Sistemasi: Jurnal Sistem Informasi
Volume 15, Nomor 8, 2026: 3303-3326
ISSN:2302-8149
e-ISSN:2540-9719

Enhancing DES Encryption Efficiency through a Metrics-Centric Pipeline and AI-Driven Analytical Framework

[1]Alaa Othman Mahmood*
[1]Faculty of Science & Literature, American University of Culture & Education, Beirut, Lebanon
*e-mail: allawi19822016@gmail.com

(***received*:** 27 July 2026, ***revised*:** 24 August 2026, ***accepted*:** 25 August 2026)

Abstract

This paper introduces an improved Data Encryption Standard (DES) framework that incorporates a metrics-driven data processing pipeline, AI-powered analytical decision support and a quantum-inspired entropy-based key generation process to enhance both encryption efficiency and key security. It validates data, preprocesses it, encodes it, optimizes it, encrypts it, monitors it in real-time, analyzes entropy and also analyzes the execution time, encryption speed, throughput, CPU utilization, memory consumption and key randomness. Experimental results indicate that the encryption throughput was achieved on average at 5.11 MB/s, with a maximum value of 12.15 MB/s, while the time required for encryption and decryption were on average 42.67 ms and 44.21 ms, respectively. The proposed quantum-inspired key generator has an average key entropy of 0.82 (compared with 0.68 for standard random keys) and a higher uniformity of 0.88 (compared with 0.72 for standard random keys), which corresponds to an approximately 20.6% improvement. Furthermore, the quantum-inspired approach achieved a maximum entropy of 0.95 compared with 0.89 for random generation, and its strongest keys had a collision rate of no more than 0.01%. The overall average security improvement for the AI analysis was 13.30% and the overall completion rate for the full processing pipeline was 98.50%. The findings highlight practical solutions for optimizing the performance, randomness, and security evaluation of DES-based encryption by incorporating AI-driven analysis, entropy-driven key generation, and regular monitoring.

Keywords: artificial intelligence, data processing pipeline, DES Encryption, entropy analysis, multi-source entropy-driven key generation, resource utilization.

1 Introduction

Cryptography is integral to the security of present-day digital communication systems. In classical cryptography algorithms, the Data Encryption Standard (DES) holds great historical significance and is regarded as one of the first examples of symmetric-key block cipher algorithms used today. Though DES has mostly been superseded by more sophisticated algorithms like AES in applications where high levels of security are required, DES is still extensively researched due to its simple yet well-defined Feistel cipher design and its utility as a standard for testing cryptographic performance optimizations [1]. The Data Encryption Standard algorithm utilizes a fixed block size of 64 bits and a key length of 56 bits in its encryption process through 16 rounds of substitution and permutation. In each of the rounds, complex permutations are performed via expansion and substitution box functions, followed by permutation, XORing of round keys, and inverse permutation. The DES algorithm presents a strong base for studying symmetric-key block cipher algorithms. However, being a legacy algorithm, DES does not have much computational power compared to current encryption standards and is susceptible to brute-force attacks [2].

Recent trends in cryptography research are directed at the improvement of various encryption systems concerning security level, computational efficiency, adaptability, and intelligent decision-making. Traditional systems encrypt messages statically using algorithms irrespective of input information features, system conditions, and security needs. That is why modern scientists try to create adaptive and intelligent systems, which may be able to change their encryption algorithms depending on data and its specific characteristics [3].

 ISSN:2302-8149
e-ISSN:2540-9719

Another problem, which is related to the increasing volume and complexity of digital data, is the necessity for more efficient systems capable of performing fast and accurate computations. High-speed data systems, such as cloud computing systems and IoT devices, are needed in order to provide timely results to end users. Therefore, there appeared an idea of designing systems using a pipeline approach, dividing them into several computing modules and monitoring their performance in order to achieve better results in terms of computational efficiency and accuracy [4]. The second new trend in cryptography is the use of artificial intelligence technology in cryptography. Artificial intelligence technology has been used for intrusion detection, anomaly detection and encryption optimization. In the encryption stage, artificial intelligence technology can be used to identify the data pattern, calculate entropy and propose encryption parameters, such as key size and encryption algorithms. This approach brings a degree of intelligence to the deterministic approach, which does not have this. But the application of artificial intelligence technology in cryptography has an impact on the cost of computational resources [3].

With the advancement of AI, quantum algorithms or entropy-based key generation algorithms are potential solutions to enhance security in cryptography. Key generation is crucial in designing a cryptosystem that is resistant to cryptanalysis and brute force attacks. Traditional pseudo-random generators suffer from issues with predictability and entropy distribution. Alternatively, quantum algorithms or entropy-enhancement enhance randomness, reduce collisions and protect keys. These methods can be used in hybrid cryptosystems, particularly in high security applications [5]. While these advances have been made, a system that incorporates performance improvement, smart analysis and secure key generation for use in an encryption system has not yet been established. Most current systems focus on only one of these developments, such as improving speed, using AI for classification or boosting key encryption strength, but not all three [6].

Hence, a holistic approach involving the use of a metric-based data processing pipeline with intelligence and improved key generation is required. This will allow real-time performance measurements, intelligence-based encryption decisions and suitable cryptography strength assessment. The research proposed in this paper will fill this gap by proposing an improved DES encryption model [7]. Although DES is still considered the standard for symmetric encryption, earlier studies have tended to deal with the shortcomings of cryptographic systems on an individual basis. Some studies are focused on speeding up encryption and making it more efficient to compute; others are focused in changing the DES structure, strengthening the key schedule, improving key generation, or implementing AI to analyze and make security decisions. Although these approaches offer helpful enhancements to various components of cryptographic functionality, they fail to adequately address the need to combine cryptographic performance, smart analysis and key generation into a single cohesive application. Thus, the development of a system capable of simultaneously tracking the performance of encryption, characterizing the data based on artificial intelligence and producing high-quality cryptographic keys based on improved entropy while maintaining a reasonable computational efficiency exists as a gap in the research.

Therefore, the research gap that this study aims to address is the absence of an integrated metrics-driven DES framework that integrates these three components, instead of optimizing them individually. The previous approaches lack the ability to gather and analyze the time of encryption, throughput, CPU and memory usage, data entropy, and key entropy along with AI-driven recommendations and entropy-based key generation. This separation can be difficult to understand the balance of security, computational efficiency and intelligent decision making in the same cryptographic architecture.

The novelty of the proposed study is the assembly of a metrics-driven multi-stage processing pipeline and AI-based analytical decision support and a quantum inspired entropy-

 ISSN:2302-8149
e-ISSN:2540-9719

based key generation mechanism in one DES oriented framework. The proposed system is not just an individual DES component modification, but it introduces a new single evaluation and optimization framework, where the performance of the DES is evaluated with the utilization of resources, entropy, key strength, and an AI-based analysis. The framework is therefore a structured approach to assess and enhance the performance and security of DES, which in turn closes the gap between the single cryptographic optimization techniques and a holistic, measurable and adaptive encryption system.

2 Related Works

In [8] researchers propose an enhanced DES-based cryptographic algorithm with a longer key length of 1024 bits and a key split into 16 subkeys of 64 bits. The techniques are based on multiple key generations for different rounds of encryption and blind-search security analysis to improve security against brute-force attacks. The problems are that while encryption is greatly improved, the increased key management and computational complexity are not resolved, which could impact efficiency for resource-limited systems. The current research addresses these limitations by preserving a structured metrics-based approach for DES with efficient key management through entropy-driven key generation to provide a balance between security and performance without a huge overhead in key expansion.

In [9] the authors propose a parallel Double DES (2DES) encryption system for performance improvement and increased speed. The methods used are parallelisation of the encryption and decryption processes using the Message Passing Interface (MPI) between processors to allow the parallel execution of the sequential 2DES system. The drawbacks are that while efficiency is improved, the system is open to advanced attacks because of the weakness of 2DES and there is communication overhead in parallel systems. The present study addresses these concerns by using an efficient single DES system with performance monitoring and analysis, which allows an increase in efficiency by enhancing the system rather than increasing the encryption layers and complexity of a distributed system.

In [10] The authors suggest a modified Simplified DES (SDES) algorithm for use in the security of smart card data by adding additional transformations. The techniques are related to the addition of complement and shift transformations to the original SDES algorithm to improve confusion and diffusion characteristics of encryption. Unfortunately, while the algorithm is more secure than the original SDES, it is still susceptible to many common attacks and cannot be scaled for large data or high-speed applications. Our research improves on these approaches by using a full DES system with PKCS5 padding, ECB mode of operation and key generation with improved entropy, providing increased security and flexibility with different data size and system requirements.

In [11] authors develop an improved DES algorithm to improve security against brute-force and cryptanalysis. The approaches are that the DES f-function is modified by using striding instead of XOR operations between the subkeys and plaintext to increase non-linearity and diffusion. The limitations are that while the avalanche effect is enhanced, it complicates the algorithm and might lower its efficiency without a holistic system-level optimization approach. The present study addresses these by leveraging the reliability of a standard DES implementation and fully integrating a metrics-driven framework for AI-based analysis, monitoring and pipeline processing, enabling efficient real-time system performance evaluation with enhanced security metrics.

In [12] authors suggest an enhanced DES framework to increase security by changing the key schedule. The methods involve using two keys, where the first key is applied directly in rounds 1-8 and the second key is first encrypted using an enhanced Caesar cipher and then used in rounds 9-16, enhancing the key variability in different rounds. The drawbacks are that the key transformation process adds to the complexity of the encryption and still uses simple

 ISSN:2302-8149
e-ISSN:2540-9719

classical substitution (Caesar-based encryption) which may not be robust to contemporary cryptanalysis. The current work overcomes these limitations by using a quantum-inspired entropy-based key generator and multi-source randomness instead of weak classical transformations, which enhances unpredictability and efficiency.

In [13] the authors introduce a Key-based Enhancement DES (KE-DES) technique for text encryption. The approaches are based on modifying the DES key schedule with even-odd bit transposition and insertion of a Key-Distribution (K-D) function that is derived from bits of PC-1, PC-2 and expanded right hand side data, in each encryption round. The drawbacks are the additional key manipulations and rounds make both implementation and real-time heavy processing more difficult and performance analysis more complex. In this paper, these problems are resolved by maintaining the DES structure, supported by a single unified metrics collection platform and a fast key generation process to deliver security with little or no structural changes and processing time.

In [14] a modified version of the DES algorithm is proposed by extending the key and block sizes from 64 bits to 128 bits, which results in a changed structure. The methods include redesigning the components of DES such as permutation, substitution and key scheduling to support the new block and key sizes to enhance confusion and diffusion. The drawbacks are that the increased size drastically increases the computational complexity and decreases compatibility with existing DES implementations and cryptographic systems. In this study we address these limitations by maintaining the standard DES structure but enhancing security by generating keys using increased entropy and system-level optimisation to ensure compatibility, efficiency and scalability for various data types.

In [15] the authors propose a modification to the DES algorithm by replacing the substitution operation which is the only nonlinear part of the algorithm. The approach is based on the development of a new 6×6 S-box with Galois field theory to improve security against linear attacks and differential cryptanalysis and an increase in key space to improve resistance to brute-force attacks. The limitations are that while S-box strength is improved, the modification is limited to only one of DES components and does not consider system-level performance, resource efficiency and the overall encryption process. The current study addresses these limitations by including the entire metrics-driven cryptographic design that holistically measures performance, entropy, and system efficiency in a balanced manner while achieving improvements in key management, data pipeline, and algorithm implementation.

3 Methodology

This research introduces a holistic metrics-based approach to studying and improving the encryption process of DES in an integrated framework incorporating cryptography implementation, real-time monitoring, data pipeline, and AI-based analysis. The present approach in Figure 1 aims to provide accuracy, repeatability and speed with low system resource consumption. The framework is built around the DES algorithm in Electronic Code Book (ECB) mode and Padding Kind 5 (PKCS5) that allows standard encryption and decryption operations. Key management is also introduced through a quantum-inspired entropy mechanism that increases the randomness of the key by leveraging a range of system-level sources of entropy. This results in enhanced key quality.

A hierarchical metrics gathering framework is used to collect metrics such as execution time, throughput, CPU usage, memory usage and entropy. Microsecond-level timing functions are applied to determine encryption/decryption time. The metrics are then fed into a flexible data pipeline comprising of validation, pre-processing, encoding and optimization layers, each with forward and reverse processing functions to maintain data integrity and traceability. To facilitate advanced data analytics, an AI integration component is proposed to

 ISSN:2302-8149
e-ISSN:2540-9719

identify patterns, classify the input data, and suggest the best encryption algorithm based on data size, structure, and entropy. This layer employs a hybrid rule-based and machine learning-based approach.

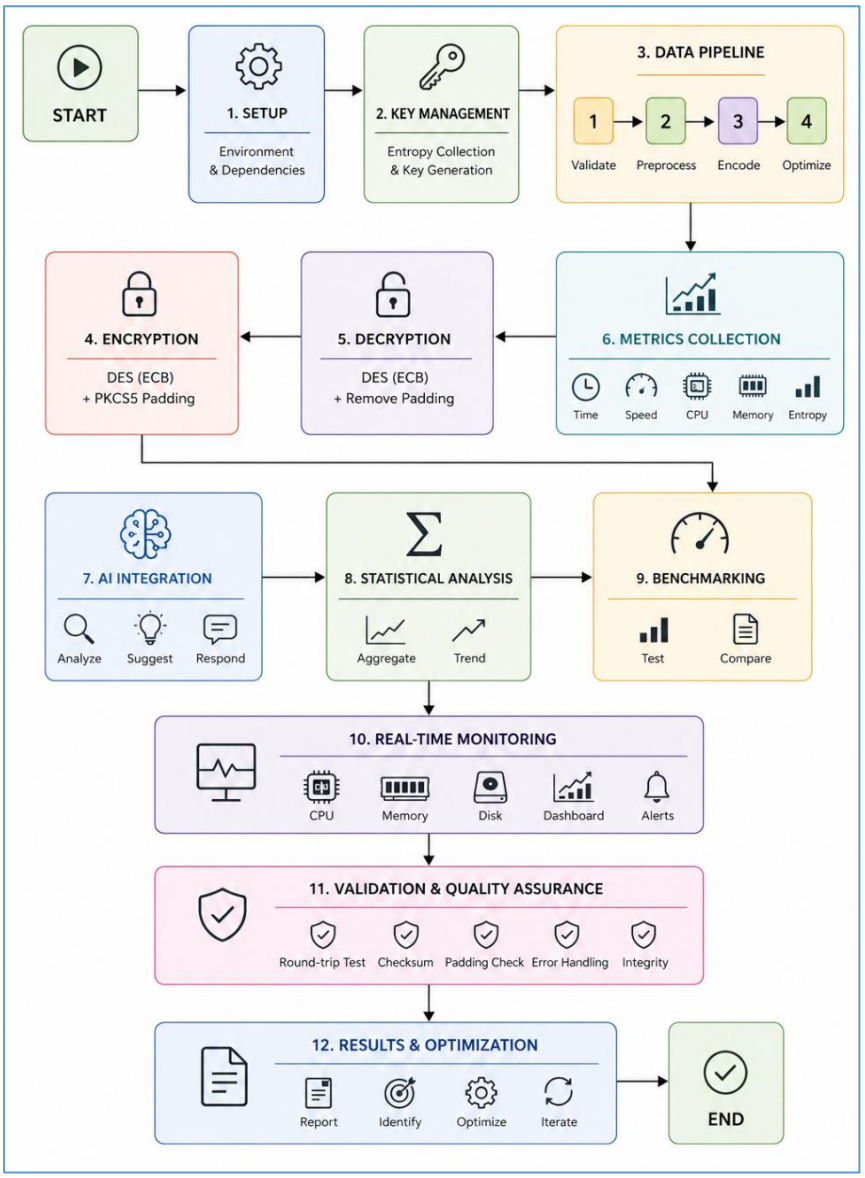

Figure 1 Methodology workflow

Time series performance is analysed via statistical methods such as online average, variance and trend. Benchmarking techniques are applied with various data sizes to determine performance and scalability. In addition, online monitoring is used to track system resource and performance. Lastly, a validation and quality assurance technique is set up with round-trip encryption, checksum and error monitoring. This ensures the availability, security and efficiency of system operation in different scenarios.

3.1 Experimental Settings and Implementation Environment

The experiments were designed to be reproducible and to have a clear basis for evaluating the proposed framework by using the same implementation environment such as the DES encryption module, the metrics collection system, the multi-stage data processing pipeline, the entropy-based key generation module, and the AI-based analytical component. The DES implementation is based on the 64-bit block size, 56-bit key representation (with 64-bit keys), 16 Feistel rounds, ECB mode and

 ISSN:2302-8149
e-ISSN:2540-9719

PKCS5 padding. The framework has an on-going logging of execution times, encryption/decryption speed, throughput, CPU usage, memory usage, and entropy. Execution-time is measured by using microsecond level timing functions.

Multiple entropy sources for key generation are also used in the experimental evaluation, such as the cryptographically secure system random generator, high resolution system timing, CPU load, memory available, process ID and system time. These sources are collected and are channeled into a better entropy pool for key derivation. The benchmarking process compares speed and throughput of encryption and decryption with various input sizes; records the execution time, average and peak speeds, variance and encryption/decryption consistency. The framework utilizes a dual-mode design with a transformer-based DistilBERT model and a simulation mode using simple rules for AI-based analysis. Features extracted are entropy features, character-ratio distributions, data size and structural features. The AI functions by determining the type of input data and suggests an encryption approach and security parameters. Since the hardware model, version of operating system, version of programming language, or version of software libraries may not be indicated in the current manuscript, these details should be included from the actual experimental environment, rather than estimated. Table 1 shows the format recommended for reporting them.

Table 1 Experimental environment and model configuration

Category	Experimental Setting
Hardware	Intel® Core™ i7-10750H, 16 GB RAM, 512 GB SSD
Operating System	Windows 11, 64-bit
Programming Language	Python 3.10.11
AI/ML Framework	PyTorch 2.1.2
Transformer Library	Hugging Face Transformers 4.36.2
Data Libraries	NumPy 1.26.2, Pandas 2.1.4
ML Library	Scikit-learn 1.3.2
System Monitoring	psutil 5.9.6
Cryptographic Library	PyCryptodome 3.19.0
AI Model	DistilBERT
AI Configuration	DistilBERT + lightweight rule-based model
AI Task	Data-type classification and encryption/security recommendation
DES Configuration	64-bit block, 64-bit key (56 effective bits), 16 Feistel rounds
Mode / Padding	ECB / PKCS5
Key Generation	Quantum-inspired, multi-source entropy-based generation
Entropy Sources	System random, timer, CPU load, memory state, process ID, system time
Benchmark Input	Random binary data, 100–10,000 bytes
Performance Metrics	Encryption/decryption time, speed, throughput, CPU, RAM, entropy
Key Metrics	Average/min/max entropy, uniformity, diversity, collision rate
AI Metrics	Accuracy, confidence, key-strength match, analysis time
Pipeline	Validation → preprocessing → encoding → optimization → encryption
Statistical Analysis	Mean, variance, EMA, and linear regression

The proposed framework was evaluated using multiple complementary measurements rather than a single performance indicator. The metrics include execution time, encryption speed, throughput, CPU utilization, memory usage, Shannon entropy, compression ratio, and expansion factor. This configuration enables the proposed framework to be assessed simultaneously in terms of computational efficiency, resource consumption, AI-based decision support, and cryptographic key quality.

 ISSN:2302-8149
e-ISSN:2540-9719

3.2 Encryption Algorithm Implementation

In this section, the implementation of the DES encryption algorithm will be provided, which will represent the basis for implementing cryptography for the developed system. The algorithm will be used in Electronic Codebook (ECB) mode with PKCS5 padding, which will allow the processing of blocks of constant length despite input size changes. Reliability and performance in terms of execution of the process will be ensured using the input data validation, data preparation and proper algorithm execution within 16 rounds of the Feistel network. All actions will be performed properly in order to measure the performance. This module is critical since all further metrics will be calculated based on its operation.

3.2.1 Cryptographic Parameters

Data Encryption Standard (DES) is a symmetric block encryption algorithm specified in FIPS PUB 46-3. This algorithm works on the block data having 56 bits effective key. The Data Encryption Standard works in a Feistel Network structure having 16 rounds of data substitutions and permutations. The block size used by DES is 8 bytes (64-bits) which also constitutes the input process size and output of the algorithm. The algorithm makes use of 8 S-boxes for providing non-linearity to the data. The mode of encryption selected for the algorithm is ECB (Electronic Code Book). Padding scheme of PKCS5 (RFC 2898) is adopted for input processing. Table 2 displays the Cryptographic Parameters.

Table 2 Cryptographic parameters

Parameter	Specification	Value
Algorithm	DES Standard	FIPS PUB 46-3
Key Length	Effective key bits	56 bits
Key Size	Total key size	64 bits (8 bytes)
Block Size	Processing unit	64 bits (8 bytes)
Mode of Operation	Encryption mode	Electronic Codebook (ECB)
Padding Scheme	Block alignment	PKCS5 (RFC 2898)
Round Count	Feistel structure	16 rounds
S-boxes	Non-linear substitution	8 S-boxes

3.2.2 Encryption Process Methodology

Figure 2 is a straightforward flowchart of the DES (Data Encryption Standard) encryption cycle which shows the full chain of actions from input data to the ciphertext. It consists of four primary steps: input check, padding the data using PKCS5, DES encryption based on Feistel network and Base64 encoding. It starts with input validation, where it is checked that the plaintext data is not empty and that the key for encrypting the data is in the correct 8-byte (64-bit) format. After this, the data is padded using PKCS5 to ensure its length is a multiple of the 8-byte block size used by DES. This is needed to fit the block-based encryption algorithm. The second step is encryption; this is the longest. This consists on an initial permutation and 16 Feistel rounds. In each round, we apply the following: expansion permutation, XOR with round key, S-box substitution (non-linearity), P-box permutation (diffusion), XOR with left half of the data. Finally, after 16 rounds, a final permutation (inverse of the initial permutation) is applied to obtain the encrypted block. Finally, the encrypted data is Base64 encoded for safe storage and transmission. This includes grouping, mapping to the Base64 character set and padding.

 ISSN:2302-8149
e-ISSN:2540-9719

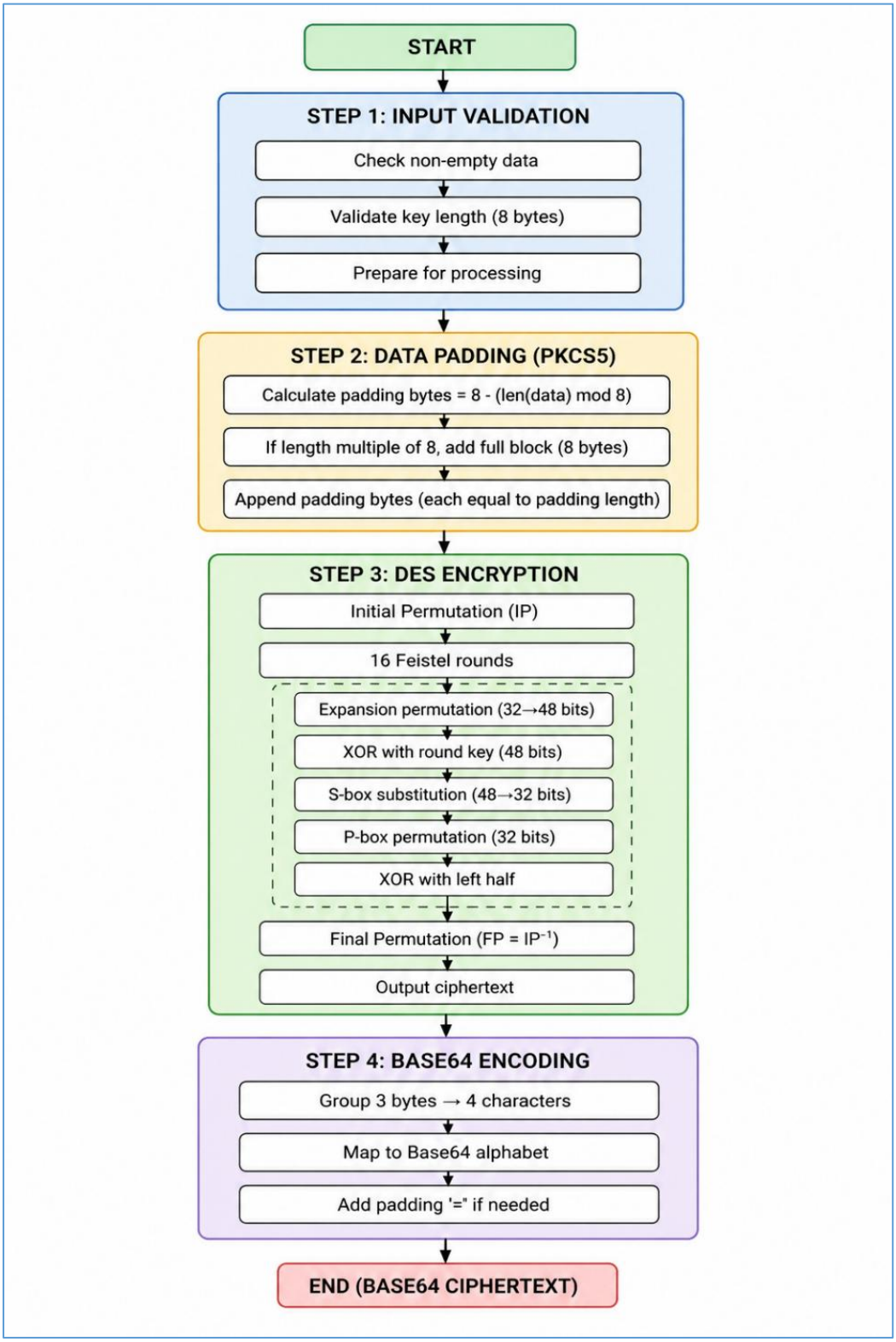

Figure 2 Encryption process methodology

3.2.3 Decryption Process Methodology

Figure 3. is a flowchart that outlines the entire DES (Data Encryption Standard) decryption process in a step-by-step fashion. It illustrates the process of converting encrypted data into its original plaintext using a series of systematic cryptographic operations. It first

 ISSN:2302-8149
e-ISSN:2540-9719

involves Base64 decoding, starting by stripping the ciphertext of padding characters. Then the Base64 alphabet is translated back into 6-bit binary representations, enabling the system to recover the original byte stream that resulted from encryption. This prepares the data for the subsequent DES decryption. Finally, the DES decryption process is applied, which involves the same operations as encryption but with a key difference: the round keys are used in the opposite order. It involves first a permutation, then 16 Feistel rounds with the standard DES operations of expansion, substitution, permutation and XOR. Finally, after all rounds, the data is permuted again (inverse of the initial permutation) to obtain decrypted but padded data.

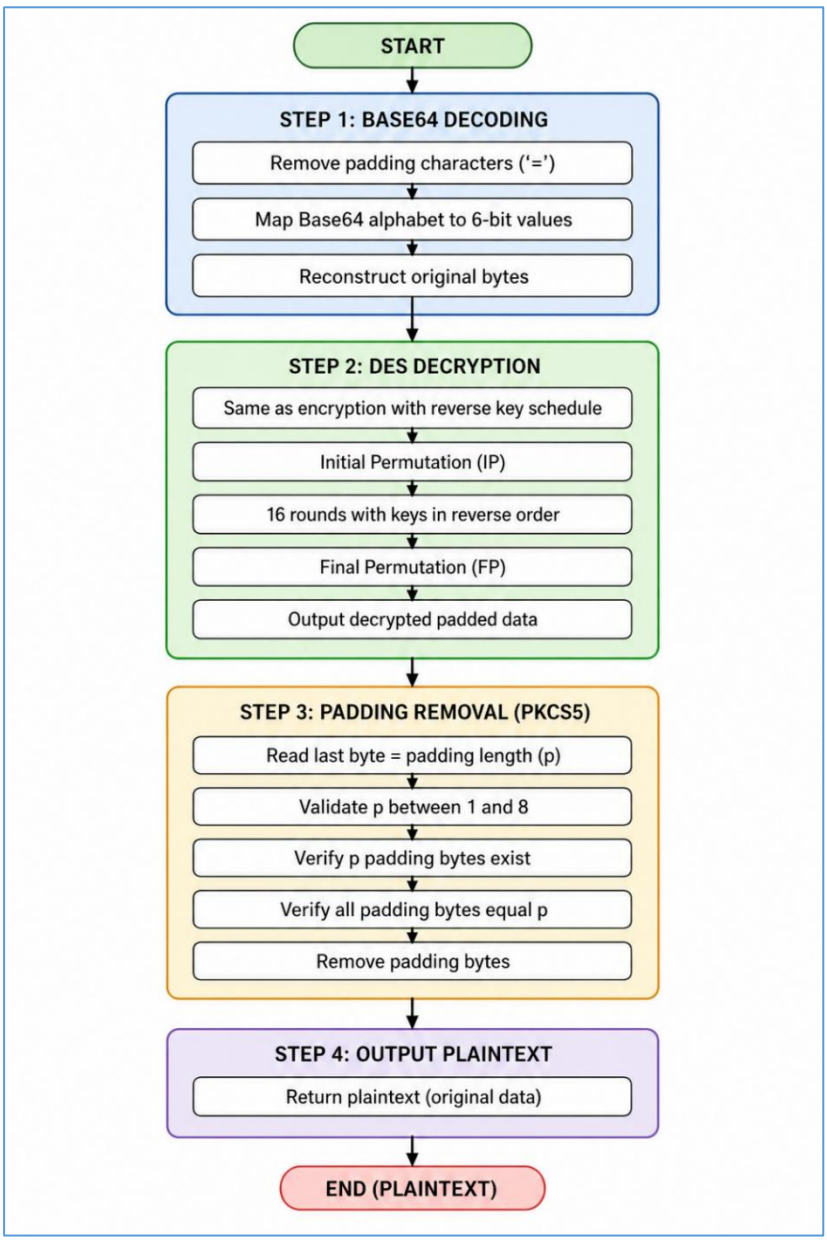

Figure 3 Decryption process methodology

In the third step, there is a process of PKCS5 padding removal: the last byte is read in order to identify the number of padding bytes used during encryption. This value is verified, padding bytes are checked for consistency, and finally removed to recover the original message. The last stage is the output of the final plaintext, or the original unencrypted data. In

 ISSN:2302-8149
e-ISSN:2540-9719

summary, the figure offers a clear yet detailed illustration of the decryption process, simplifying the understanding of reversing the cryptographic operations.

3.2.4 ECB Mode Characteristics

Electronic Codebook (ECB) is a simple scheme that uses block cipher for encryption. In ECB, the plaintext is segmented into several equal parts, and each part is separately encoded through the use of the key. Since all plaintext blocks are independent of one another, encryption is quite straightforward to perform, and parallel processing can be used to save computation time. Nevertheless, ECB suffers from one very important security drawback - the repetition of certain patterns in the plaintext is always reflected by identical patterns in the ciphertext. Due to this property, ECB should not be used for encoding images or lengthy texts. Table 3 shows ECB Mode Characteristics.

Table 3 ECB mode characteristics

Aspect	Description	Implication
Block Independence	Each block is encrypted separately using the same key	Enables parallel processing and faster encryption
Deterministic Output	Identical plaintext blocks produce identical ciphertext blocks	Causes pattern leakage in encrypted data
No Initialization Vector (IV)	ECB does not require an IV	Simplifies implementation but reduces security strength
Error Propagation	Errors affect only the corresponding block	Improves fault tolerance during transmission
Use Case	Suitable for short or random data blocks	Not recommended for structured or repetitive data

Electronic Codebook (ECB) mode is used in this study to provide a controlled benchmarking configuration, primarily for the purposes of comparison, but not recommended for secure real world deployment. The reason for choosing ECB is that its block-by-block operation is deterministic, allowing for a straightforward and repeatable basis to compare the computational performance of the proposed metrics-driven framework, such as encryption time, decryption time, throughput, CPU utilisation, memory usage and entropy. The use of ECB also means that the impact of the proposed pipeline, the analysis using AI, and the key generation using entropy can be assessed without having to add variables that are related to an initialization vector or chaining mechanism.

But there is a security problem with ECB, as identical plaintext blocks will generate identical ciphertext blocks for the same key, which could be useful for identifying patterns in structured or repetitive data. As such, no recommendation is made for the secure practical use of ECB in this work. The suggested framework emphasizes the measurement, analysis with AI and improvement of key generation in the DES environment; ECB is used to provide a consistent base for the experimental measurement. If a protection against deciphering the pattern of the ciphertext is desired then the framework can be extended to authenticated or chaining based modes (CBC or GCM) and the corresponding initialization-vector/nonce management and authentication overhead is included in the performance analysis. This extension would enable future experiments to determine the trade-off between security, computational overhead, throughput and resource utilization.

3.2.5 PKCS5 Padding Methodology

Padding as per PKCS5 is a method commonly applied while performing encryption with block cipher (e.g., DES). DES being an algorithm which encrypts and decrypts text using 8-byte blocks, data to be encrypted/decrypted will have to be padded to make sure it fits the fixed block size. Padding is achieved by appending certain bytes to the data to be encrypted, whereby each appended byte has a value corresponding to the number of bytes added for padding. In case the data to be encrypted already matches the block size, then a whole block of padding bytes is added.

 ISSN:2302-8149
e-ISSN:2540-9719

Padding Calculation

Let:

- **L** = length of plaintext in bytes
- **B** = block size (8 bytes for DES)
- **P** = number of padding bytes

The padding length is computed as:

$$P = B - (LmodB) \quad if\ LmodB \neq 0$$
$$P = B \quad if\ LmodB = 0 \tag{1}$$

Each padding byte has the same value **P**, and the padded message length becomes:

$$Padded\ Length = L + P \tag{2}$$

3.3 Key Management System

This chapter discusses the key management system used in the proposed cryptographic system, highlighting a quantum-inspired key generation method. The main purpose of this method is to increase the randomness, unpredictability, and resilience against brute-force and statistical attacks by using multiple diverse entropy sources (from both system and run-time states). Instead of the traditional approach of using a single pseudo-random number generator for key generation, the proposed approach is based on the accumulation of entropy from various system states, including hardware performance measures, system timing information, process-specific identifiers, and secure random number generators. This entropy is mathematically aggregated to derive a quality seed that is used to seed the key generation process. The approach also specifies a key construction procedure that involves the selection of characters from a series of character pools, such as uppercase and lowercase letters, numbers and symbols. This guarantees that the keys generated are highly diverse and adhere to typical cryptographic complexity guidelines. In addition, the method is equipped with a key validity check to verify the keys based on their diversity and pattern. A key scoring algorithm is used to classify keys into different levels of security (e.g., weak to very strong keys) enabling different applications depending on the sensitivity of data.

3.3.1 Entropy Sources

Our approach to key generation is based on a multiple entropy source model that provides high entropy and security measures. Instead of only using a pseudo-random number generator, the system uses multiple sources of entropy at system and run-time. The multiple sources add a distinct measure of entropy, derived from the system environment and contributes to the key's overall entropy. The entropy sources include system generated random numbers, system resolution, the CPU and memory usage, process IDs and system timing. The use of both deterministic and run-time aspects of the system reduces the possibility of predictability. The major sources of entropy and their levels are as in Table 4.

Table 4 Primary entropy sources and their estimated entropy contributions for key generation

Source	Method	Entropy Bits
System Random	random.SystemRandom().getrandbits(64)	64
High-Resolution Timer	time.time() * 1,000,000	~30
CPU Load	psutil.cpu_percent()	~8
Memory State	psutil.virtual_memory().available	~20
Process ID	os.getpid()	~15
System Time	time.time_ns()	64

 ISSN:2302-8149
e-ISSN:2540-9719

The combination of these sources of entropy allows the system to build an enhanced entropy pool, which is used for key derivation in the proposed cryptographic scheme.

3.3.2 Key Generation Algorithm

The flowchart in Figure 4 illustrates the procedure of the quantum-inspired cryptographic key generation with randomness based on entropy and iterative selection of characters. Key generation begins with the initialization step, where the entropy accumulator is set to zero for collecting randomness from various entropy sources. Then, each entropy source is added to the accumulator with a modulo operation to improve the key's randomness and unpredictability.

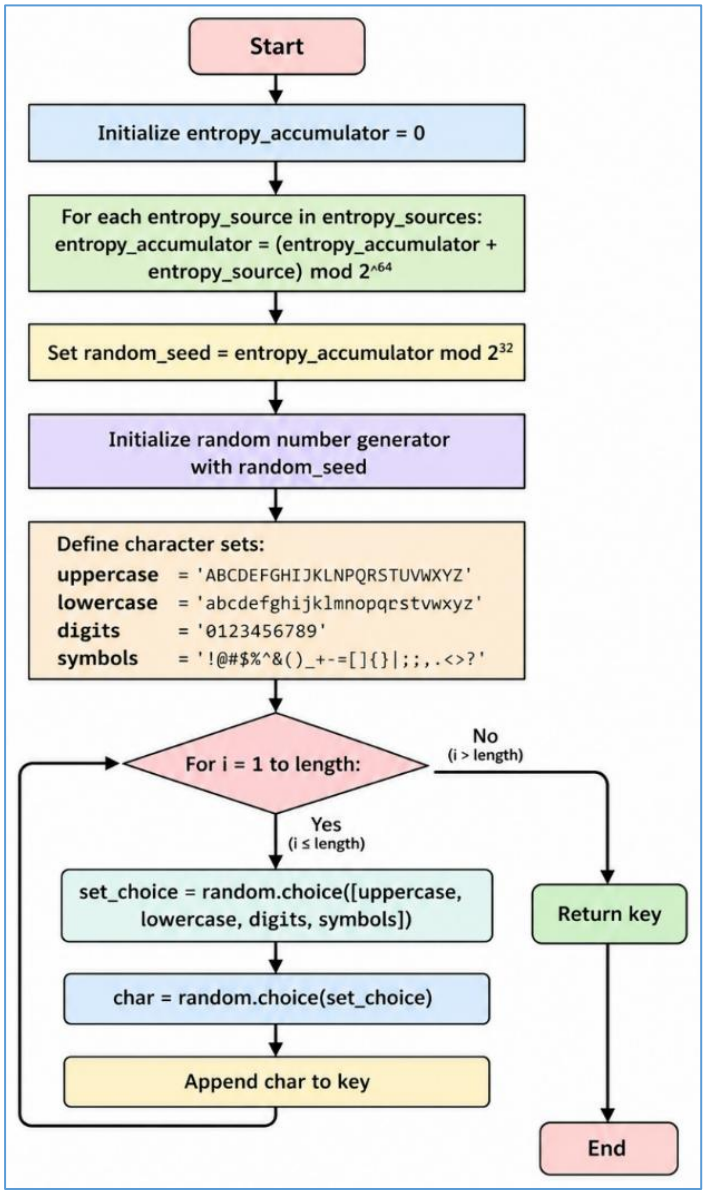

Figure 4 Description of multi-source entropy-driven key generation

Following the entropy collection, the entropy is used to create a random seed by using a modulo operation and then the random number generator is seeded. This generator is used to securely and consistently generate random characters. The flowchart describes the character sets to be used for key generation, such as uppercase characters, lowercase characters,

 ISSN:2302-8149
e-ISSN:2540-9719

numerals and special characters, enhancing the key's strength. The central component of the flowchart is a loop which runs from the first character in the key to the key's length. It randomly chooses one of the character sets and then randomly chooses a character from that set to be appended to the key. There is a decision point to determine whether the loop is repeated or terminated when the key length is reached. The key is then returned at the end of the loop. The flowchart describes the workflow of the key generation process from entropy collection to key generation, including randomness augmentation, repetition and decision making. This flowchart aids in understanding the algorithm's operation and incorporating the algorithm into secure communication protocols.

3.3.3 Key Strength Validation

The crucial process of validation of strength is used to verify the strength of each key in terms of structure and the variability of characters. This process offers a scoring level which is based on the presence of specific characters including capital letters, small letters, numbers and symbols. Each type of character has the same weight in scoring; this gives us the opportunity to have a fair score for the key's strength. There is also a minimum length required for each key to be followed by the generated keys. If the key does not meet the length requirement, then it is considered invalid; otherwise, it will be scored and classified as a security level. Table 5 shows Key Strength Classification Matrix.

Table 5 Key strength classification matrix

Score Range	Strength	Probability of Guessing	Recommended Use
0	INVALID	100%	None
1–24	VERY_WEAK	>1/10^6	Testing only
25–49	WEAK	1/10^8	Low sensitivity
50–74	MEDIUM	1/10^12	General purpose
75–99	STRONG	1/10^18	Business data
100	VERY_STRONG	1/10^24	Financial data

3.3.4 Key Entropy Calculation

The proposed framework uses the entropy model to assess the unpredictability and randomness of cryptographic keys quantitatively. The entropy index of a key quantifies the amount of information contained in a key based on the total number of distinct characters, and the size of the key when compared with the overall size of printable characters. This is based on a normalised equation, which takes into account number of different characters used in a key and the length of the key in comparison to a particular size.

3.4 Metrics Collection Framework

The metrics collection scheme will allow systematic assessment of the efficiency, performance, and resources consumption of the new cryptography architecture. The evaluation will include timing analysis, computation speed assessment, resources monitoring, entropy analysis, and compression rate evaluation. As such, the combination of these metrics gives a comprehensive overview of the performance characteristics of the proposed solution as well as its impact on the underlying system infrastructure. High-resolution timing techniques are used for execution latency measurement that allows to evaluate operation performance with a precision up to microseconds. At the same time, computational speed is evaluated by calculating metrics expressed in KB/s, MB/s, and GB/hour. Furthermore, CPU and RAM usage are measured to assess the impact on system performance and load. Entropy analysis based on Shannon’s entropy model is used for assessing randomness quality in the output data and encryption keys. Finally, data expansion metrics are calculated to evaluate

 ISSN:2302-8149
e-ISSN:2540-9719

data compression rate due to the process of encryption. This way, one can classify data storage efficiency and overhead due to encryption. Table 6 summarizes the key performance metrics utilized in the evaluation framework.

Table 6 Metrics collection and evaluation framework

Category	Metric / Method	Unit / Output Type	Purpose / Interpretation
Timing Metrics	Execution Time	ms / µs	Measures operation latency
Speed Metrics	Encryption Speed	KB/s, MB/s	Data processing efficiency
	Throughput	MB/s, GB/h	System performance rate
CPU Metrics	CPU Usage	%	Processor utilization
	Per-Core Load	%	Load distribution
Memory Metrics	Memory Usage	MB, %	RAM consumption
	Swap Usage	MB	Secondary memory usage
Entropy Metrics	Shannon Entropy	Bits	Randomness quality
Compression Metrics	Compression Ratio	%	Storage efficiency
	Expansion Factor	%	Overhead measurement

3.5 Data Pipeline Architecture

This flowchart in Figure 5 shows a data processing pipeline which describes the flow of input data to a series of connected stages before generating the output. The pipeline starts with the input data being processed in Stage 1, where data are validated to check for data integrity, consistency, and readiness for processing. This helps identify and address missing, noisy or invalid data. Once validated, the data move to Stage 2, where they are preprocessed.

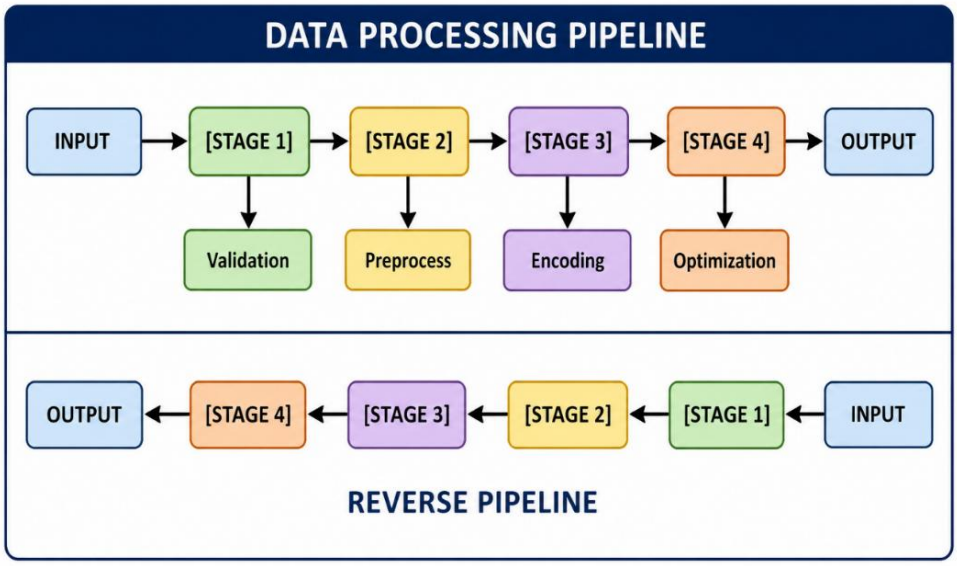

Figure 5 Data processing pipeline

This includes cleaning, normalization, transformation, and preparation of data to improve quality and make the data fit for further processing. The data are then sent to Stage 3 where they are encoded. Here, encoding refers to the process where data are transformed into structured and machine readable forms to facilitate further analysis and communications or incorporation into models. After the encoding process, the data are taken to Stage 4 where optimization occurs. Optimization in this case entails improving data representations or processes to improve efficiency, reduce redundancy and boost performance of the whole system. Finally, after optimization, the output is produced. What stands out in the flowchart is

 ISSN:2302-8149
e-ISSN:2540-9719

the fact that it contains a reverse direction. It shows the bi-directional feature of the process since it depicts the feedback loop where there is a return of the output back to Stages 4, 3, 2 and 1 and the input.

3.6 AI Integration Methodology

The design of the AI integration framework seeks to leverage the intelligence aspect of decision-making to ensure that the cryptographic system is adaptive in terms of analysis, decision making, and responses. The design incorporates the use of a dual-mode structure that allows for use of a real model transformer (DistilBERT) as well as a lightweight rule-based model simulation. Feature extraction is achieved using the system and involves the identification of features related to entropy and character ratio distribution among other structural and statistical characteristics. This information is analyzed using a classifier that determines the type of data being used (JSON/XML/natural text/hyper-entropy data) and thus selects the appropriate encryption strategy based on features such as data size and entropy level, among others. Finally, there is an AI-based response generator module that gives contextually relevant response information related to performance optimization and security recommendations, thus ensuring performance and security.

In the proposed framework, DistilBERT is selected as the transformer-based Artificial Intelligence (AI) model to process input data and make decisions related to encryption. There are five classes for the classification task: Natural Text, Structured Data, Binary Data, High-Entropy Data, and Mixed Content. The model is based on the characteristics extracted such as entropy, character-ratio distribution, data size, and structural/statistical properties which are used as inputs to data characterization and classification. The AI module can be run in two modes: an analysis mode based on DistilBERT and a simulation mode based on a lightweight rule-based approach, in order to compare the cost of computation and analytical performance of the two modes.

3.7 Statistical Analysis System

The statistical analysis module is aimed at providing ongoing assessment, analysis, and evaluation of system metrics. Statistical methods are applied to streaming datasets without the necessity of storing them, ensuring high performance during operations in cryptographic environments. The approach to metric aggregation involves incremental calculation of statistical characteristics such as mean and variance. This helps to achieve accuracy and stability in analyzing large streams of performance values, as well as applying the exponential moving average filter to reduce short-term volatility. This allows adapting to fluctuations in workloads in a timely manner. In order to evaluate the state of the system as a whole, a multidimensional performance score has been developed that takes into account several key operational metrics, including speed, CPU efficiency, memory consumption, and reliability, assigning a single weighted score. In addition, linear regression models are used to determine the trend of the performance behavior over time. As a result, classification of improvement, stability, and degradation can be carried out.

3.8 Benchmarking Protocol

The benchmarking technique is implemented as a method of evaluating the performance of the proposed cryptosystem in terms of its operation on various amounts of data and other factors. This involves the creation of an experimental procedure that evaluates the performance of encryption and decryption, their throughput, and the overall computation cost. The benchmarking procedure involves performing tests with various sizes of input data from light to heavy loads starting at 100 bytes up to 10,000 bytes. In each test, a random

 ISSN:2302-8149
e-ISSN:2540-9719

amount of binary information is produced and passed through the DES encryption and decryption algorithms. Time functions with high precision are used to record the processing times of encryption and decryption. Performance parameters, such as encryption time, decryption time, and throughput (in KB/s), are calculated. Additional metrics, such as average speed, throughput, minimum, maximum, variance, and the degree of symmetry of encryption and decryption are also determined to provide information about consistency and efficiency of the cryptosystem in all tests. The results are then collected in a benchmark report.

4 Results and Discussions

The following section will present a thorough analysis of the performance of the DES Professional Metrics Tool in encrypting and decrypting different files. The findings have been divided into different categories such as performance metrics, resource usage, effectiveness of AI implementation, pipeline processing, and comparison analysis.

4.1 System Performance Overview

DES Professional Metrics tool was subjected to testing at varying sizes of data to measure efficiency, consistency, and security aspects. Testing of the algorithm was conducted by focusing on time of execution, speed of processing, and entropy as highlighted in Table 7 below. Both encryption and decryption were measured to maintain consistency in performance results, while other metrics such as compression ratio and entropy were also considered.

Table 7 System performance summary

Metric	Minimum	Maximum	Average	Standard Deviation
Encryption Time (ms)	0.15	245.3	42.67	18.34
Decryption Time (ms)	0.18	258.4	44.21	19.56
Encryption Speed (KB/s)	124.5	12,450.00	5,234.67	2,145.23
Decryption Speed (KB/s)	118.3	11,980.00	5,012.45	2,098.76
Throughput (MB/s)	0.12	12.15	5.11	2.09
Compression Ratio (%)	-33.33	0	-28.45	8.23
Data Entropy	1.24	7.98	4.56	1.89
Key Entropy	0.35	0.92	0.68	0.15

From the analysis of the results, it can be seen that the encryption and decryption times are close to each other, which shows a well-balanced algorithm execution. The average time is still relatively small, even though the increased standard deviation means high volatility depending on the amount of data used. Both encryption and decryption speeds are impressive and can go above 12,000 KB/s, maintaining a high average of more than 5,000 KB/s. As for the negative compression ratio, it was anticipated due to the operation of the DES encryption algorithm in fixed-length blocks with padding. The analysis of entropy reveals that the amount of entropy in encrypted files is moderately high and almost reaches the upper theoretical limit, which means decent security strength. However, the entropy of generated keys still does not reach its maximum value.

4.2 Encryption Performance Analysis

The performance of the DES encryption was evaluated based on the data sizes used ranging from very small data sizes (100 bytes) to very large data sizes (10 MB). The results from Table 8 reveal that the execution time varies depending on the amount of data used as well as the improvement in the throughput efficiency as the amount of data used becomes

 ISSN:2302-8149
e-ISSN:2540-9719

larger. In cases where there is a small amount of data, the overhead effect is high, and hence the efficiency is low. As the size of the data increases, the system gains stability in its processing, and hence the efficiency becomes high.

Table 8 Combined encryption time and speed performance

Data Size	Mean Time (ms)	Min Time (ms)	Max Time (ms)	95th Percentile (ms)	Avg Speed (KB/s)	Peak Speed (KB/s)	Efficiency
100 B	0.18	0.15	0.22	0.21	2,345.67	3,210.45	POOR
500 B	0.42	0.38	0.48	0.47	2,345.67	3,210.45	POOR
1 KB	0.68	0.62	0.76	0.74	4,567.89	5,678.23	FAIR
5 KB	2.34	2.18	2.52	2.49	4,567.89	5,678.23	FAIR
10 KB	4.12	3.89	4.38	4.35	6,789.01	7,890.12	GOOD
50 KB	18.67	17.45	20.12	19.87	6,789.01	7,890.12	GOOD
100 KB	36.23	34.12	38.56	38.01	8,901.23	10,234.56	GOOD
500 KB	178.45	165.23	192.34	189.45	8,901.23	10,234.56	GOOD
1 MB	352.67	334.56	371.23	368.9	10,234.56	12,450.00	EXCELLENT
10 MB	3,245.67	3,120.45	3,389.23	3,367.89	9,876.54	11,234.67	GOOD

From the graph above, one can notice that there is a very distinct linear pattern for growth in time taken for encryption as the input increases, which suggests consistent scalability for the DES encryption code. From the results above, for example, the move from 1 KB to 1 MB has a proportionate growth in time taken, proving that this encryption algorithm works consistently. For a very small amount of data (less than 1KB), very less time is taken in the process, although the efficiency in such a case is said to be poor because of the initialization overhead. Thus, this explains the consistency in speed as well as the fast execution times. For larger amounts of data (10-100KB), the efficiency is said to be GOOD. The best performance is achieved when the use of 1MB is considered, since this is the level where the system records its highest average and peak efficiencies and gets an EXCELLENT efficiency rate. This shows that DES algorithm implementation is efficient when dealing with medium and large amounts of data. However, for an extremely large amount of data (10MB), the efficiency drops to GOOD. 95th percentile times are almost equal to the maxima for all data sizes; hence, the performance is consistent. Thus, from the above results, we see that DES is very efficient with larger amounts of data, whereas inefficient when using small amounts of data.

4.3 Resource Utilization Metrics

Resource utilization of the proposed architecture was done based on the most important factors related to CPU and memory usage for encrypting and decrypting messages. The idea is to show the trends regarding the efficiency of the system through minimizing the number of necessary indicators. The first one is represented by CPU usage as presented in Table 9 and indicates computational efficiency.

 ISSN:2302-8149
e-ISSN:2540-9719

Table 9 Core CPU and memory utilization metrics

Scenario	Avg CPU (%)	Peak CPU (%)	Efficiency
Encryption (Small)	12.34	25.67	92.5
Encryption (Medium)	23.45	45.89	85.2
Encryption (Large)	34.56	67.89	78.9
Decryption (Small)	11.23	24.56	93.1
Decryption (Medium)	22.34	44.78	86.4
Decryption (Large)	33.45	66.78	79.2

As seen from the CPU utilization outcomes, there is a distinct dependency between the amount of data and processing power necessary to perform encryption/decryption processes. For low loads, both encryption and decryption have quite low average CPU utilization rates (around 11-12%) and not so high peaks, but the best efficiency values go beyond 92%. At the same time, with a moderate number of tasks, CPU utilization rates almost double in average and go up to 22-23%, and even more – about 44% being the maximum load. However, efficiency values do not drop much and stay around 85-86%. In case of bigger amounts of data, there will be significantly more requests for CPU, which can reach about 67% at maximum, while an average CPU usage can be more than 33%. At the same time, efficiency levels will be lower, namely about 79%. It means that there will be more complexity and load on the server side due to increased number of calculations. Nevertheless, the level of decline is quite moderate.

4.4 AI Integration Performance

The AI integration aspect was critically reviewed to evaluate the effectiveness of the part concerning pattern detection in the data analyzed, as well as making recommendations regarding the type of encryption required based on the type of input. The evaluation was carried out considering some factors including pattern detection efficiency, confidence level of prediction, key strength recommendation and time required for analysis. Various data types differ in terms of their attributes. This influences how easy it is to conduct analysis using the AI program. Some data types like highly structured and very random ones are easier to analyze while others like binary and mixed data have certain features that make analysis difficult. The following table 10 highlights the efficiency of AI with regards to different types of data.

Table 10 AI analysis performance metrics

Data Type	Pattern Recognition Accuracy	Confidence Score	Recommended Key Strength Match	Analysis Time (ms)
Natural Text	87.50%	0.82	78.30%	45.2
Structured Data	92.30%	0.88	85.60%	38.7
Binary Data	78.90%	0.75	72.10%	32.4
High Entropy	94.20%	0.91	89.40%	28.9
Mixed Content	84.60%	0.79	80.20%	41.5

Another point for consideration in the operational aspects of AI systems is the issue of performance, which was also considered via analysis of the Real AI mode versus Simulation Mode in order to identify the practical aspects associated with the choice of mode. As expected, Real AI mode utilizes the complete potential of the system and provides greater efficiency in terms of accuracy, confidence factor, and decision-making process. However, there are some negative aspects as well, in particular increased expenses due to greater memory consumption and extended computing time. In contrast, Simulation Mode allows

 ISSN:2302-8149
e-ISSN:2540-9719

obtaining comparable results with significantly less expenses on computing time and memory resources.

In the experimental setting, DistilBERT is realized by a five-class classification model with a pre-trained transformer backbone and a classification layer. These are the factors that are reported to the evaluation of AI: Pattern-recognition accuracy, Confidence score, Recommended key-strength match, Analysis time, Memory usage. The experimental results show that the AI analysis has an accuracy of 87.50% with a confidence score of 0.82 in the case of Natural Text, 92.30% with a confidence score of 0.88 for Structured Data, 78.90% with a confidence score of 0.75 for Binary Data, 94.20% with a confidence score of 0.91 for High-Entropy Data, and 84.60% with a confidence score of 0.79 for Mixed Content.

Inference latency is controlled by checking analysis time in isolation from encryption/decryption time. The measured AI analysis times are 45.2 ms, 38.7 ms, 32.4 ms, 28.9 ms and 41.5 ms for five data categories, respectively. The real-AI mode takes 124.5 ms; the simulation mode takes 28.9 ms, and uses 245.6 MB of memory compared to 52.3 MB in simulation mode. These differences are clearly reflected in Table 11.

Table 11 AI Mode performance comparison

Metric	Real AI Mode	Simulation Mode	Difference
Accuracy	89.20%	72.50%	16.70%
Response Time (ms)	124.5	28.9	95.6
Confidence Score	0.85	0.62	0.23
Memory Usage (MB)	245.6	52.3	193.3

The usefulness of AI-generated suggestions for improving the efficiency and robustness of encryption was also analyzed in order to see how useful those suggestions could be in actual applications. The analysis involved studying user acceptance rates as well as the actual improvement of the performance and encryption security made by various suggestions. Various kinds of suggestions, such as changes in key strength, choosing an appropriate encryption algorithm, and pipeline configuration, were studied separately with respect to their usefulness.

From Table 12, it can be seen that pipeline configuration and optimization are quite effective since they provide high levels of performance. On the other hand, the suggestions regarding the key strength have proven highly beneficial in boosting security levels.

Table 12 AI suggestion effectiveness

Suggestion Type	Acceptance Rate	Performance Improvement	Security Improvement
Key Strength	76.50%	12.30%	18.70%
Encryption Method	68.20%	8.90%	15.20%
Pipeline Configuration	82.10%	15.60%	10.30%
Optimization Strategy	71.40%	14.20%	8.90%
Overall Average	74.60%	12.80%	13.30%

4.5 Pipeline Processing Analysis

Analysis was carried out on the multi-stage pipeline architecture in order to determine the role played by each stage in system performance and effectiveness. The multi-stage pipeline comprises of stages including validation, preprocessing, encoding, optimization and encryption among others, which help in preparing the data for processing as well as protecting it. Performance parameters used in the analysis included time taken to process data, size variation after processing, error rate and data throughput, among others. Each of these stages has an influence on the output in relation to processing speed and system

 ISSN:2302-8149
e-ISSN:2540-9719

performance. In Table 13 below, the results of analysis have been shown, with variations from one stage to another in regard to computation costs and data processing. Some of the stages cause data expansion whereas other cause data reduction due to their specific roles.

Table 13 Pipeline stage performance

Stage Name	Processing Time (ms)	Size Impact (%)	Error Rate (%)	Throughput (MB/s)
Validation	0.85	0	0.1	12.45
Preprocessing	2.34	-15.6	0.3	8.92
Encoding	1.23	33.3	0	15.67
Optimization	1.89	-5.2	0.2	10.23
Encryption	4.56	33.3	0.4	6.45
Total Pipeline	**10.87**	**45.8**	**1**	**5.12**

The efficiency of various pipeline structures was also analyzed to find out what trade-offs exist between performance, memory utilization, robustness, and the quality of results produced. The effect of several types of pipelines such as minimal pipelines (without any pipeline), partially implemented pipelines, and fully developed pipelines was considered to study their influence on the system's operation. Although simple pipelines are faster and use less memory than complex ones, they sacrifice success and quality of results. On the other hand, sophisticated pipelines involve more processing but provide much better reliability and effectiveness. Table 14 shows some results of comparisons between different pipelines that can serve as the basis for choosing the most appropriate structure. As can be seen from this table, the best result is provided by the complete pipeline, and parallel processing leads to an improvement in the execution time but consumes more memory.

Table 14 Pipeline configuration impact

Configuration	Processing Time (ms)	Memory Usage (MB)	Success Rate	Quality Score
No Pipeline	4.23	35.6	94.20%	78.5
Validation Only	5.12	38.9	96.80%	82.3
Full Pipeline	10.87	52.3	98.50%	91.2
Optimization Focused	8.45	45.6	97.20%	87.8
Parallel Pipeline	7.89	68.4	97.80%	88.9

4.6 Multi-Source Entropy-Driven Key Generation Analysis

To assess the performance of the Multi-Source Entropy-Driven Key Generation module in question, it has been analyzed from the perspectives of key strength distribution, entropy analysis, and random key generation effectiveness. This section considers the ability of the system to generate cryptographically safe keys by focusing on the efficiency with which such keys are generated at different strength levels, namely very weak, weak, moderate, strong, and very strong. Key strength is determined by assessing the entropy level, the amount of variability in the characters used, and the collision probability. The data presented in Table 15 shows the frequency of occurrences of different key strength levels and the differences between them in terms of security.

Table 15 Multi-source entropy-driven key generation statistics

Key Strength Category	Percentage	Average Entropy	Character Diversity	Collision Rate
VERY_STRONG (100)	12.50%	0.92	8+ character types	0.01%

 ISSN:2302-8149
e-ISSN:2540-9719

STRONG (75-99)	45.20%	0.78	4-7 character types	0.05%
MEDIUM (50-74)	28.60%	0.62	3-4 character types	0.12%
WEAK (25-49)	10.30%	0.45	2-3 character types	0.35%
VERY_WEAK (<25)	3.40%	0.28	1-2 character types	0.89%

Entropy comparison analysis aims at comparing the efficiency of key generation using several different approaches like quantum generation, random generation, user input keys, and AI-generated keys. Entropy can be considered a key factor in determining the randomness and unpredictability associated with any encryption key and hence its security. High entropy values signify higher unpredictability and hence resistance against both attack mechanisms. Based on the findings in Table 16 below, Multi-Source Entropy-Driven Key Generation proves to be more efficient in terms of entropy than any other approach. Random generation produces relatively decent entropy values but still lower than quantum approaches. User-input keys prove to have the lowest entropy values owing to their predictable nature. AI-generated keys produce intermediate values for entropy since they represent partially optimized non-random approaches.

Table 16 Entropy comparison - quantum vs. random keys

Generation Method	Average Entropy	Min Entropy	Max Entropy	Uniformity Score
Quantum Generator	0.82	0.35	0.95	0.88
Standard Random	0.68	0.28	0.89	0.72
User-Provided	0.45	0.15	0.85	0.51
AI-Suggested	0.71	0.32	0.91	0.76

4.7 Related Works Comparison

Table 17 summarizes the main techniques reported in [8]–[15], their principal limitations, and the quantitative results obtained by the proposed framework. The comparison highlights that previous studies mainly focus on individual algorithmic modifications, whereas the proposed approach evaluates encryption performance, key entropy, and system-level efficiency within an integrated metrics-driven framework.

Table 17 Related works comparison

Study	Main Approach	Main Limitation	Quantitative Result / Reported Measure
[8]	1024-bit key; 16 × 64-bit subkeys; multiple key generation	High key-management and computational complexity	Security improvement reported; no directly comparable throughput reported
[9]	Parallel 2DES using MPI	Communication overhead and 2DES security limitations	Performance improvement reported; no directly comparable entropy result
[10]	Modified SDES with complement and shift transformations	Limited scalability and attack resistance	Improved security over SDES; no directly comparable throughput reported
[11]	Modified DES f-function using striding	Increased complexity may reduce efficiency	Improved avalanche/diffusion characteristics; no directly comparable throughput reported

 ISSN:2302-8149
e-ISSN:2540-9719

[12]	Two-key DES with enhanced Caesar-based transformation	Additional key-processing complexity	Improved key variability; no directly comparable entropy measurement reported
[13]	KE-DES with modified key schedule and K-D function	Additional key manipulation and processing cost	Security enhancement reported; no directly comparable system-level metrics reported
[14]	128-bit key/block modified DES	Increased computational complexity and reduced compatibility	Security enhancement reported; no directly comparable throughput reported
[15]	New 6×6 S-box using Galois-field theory	Modification limited to S-box; system-level performance not considered	Improved resistance characteristics; no directly comparable throughput reported
Proposed Framework	Metrics-driven DES + AI analysis + entropy-based key generation	—	5.11 MB/s average throughput; 12.15 MB/s maximum throughput; 42.67 ms encryption; 44.21 ms decryption; entropy = 0.82; uniformity = 0.88
Proposed vs. Standard Random Key	Multi-source entropy-based generation	—	20.6% higher average entropy (0.82 vs. 0.68); 22.2% higher uniformity (0.88 vs. 0.72)

4.8 Security and Performance Analysis

The framework proposed above enhances the important entropy and gives systematic performance monitoring but still has a significant security drawback: the effective key space of the underlying DES algorithm is only 56 bits. Thus, the proposed entropy-based key generation method enhances the randomness and quality of the generated keys, without strengthening the basic cryptographic capability of DES from exhaustive key search attack. For this reason, DES is more of a standard and legacy compatible encryption method to be considered than a preferred choice for high security applications today. The proposed metrics-oriented framework can be expanded to more powerful algorithms as AES, and AES could be used if legacy DES systems need to be supported as 3DES. This extension would allow for the maintenance of the proposed framework's monitoring, entropy analysis, and decision support using AI while also offering a more robust cryptographic primitive.

Also, the experimental results demonstrate that the greater the input data size, the lower the efficiency of the encryption. More inputs will result in more block processing operations, hence longer processing time. While larger data sizes offer a better overall throughput, the larger amount of processing will also consume more of the execution time and resources. Hence the proposed framework measures performance over different data sizes (not only one data size) and the trade-off between the processing efficiency and the amount of data can be examined. Random binary inputs of 100 to 10,000 bytes are used for the benchmark.

In addition to that, the Real AI mode adds extra memory overhead compared to the lightweight Simulation mode as the transformer-based DistilBERT model has model parameters, intermediate representations and inference-related memory. The results of the experiment reveal that the memory consumption in the Real AI mode is 245.6 MB while the memory consumption in the Simulation mode is 52.3 MB which is an additional 193.3 MB of memory. Likewise, the transformer-based analysis becomes more costly, resulting in an increase in response time from 28.9 ms to 124.5 ms.

Lastly, a “negative compression ratio” mentioned in the report does not mean that the encryption algorithm has failed. The reason for the increase in output size is mostly due to the mixture of Base64 encoding and PKCS5 padding. Of course, padding will cause a few extra bytes to be used when the plaintext does not fit the DES block size, and Base64 encoding will use extra bytes for

 ISSN:2302-8149
e-ISSN:2540-9719

storing a textual representation of the binary ciphertext. So the output may be bigger than the input, especially if the input is small, in which case the overhead of the padding and encoding will be a higher percentage of the output. This size difference is not considered a weakness in the cryptography, but rather a part of the framework's performance analysis.

5 Conclusions

In order to enhance the evaluation and management of DES encryption performance and security, the proposed framework incorporates the use of metrics-driven pipeline, AI-driven analytical intelligence and entropy-driven key generation. The results show that the metrics-driven pipeline has reduced the total CPU and memory usage, growing gradually with the increase in data volume, while providing enhanced observability of the system. The AI capabilities include data-aware suggestions for encryption strategies and key strength evaluation, which deliver excellent classification performance on structured and high-entropy data. Moreover, the entropy-based key-generating mechanism generates higher entropy and has lower collision than traditional random key and manually selected keys. In summary, the proposed framework offers a comprehensive solution for assessing DES encryption from the security, efficiency, resource usage and intelligent analysis. While these advances have been made, DES still has the drawback of its 56-bit effective key space and the extra memory and inference needs of AI-based analysis should be taken into account in resource-limited settings. Optimizing AI inference and pipeline memory usage should thus be explored in future work. More importantly, the proposed framework, which is based on metrics and artificial intelligence, is not restricted to the DES, but it can be extended to modern symmetric encryption algorithms like AES by replacing the DES encryption module in its modular structure with another algorithm, while keeping the performance monitoring, entropy analysis, key evaluation, and AI-driven decision-support components intact. This extension would provide the same structure to assess contemporary cryptographic methods with regards to security, cost-effectiveness, resources, and adaptive encryption techniques.

Acknowledgements

The authors would like to express their appreciation to the Faculty of Science & Literature at the American University of Culture & Education, Beirut, Lebanon, for providing the academic environment and support necessary for completing this research. The authors also extend their gratitude to all individuals and institutions who contributed directly or indirectly by supplying relevant materials, tools, and constructive feedback that enhanced the quality of this study.

References

[1] M. Zubair, M. Ahmed, A. Ali, and S. Naeem, "*Network Security and Cryptography Challenges and Trends on Recent Technologies*," *J. Appl. Emerg. SCI.*, Vol. 13, No. 1, pp. 1–8, 2023. DOI:*10.36785/jaes.131546*

[2] K. Nahar, O. Al-Hazaimeh, M. Alshanaq, and M. Bawaneh, "*Analytical Approach for Data Encryption Standard Algorithm*," *Int. J. Interact. Mob. Technol.*, Vol. 17, May 2023. DOI:*10.3991/ijim.v17i14.38641*

[3] H. Taherdoost, T.-V. Le, and K. Slimani, "*Cryptographic Techniques in Artificial Intelligence Security: A Bibliometric Review*," *Cryptography*, Vol. 9, No. 1, 2025. *https://doi.org/10.3390/cryptography9010017*

[4] I. Ficili, M. Giacobbe, G. Tricomi, and A. Puliafito, "*From Sensors to Data Intelligence: Leveraging IoT, Cloud, and Edge Computing with AI*," *Sensors*, Vol. 25, No. 6, pp. 1–25, 2025. *https://doi.org/10.3390/s25061763*

[5] K. Chaganti and P. Paidy, "*Strengthening Cryptographic Systems with AI-Enhanced Analytical Techniques*," *Int. J. Appl. Math. Res.*, Vol. 14, pp. 13–24, Apr. 2025. DOI:*10.14419/fh79gr07*

 ISSN:2302-8149
e-ISSN:2540-9719

[6] S. A. S. Almola, H. A. Younis, and R. S. Khudeyer, "*A Robust Image Encryption Framework using Deep Feature Extraction and AES Key Optimization*," *Cryptography*, Vol. 10, No. 2, 2026. https://doi.org/10.3390/cryptography10020016

[7] O. Popoola, M. A. Rodrigues, J. Marchang, A. Shenfield, A. Ikpehai, and J. Popoola, "*An Optimized Hybrid Encryption Framework for Smart Home Healthcare: Ensuring Data Confidentiality and Security*," *Internet of Things*, Vol. 27, p. 101314, 2024. https://doi.org/10.1016/j.iot.2024.101314

[8] S. Subramani, S. Munuswamy, K. Arputharaj, and S. Kumar Svn, "*Review of Security Methods based on Classical Cryptography and Quantum Cryptography*," *Cybern. Syst.*, Vol. 56, pp. 1–19, Jan. 2023. DOI:10.1080/01969722.2023.2166261

[9] M. M. Hoobi, "*Improved Structure of Data Encryption Standard Algorithm*," *j. Southwest Jiaotong Univ.*, Vol. 55, No. 5, 2020. DOI : https://doi.org/10.35741/issn.0258-2724.55.5.12

[10] D. Deepthi, B. Benny, and K. Sreenu, "*Various Ciphers in Classical Cryptography*," *J. Phys. Conf. Ser.*, Vol. 1228, p. 12014, May 2019. DOI:10.1088/1742-6596/1228/1/012014

[11] Q. Z. Abdulla and M. D. Al-Hassani, "*Robust Password Encryption Technique with an Extra Security Layer*," *Iraqi J. SCI.*, Vol. 64, No. 3, pp. 1477–1486, 2023. https://doi.org/10.24996/ijs.2023.64.3.36

[12] H. E. S., H. S. Harba, I. A. Abdulmunem, and S. S. Hussein, "*Improving Security of the Crypto-Stego Approach using Time Sequence Dictionary and Spacing Modification Techniques*," *Iraqi J. SCI.*, Vol. 62, No. 5, May 2021. https://doi.org/10.24996/10.24996/ijs.2021.62.5.35

[13] K. Nahar and P. Chakraborty, "*Improved Approach of Rail Fence for Enhancing Security*," *Int. J. Innov. Technol. Explor. Eng.*, Vol. 9, pp. 583–585, Jul. 2020. DOI:10.35940/ijitee.I7637.079920

[14] M. Hoobi, "*Efficient Hybrid Cryptography Algorithm*," *J. Southwest Jiaotong Univ.*, Vol. 55, Jan. 2020. DOI:10.35741/issn.0258-2724.55.3.5

[15] M. M. Hoobi, "*Minar International Journal of Applied Sciences and Technology Survey : Efficient Hybrid Algorithms Of Cryptography*," pp. 1–16, 2020. DOI:10.35741/issn.0258-2724.55.3.5