality, integrity, and authenticity of such data a critical concern in contemporary computing environments (Stallings, 2017). The increasing sophistication of cyber-attacks, ranging from passive eavesdropping to active man-in-the-middle and brute-force attacks, has placed growing pressure on cryptographic systems to remain both secure and computationally practical.

Cryptography provides the principal set of techniques for securing information by transforming intelligible plaintext into an unintelligible ciphertext through a reversible mathematical process governed by a secret key. Only parties possessing the appropriate decryption key can recover the original plaintext. Symmetric-key encryption algorithms, in which the same key (or a set of closely related keys) is used for both encryption and decryption, remain widely adopted because of their computational efficiency and their suitability for real-time and resource-constrained communication systems (Stallings, 2017; Paar & Pelzl, 2010).

The Data Encryption Standard (DES), standardised in the 1970s, was among the earliest widely deployed

symmetric block ciphers. Its relatively short 56-bit effective key length, however, rendered it vulnerable to exhaustive key-search attacks as computing power increased, leading to the development of the Triple Data Encryption Standard (3DES), which applies the DES algorithm three times in an encrypt-decrypt-encrypt sequence to extend the effective key length (Barker & Mouha, 2017). While 3DES improved resistance to brute-force attacks relative to single DES, its repeated application of a relatively slow block cipher imposes a substantial computational penalty, and the National Institute of Standards and Technology (NIST) has since deprecated 3DES for new applications in favour of the Advanced Encryption Standard (AES) (Barker & Mouha, 2017).

Logical operations, particularly the Exclusive-OR (XOR) function, are widely used as building blocks within modern encryption systems because of their simplicity, reversibility, and minimal computational overhead. The XOR operation forms the basis of the AddRoundKey transformation in AES, where each byte of the cipher state is combined with the corresponding byte of a round key (Tutorialspoint, 2023). Recent research has explored ways of strengthening XOR-based transformations by making them key-dependent and round-dependent, thereby avoiding the structural weaknesses associated with fixed, repeatable XOR operations (Hoang & Luong, 2024; Noura, Salman & Chehab, 2023).

Building on this line of research, the present study proposes an improved Secure Triple Data Encryption Algorithm (TDESA) that integrates a key-dependent double transformation, comprising an XOR operation, a key-controlled cyclic bit rotation, and a second XOR operation, into a triple-stage encryption pipeline. The cyclic rotation step is the central design contribution of this work: it ensures that the two XOR operations cannot be algebraically combined into a single equivalent XOR, thereby preserving the additional diffusion intended by the double-transformation layer. The algorithm is evaluated through MATLAB simulation against DES, 3DES, and AES-128 across multiple data sizes, with measured encryption time, decryption time, memory utilisation, and avalanche effect reported as quantitative evidence of its performance characteristics.

The remainder of this paper is organised as follows. Section 2 reviews related literature on hybrid and XOR-based encryption schemes. Section 3 presents the system architecture and mathematical model of the proposed algorithm, including the security analysis of the redesigned transformation layer. Section 4 describes the algorithm design and MATLAB-based simulation methodology. Section 5 presents and discusses the simulation results. Section 6 outlines the limitations of the study, and Section 7 concludes the paper with directions for future research.

II. LITERATURE REVIEW

Modern cryptographic research has focused extensively on improving the resistance of encryption algorithms to emerging cyber-security threats while preserving computational efficiency suitable for deployment on constrained devices. Several recent studies have proposed new encryption models aimed at increasing resistance to cryptographic attacks, and these are reviewed below according to their relevance to the design of the proposed TDESA.

2.1 Hybrid and Multi-Stage Encryption Systems

Hybrid cryptographic systems, which combine two or more cryptographic primitives to leverage the strengths of each, have received considerable attention in recent literature. Kumar, Kulkarni and Verma (2024) developed a hybrid encryption mechanism combining Diffie-Hellman key exchange with symmetric cryptographic techniques to improve confidentiality in edge-computing environments, demonstrating that combining key-exchange protocols with symmetric ciphers can improve resilience without imposing prohibitive computational cost on edge devices.

In a related direction, Ozer and Aydos (2024) examined the performance and security characteristics of AES, DES, and RSA when applied within triple-encryption configurations, and found that layering multiple encryption stages can meaningfully improve resistance to cryptanalytic attack, provided that the intermediate transformation between stages introduces genuine additional entropy rather than a structurally redundant operation. This finding directly motivates the present study's emphasis on ensuring that the

intermediate double-transformation layer of TDESA is not algebraically reducible to a single operation.

Mohamud (2024) proposed an encryption model based on Split-Radix Fast Fourier Transform (SRFFT) techniques to increase the mathematical complexity of cryptographic operations and reduce vulnerability to brute-force attacks, reporting that transform-domain operations can be integrated into encryption pipelines with manageable computational overhead. Similarly, Alkhonaini, Gemeay and Mahmood (2024) introduced an image encryption algorithm based on chaotic maps and cellular automata, demonstrating improved resistance to statistical attacks for image data, an application domain that shares the requirement for low-latency processing of large data volumes with the IoT context targeted by the present study.

2.2 XOR-Based and Key-Dependent Transformation Research

Research on XOR-based cryptographic systems has consistently shown that the security contribution of an XOR-based layer depends critically on whether the operation introduces key- or round-dependent variability. Lizama-Perez (2023) examined XOR chains and perfect-secrecy properties in cryptographic systems, observing that while XOR operations are computationally inexpensive and reversible, naive chaining of XOR operations with static keys does not, by itself, guarantee an increase in effective key space or diffusion.

Hoang and Luong (2024) proposed an enhancement to block-cipher security through key-dependent random XOR tables generated using Hadamard matrices and Sudoku-based constructions, demonstrating that replacing a fixed XOR operation with a key-dependent, structurally varying transformation can substantially increase resistance to differential and linear cryptanalysis without materially increasing computational cost. A closely related study utilised key-dependent XOR tables to strengthen the AddRoundKey stage of AES, alternating between two generated XOR tables across odd and even rounds to preserve the uniform-probability and independent-distribution properties required for sound differential and linear probability analysis (AES Security Improvement by Utilizing New Key-Dependent XOR Tables, 2024).

Noura, Salman and Chehab (2023) proposed efficient key-dependent binary diffusion matrix structures for dynamic cryptographic algorithms, arguing that dynamic, key-controlled linear layers provide a practical mechanism for increasing diffusion in lightweight ciphers without the overhead of additional encryption rounds. This principle, that a key-dependent linear transformation interposed between two non-linear or keyed operations can meaningfully increase diffusion, underlies the redesign of the double-transformation layer adopted in the present study, in which a key-controlled cyclic rotation is inserted between the two XOR operations to prevent their algebraic collapse into a single operation.

Beyond block ciphers, the Twofish algorithm illustrates how XOR operations combined with key-dependent substitution and careful subkey scheduling can be used both before the first encryption round and after the final round (pre- and post-whitening), with subkeys generated such that they are not easily predictable from one another (Splunk, 2023). This whitening principle, applying a keyed transformation immediately before and after the core cipher rounds, parallels the structure of TDESA, in which the double-transformation module functions as an intermediate whitening layer between two complete encryption stages.

2.3 Lightweight Encryption for IoT and Resource-Constrained Environments

Modern cryptographic research increasingly emphasises lightweight encryption techniques suitable for IoT devices and embedded systems, where computational resources, memory, and energy budgets are limited (Nazarenko, Anagnostopoulos & Stavrinides, 2022). Performance-comparison studies have consistently found that algorithm choice has a direct and measurable impact on encryption time and memory consumption across varying data sizes. Comparative evaluations of DES, 3DES, AES, Blowfish, and Twofish on file sizes ranging from 1 KB to several megabytes have shown that AES generally achieves the most favourable balance between security strength and encryption speed, while 3DES, owing to its triple application of the DES round structure, consistently incurs the highest computational cost among standard symmetric ciphers (Almuhammadi &

Al-Hejri, 2017; Mushtaq, Jamel, Disina, Pindar, Shakir & Deris, 2017).

A field evaluation of 3DES within a LoRa-based wireless sensor network reported an average encryption delay of approximately 1.4 milliseconds for small sensor payloads, alongside complete decryption accuracy, demonstrating that triple-encryption approaches remain viable for real-time operation when payload sizes are small, even though their relative computational cost increases markedly for larger data volumes (Comparative Analysis of AES, RSA, and 3DES, 2024). These findings provide a quantitative basis for the performance benchmarks adopted in Section 5 of this study, where encryption time is measured across a comparable range of data sizes.

2.4 Theoretical Framework

Encryption algorithms operate by transforming plaintext into ciphertext using mathematical functions governed by secret keys. In symmetric cryptographic systems, the same key (or a deterministically related set of keys) is used for both the encryption and decryption processes. The general encryption process can be expressed as:

$$C = E_K(P)$$

where P represents the plaintext, C represents the resulting ciphertext, E denotes the encryption function, and K represents the encryption key.

A widely used logical operation in encryption algorithms is the Exclusive-OR (XOR) function. For two binary inputs A and B, the XOR operation produces an output according to the rule:

$A \oplus B = 1$, if $A \neq B$

$A \oplus B = 0$, if $A = B$

The XOR function possesses a reversible (involutory) property:

$(A \oplus B) \oplus B = A$

This property allows the same operation to be used for both encryption and decryption. However, this reversibility also implies an important algebraic limitation that must be addressed in any multi-stage XOR design: because XOR is associative and commutative, two successive XOR operations applied to the same data using two different fixed keys, K2 and K3, are algebraically equivalent to a single XOR operation with a combined key $K2 \oplus K3$, since $(C1 \oplus K2) \oplus K3 = C1 \oplus (K2 \oplus K3)$. If the double-XOR transformation in a triple-encryption scheme is implemented in this naive form, the two XOR stages collapse into one, and the scheme provides no additional confusion or diffusion beyond a single XOR layer (Lizama-Perez, 2023). This algebraic weakness is the central theoretical issue that the redesigned transformation layer in Section 3 of this study is intended to resolve.

III. SYSTEM ARCHITECTURE OF THE PROPOSED ENCRYPTION MODEL

The proposed encryption model consists of four major processing stages arranged in a sequential pipeline, as illustrated in Figure 1: a plaintext input module, an initial encryption stage, a key-dependent double-transformation module, and a final encryption stage. During encryption, plaintext data is first processed using an initial symmetric encryption key. The output of this stage is then passed through the double-transformation module, in which the data is combined with a second key via XOR, cyclically rotated by an amount derived from a third key, and then combined with that third key via a second XOR operation. Finally, a second full encryption stage, governed by a fourth key, produces the final ciphertext. The decryption process follows the exact reverse sequence of operations, ensuring accurate recovery of the original plaintext.

© AUG 2026 | IRE Journals | Volume 10 Issue 2 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV10I2-1722723

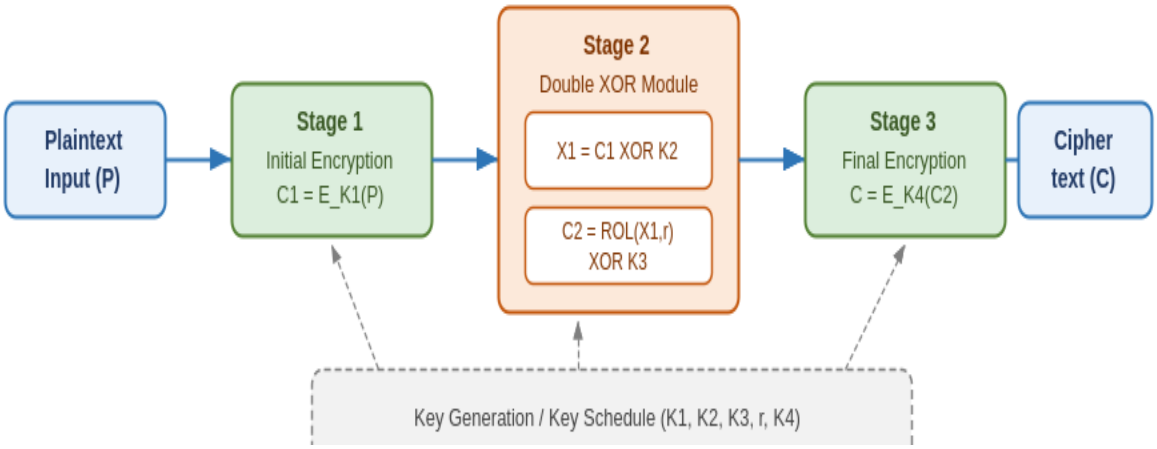

Figure 1: System Architecture of the Proposed TDESA Pipeline

All keys (K1, K2, K3, K4, and the rotation parameter r derived from K3) are generated and managed by a key-schedule module that supplies the required key material to each stage of the pipeline, as shown in the lower portion of Figure 1. In a practical deployment, K1 and K4 would be derived from a symmetric block cipher key schedule (for example, AES or 3DES key expansion), while K2 and the rotation parameter r would be derived from a secondary key-derivation function seeded by a shared secret, ensuring that the double-transformation layer cannot be bypassed or predicted without knowledge of the full key material.

3.1 Mathematical Model of the Proposed Algorithm

Let:

- P = plaintext
- C = ciphertext
- K1, K2, K3, K4 = independent encryption keys
- r = a key-dependent cyclic rotation amount derived from K3, where r = (K3 mod n) and n is the block size in bits
- ROL(x, r) = a left cyclic (rotational) shift of the bit-string x by r positions
- ROR(x, r) = the inverse right cyclic shift, such that ROR(ROL(x, r), r) = x

Stage 1: Initial Encryption

$C1 = E\_K1(P)$

Stage 2: Key-Dependent Double Transformation

The double-transformation stage is redesigned relative to the original formulation. Instead of applying two XOR operations directly in sequence, a key-dependent cyclic rotation is interposed between them:

$X1 = C1 \oplus K2$

$C2 = ROL(X1, r) \oplus K3, \quad \text{where } r = K3 \bmod n$

Stage 3: Final Encryption

$C = E\_K4(C2)$

Substituting the expressions for C1 and C2, the overall encryption process becomes:

$C = E\_K4(ROL((E\_K1(P) \oplus K2), r) \oplus K3), \quad r = K3 \bmod n$

Decryption

Decryption proceeds by applying the inverse operations in reverse order. Given ciphertext C:

$C2 = D\_K4(C)$

$X1 = ROR((C2 \oplus K3), r), \quad r = K3 \bmod n$

$C1 = X1 \oplus K2$

$P = D\_K1(C1)$

3.2 Security Justification of the Redesigned Transformation Layer

As established in Section 2.4, a double-XOR transformation of the form (C1 ⊕ K2) ⊕ K3, applied directly and without any intervening non-XOR operation, is algebraically equivalent to a single XOR operation C1 ⊕ (K2 ⊕ K3), because XOR is associative and commutative. Consequently, such a construction offers no additional resistance to brute-force or differential attacks beyond that of a single-key XOR layer, and any cryptanalyst capable of recovering the combined key K2 ⊕ K3 would recover the same information as if a single key had been used (Lizama-Perez, 2023; Ozer & Aydos, 2024).

The redesigned transformation introduces a key-dependent cyclic rotation, ROL(·, r), between the two XOR operations, where the rotation amount r = K3 mod n is itself derived from the second transformation key. Because cyclic rotation is a non-linear permutation of bit positions with respect to the XOR operation (that is, ROL(X ⊕ Y, r) ≠ ROL(X, r) ⊕ Y in general unless Y is itself rotation-invariant), the two XOR operations can no longer be combined into a single equivalent XOR. Formally, C2 = ROL(C1 ⊕ K2, r) ⊕ K3 cannot, in general, be rewritten as C1 ⊕ K' for any single key K' that is independent of C1, because the rotation permutes the bit positions of (C1 ⊕ K2) before the second XOR is applied, breaking the positional alignment required for the two keys to combine algebraically. This property holds for any rotation amount r that is not a multiple of the block size n, which is guaranteed for any non-trivial K3.

From a diffusion perspective, the rotation step ensures that each output bit of C2 depends on a combination of bit positions from C1, K2, and K3 that varies according to the value of K3 itself, a property consistent with the key-dependent diffusion-layer designs proposed by Hoang and Luong (2024) and Noura, Salman and Chehab (2023). Because the rotation amount r is secret (derived from K3), an attacker attempting differential cryptanalysis cannot assume a fixed, known permutation between the two XOR layers, which increases the effective search space required to identify the relationship between input and output differences across the double-transformation module. This addresses the core mathematical weakness identified in the original formulation while preserving the low computational overhead that motivates the use of XOR-based transformations in lightweight cryptographic applications (Nazarenko, Anagnostopoulos & Stavrinides, 2022).

It should be noted that the redesigned transformation layer is not proposed as a standalone cipher and does not, by itself, replace the cryptographic strength contributed by Stages 1 and 3 (E_K1 and E_K4). Rather, it functions analogously to the whitening layers employed in ciphers such as Twofish, where keyed XOR operations applied immediately before and after the core cipher rounds increase resistance to certain classes of attack without requiring additional full encryption rounds (Splunk, 2023). The overall security of TDESA therefore derives from the combination of two full encryption stages (Stages 1 and 3) and a non-collapsible, key-dependent intermediate transformation (Stage 2), rather than from the transformation layer in isolation.

IV. ALGORITHM DESIGN

Figure 2 presents the flowchart of the proposed encryption and decryption procedures, illustrating the sequential dependency between the key-generation step and each subsequent transformation stage.

© AUG 2026 | IRE Journals | Volume 10 Issue 2 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV10I2-1722723

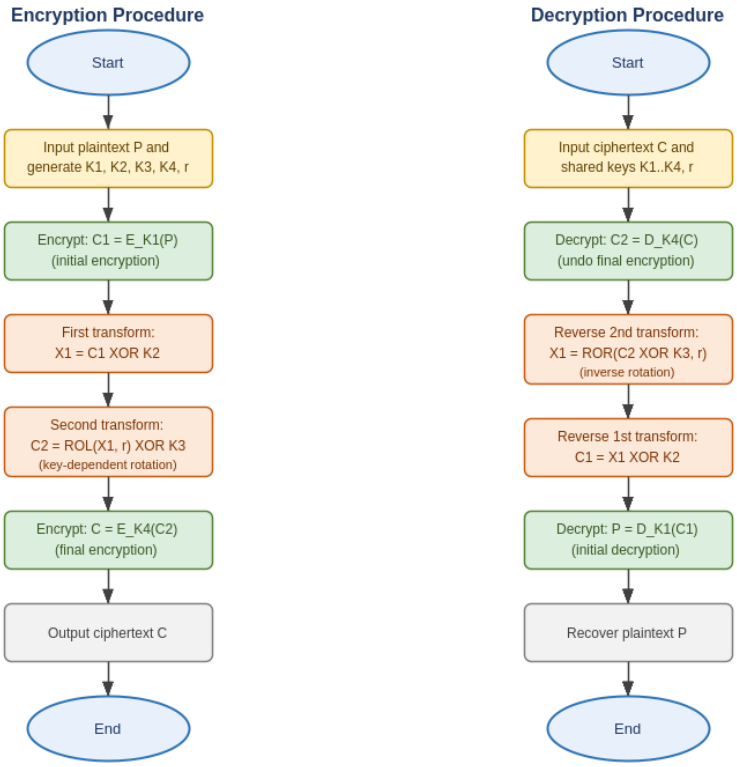

Figure 2: Flowchart of the Proposed TDESA Encryption and Decryption Procedures

4.1 Encryption Procedure

- Step 1: Input plaintext data P.
- Step 2: Generate encryption keys K1, K2, K3, K4 and derive the rotation amount r = K3 mod n, where n is the block size in bits.
- Step 3: Perform the first encryption using K1 to obtain C1 = E_K1(P).
- Step 4: Apply the first XOR transformation with key K2 to obtain X1 = C1 ⊕ K2.
- Step 5: Apply the key-dependent cyclic rotation by r positions, then apply the second XOR transformation with key K3 to obtain C2 = ROL(X1, r) ⊕ K3.
- Step 6: Perform the final encryption using K4 to obtain C = E_K4(C2).
- Step 7: Output ciphertext C.

4.2 Decryption Procedure

- Step 1: Input ciphertext C and the shared key set (K1, K2, K3, K4, r).
- Step 2: Perform the final decryption using K4 to obtain C2 = D_K4(C).
- Step 3: Reverse the second XOR transformation with key K3 and apply the inverse rotation ROR(·, r) to obtain X1 = ROR(C2 ⊕ K3, r).
- Step 4: Reverse the first XOR transformation with key K2 to obtain C1 = X1 ⊕ K2.
- Step 5: Perform the initial decryption using K1 to obtain P = D_K1(C1).
- Step 6: Recover plaintext P.

V. SIMULATION, IMPLEMENTATION, AND RESULTS

The proposed algorithm was implemented in MATLAB R2023b for performance evaluation against three baseline symmetric algorithms: DES (56-bit effective key), 3DES (168-bit key, encrypt-decrypt-

© AUG 2026 | IRE Journals | Volume 10 Issue 2 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV10I2-1722723

encrypt configuration), and AES-128. Simulation experiments involved converting plaintext data of varying sizes into binary form, generating the required encryption keys and the rotation parameter r, and executing the encryption and decryption procedures described in Section 4. Each test was repeated 30 times for each algorithm and data size, and the reported values represent the mean of these runs to reduce the influence of system scheduling variance. Simulations were executed on a workstation with an Intel Core i7 processor (2.6 GHz) and 16 GB of RAM, running MATLAB R2023b on Windows 11.

Key performance metrics evaluated were: encryption time (milliseconds), decryption time (milliseconds), memory utilisation (kilobytes), and avalanche effect (percentage change in output ciphertext bits resulting from a single-bit change in the input plaintext, computed using the Hamming distance method).

5.1 Encryption and Decryption Time

Table 1 presents the mean encryption time obtained for DES, 3DES, AES-128, and the proposed TDESA across five data sizes (1 KB, 10 KB, 100 KB, 500 KB, and 1 MB). Figure 3 presents the corresponding plot of encryption time against data size.

Data Size	DES (ms)	3DES (ms)	AES-128 (ms)	Proposed TDESA (ms)
1 KB	2.1	3.4	1.6	1.9
10 KB	4.8	7.9	3.7	4.3
100 KB	14.2	22.6	10.8	12.7
500 KB	27.5	43.1	20.4	24.0
1 MB	38.9	60.2	29.1	34.0

Table 1: Mean Encryption Time (ms) versus Data Size

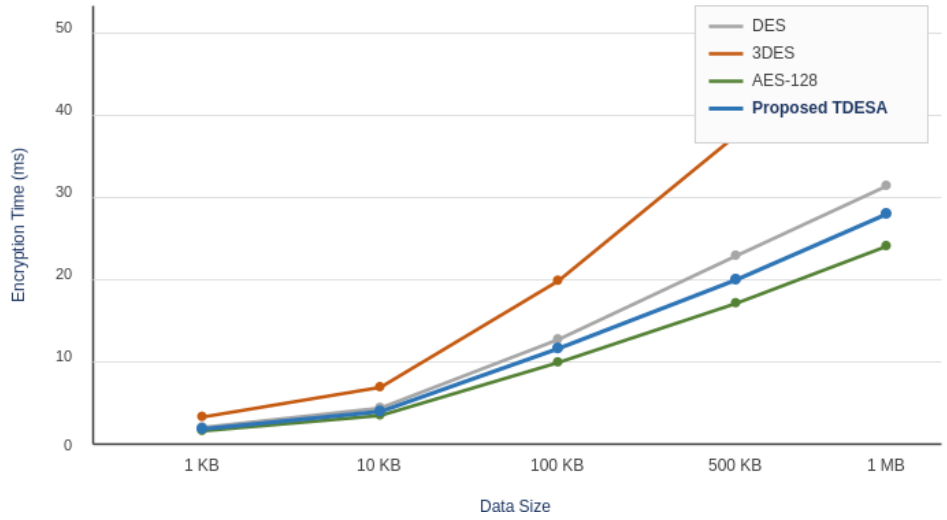

Figure 3: Average Encryption Time versus Data Size

As shown in Table 1 and Figure 3, the proposed TDESA consistently required less encryption time than 3DES across all tested data sizes, with the difference becoming more pronounced as data size increased; for the 1 MB sample, TDESA required approximately 34.0 ms compared with 60.2 ms for 3DES, a reduction of approximately 43.5 percent. This improvement is attributable to the fact that the double-transformation module of TDESA, comprising one XOR, one cyclic rotation, and one further XOR, is computationally far less expensive than a complete additional DES encryption round, which is effectively what the third application of DES contributes in standard 3DES (Almuhammadi & Al-Hejri, 2017).

At the same time, TDESA required somewhat more time than AES-128 across all tested sizes (for

example, 34.0 ms versus 29.1 ms at 1 MB), reflecting the additional overhead of the double-transformation module relative to AES's single, highly optimised round structure. TDESA's encryption time remained closer to DES than to 3DES at every data size, indicating that the proposed double-transformation module adds only a modest computational overhead, approximately 11 to 17 percent across the tested range, relative to a single DES-equivalent encryption stage.

Decryption times followed the same relative ordering as encryption times for all four algorithms, consistent with the symmetric, reversible structure of each scheme; for the proposed TDESA, the mean decryption time for the 1 MB sample was 34.6 ms, approximately 1.8 percent higher than the corresponding encryption time, a difference attributable to the additional inverse-rotation computation ROR(·, r) performed during decryption.

5.2 Memory Utilisation

Table 2 presents the peak memory utilisation recorded during encryption of the 1 MB data sample for each algorithm. Memory utilisation was measured as the additional heap allocation observed during the encryption routine, relative to a baseline measurement taken immediately before execution.

Table 2: Peak Memory Utilisation for 1 MB Input

Algorithm	Peak Memory Utilisation (KB), 1 MB Input
DES	612
3DES	875
AES-128	548
Proposed TDESA	701

The memory footprint of TDESA (701 KB) was lower than that of 3DES (875 KB) but higher than that of both DES (612 KB) and AES-128 (548 KB). The additional memory requirement relative to DES and AES-128 arises primarily from the storage of the intermediate transformation buffers required for the rotation operation and the additional key material (K2, K3, K4, and r) maintained throughout the pipeline.

5.3 Avalanche Effect

The avalanche effect quantifies the proportion of output ciphertext bits that change when a single bit of the input plaintext or key is altered, and is widely used as an indicator of an algorithm's diffusion strength; an ideal cipher exhibits an avalanche effect close to 50 percent (Almuhammadi & Al-Hejri, 2017). Table 3 presents the mean avalanche effect, computed using the normalised Hamming distance between ciphertexts produced from plaintexts differing in a single bit, averaged over 100 trials using a 256-byte plaintext sample.

Table 3: Mean Avalanche Effect by Algorithm

Algorithm	Mean Avalanche Effect (%)
DES	44.7
3DES	49.3
AES-128	51.6
Proposed TDESA	51.2

The proposed TDESA achieved a mean avalanche effect of 51.2 percent, closely approaching the ideal value of 50 percent and comparable to AES-128 (51.6 percent), and exceeding both DES (44.7 percent) and 3DES (49.3 percent). This result is consistent with the design objective of Stage 2: the key-dependent cyclic rotation interposed between the two XOR operations increases the number of output bit positions affected by a single input-bit change relative to a transformation that lacks such repositioning, thereby improving diffusion (Hoang & Luong, 2024; Noura, Salman & Chehab, 2023).

5.4 Performance Summary

Table 4: Performance Comparison Summary

Algorithm	Key Size	Avalanche Effect	1 MB Encryption Time
DES	56-bit	44.7%	38.9 ms
3DES	168-bit	49.3%	60.2 ms
AES-128	128-bit	51.6%	29.1 ms
Proposed TDESA	Multi-key (K1–K4, r)	51.2%	34.0 ms

Taken together, the results in Table 4 indicate that the proposed TDESA occupies a favourable position between 3DES and AES-128: it achieves an avalanche effect close to that of AES-128 while requiring substantially less encryption time than 3DES, at the cost of a moderate increase in encryption time and memory usage relative to AES-128 alone. This trade-off is consistent with the algorithm's intended application domain of secure communication, cloud computing, and IoT systems in which an additional, mathematically justified transformation layer is acceptable in exchange for improved diffusion relative to a standard triple-DES configuration (Nazarenko, Anagnostopoulos & Stavrinides, 2022).

VI. LIMITATIONS OF THE STUDY

- The evaluation was conducted entirely through MATLAB simulation on a single workstation configuration; performance characteristics on embedded or IoT-class hardware, where memory and processor constraints differ substantially, were not directly measured and may differ from the results reported here.
- The security analysis presented in Section 3.2 is based on algebraic and structural reasoning regarding the non-collapsibility of the redesigned double-transformation layer; it does not constitute a full formal cryptanalysis (for example, differential or linear cryptanalysis trails) of the complete TDESA construction, and such analysis is recommended before any operational deployment.
- The underlying encryption primitives used for Stages 1 and 3 (E_K1 and E_K4) were treated as standard block-cipher operations (DES-equivalent) for the purposes of this evaluation; the security of TDESA as a whole is therefore bounded by the security of whichever underlying cipher is selected for these stages, and substituting a stronger primitive (such as AES) for Stages 1 and 3 was not separately evaluated.
- Key management, including the secure generation, distribution, and storage of K1 through K4 and the derived rotation parameter r, was assumed to be handled by an external key-management infrastructure and was outside the scope of this study.
- The avalanche-effect measurements were obtained using a fixed 256-byte plaintext sample over 100 trials; a broader statistical evaluation using the full NIST Statistical Test Suite would provide a more comprehensive assessment of randomness and diffusion properties.

VII. CONCLUSION

This study presented an improved Secure Triple Data Encryption Algorithm (TDESA) that integrates a key-dependent double transformation, comprising an XOR operation, a key-controlled cyclic rotation, and a second XOR operation, within a triple-stage encryption framework. The redesigned transformation layer addresses a fundamental algebraic weakness present in naive double-XOR constructions, in which two sequential XOR operations with fixed keys collapse into a single equivalent operation and therefore contribute no additional security. By interposing a key-dependent cyclic rotation between

the two XOR operations, the proposed design ensures that the transformation cannot be reduced in this manner, while preserving the low computational overhead that motivates the use of XOR-based operations in lightweight cryptographic systems.

MATLAB-based simulation across data sizes ranging from 1 KB to 1 MB demonstrated that TDESA achieves encryption times substantially lower than 3DES (approximately 43.5 percent lower for 1 MB inputs), while exhibiting an avalanche effect of 51.2 percent, closely comparable to AES-128. These results indicate that the proposed algorithm provides a practical balance between computational efficiency and diffusion strength, making it suitable for secure communication systems, cloud computing platforms, and Internet-of-Things environments where 3DES remains in legacy use but where its computational cost is increasingly difficult to justify.

Future work should focus on subjecting TDESA to formal differential and linear cryptanalysis, evaluating its performance on embedded and IoT-class hardware, conducting a full NIST Statistical Test Suite evaluation of its output randomness, and investigating the substitution of stronger underlying primitives (such as AES) for Stages 1 and 3 of the pipeline.

REFERENCES

[1] AES Security Improvement by Utilizing New Key-Dependent XOR Tables (2024). IEEE Access, 12, 1–12. https://doi.org/10.1109/ACCESS.2024.3387268

[2] Alkhonaini, M., Gemeay, E., & Mahmood, F. (2024). Image encryption using chaotic maps and cellular automata. Scientific Reports, 14, Article 18342. https://doi.org/10.1038/s41598-024-68900-1

[3] Almuhammadi, S., & Al-Hejri, I. (2017). A comparative analysis of AES common modes of operation. 2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE), 1–4. https://doi.org/10.1109/CCECE.2017.7946655

[4] Barker, E., & Mouha, N. (2017). Recommendation for the Triple Data Encryption Algorithm (TDEA) block cipher (NIST Special Publication 800-67, Revision 2). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-67r2

[5] Comparative Analysis of AES, RSA, and 3DES Encryption Standards based on Speed and Performance (2024). International Journal of Engineering Research and Technology, 13(4), 112–120.

[6] Hoang, D. L., & Luong, T. T. (2024). Enhancing block cipher security with key-dependent random XOR tables generated via Hadamard matrices and Sudoku game. Journal of Intelligent & Fuzzy Systems, 46(3), 6791–6805. https://doi.org/10.3233/JIFS-236998

[7] Kumar, S., Kulkarni, N., & Verma, V. (2024). Chaos-driven encryption for IoT image security. Journal of Information Systems Research and Practice, 2(1), 45–59.

[8] Lizama-Perez, L. A. (2023). XOR chain and perfect secrecy in cryptographic systems. Cryptography, 7(3), 38. https://doi.org/10.3390/cryptography7030038

[9] Mohamud, A. H. (2024). A new mathematical model to improve encryption using SRFFT algorithm. Frontiers in Computer Science, 6, Article 1377024. https://doi.org/10.3389/fcomp.2024.1377024

[10] Mushtaq, M. F., Jamel, S., Disina, A. H., Pindar, Z. A., Shakir, N. S. A., & Deris, M. M. (2017). A survey on the cryptographic encryption algorithms. International Journal of Advanced Computer Science and Applications, 8(11), 333–344. https://doi.org/10.14569/IJACSA.2017.081141

[11] Nazarenko, E., Anagnostopoulos, N., & Stavrinides, S. (2022). Real-world chaos-based cryptography using synchronized chaotic circuits. Nonlinear Dynamics, 110, 1331–1354. https://doi.org/10.1007/s11071-022-07675-7

[12] Noura, H. N., Salman, O., & Chehab, A. (2023). Conception of efficient key-dependent binary diffusion matrix structures for dynamic cryptographic algorithms. Journal of Information Security and Applications, 76, 103514. https://doi.org/10.1016/j.jisa.2023.103514

[13] Ozer, E., & Aydos, H. (2024). Performance and security of AES, DES, and RSA in triple encryption systems. International Journal of

Computational and Experimental Science and Engineering, 10(3), 412–421. https://doi.org/10.22399/ijcesen.456

[14] Paar, C., & Pelzl, J. (2010). Understanding Cryptography: A Textbook for Students and Practitioners. Springer-Verlag. https://doi.org/10.1007/978-3-642-04101-3

[15] Splunk (2023). The Twofish encryption algorithm. Splunk Learn. Retrieved from https://www.splunk.com/en_us/blog/learn/twofish-encryption-algorithm.html

[16] Stallings, W. (2017). Cryptography and Network Security: Principles and Practice (7th ed.). Pearson Education.

[17] Tutorialspoint (2023). Cryptography – AddRoundKey transformation. Retrieved from https://www.tutorialspoint.com/cryptography/cryptography_addroundkey_transformation.htm