Statistical Security Analysis NIST and Security Enhancement of CBT Cipher for Secure Data and Image Transmission

Walid W. Souror[1], Mohamed Fouad[2], Ali E. Takieldeen[3]
Electronics and Communication Department, Faculty of Engineering, Zagazig University, Egypt[1,2]
Artificial Intelligence Department, Faculty of Artificial Intelligence, Delta University for Science and Technology[3]

***Abstract*—Strong cybersecurity measures are essential in our digital world. This study presents a new cryptographic algorithm called CBT "Combined Blowfish Trilogy" which greatly improves the security of the traditional Blowfish algorithm without compromising computational efficiency. The encryption algorithm proposed herein presents a hybrid system which combines carefully designed countermeasures against side-channel attacks with three consecutive Blowfish Feistel network stages and surpasses the performance of DES, 3DES, AES, and Blowfish itself. The proposed algorithm is assessed using the Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications (NIST STS) consisting of the Mono-Bit, Run, Serial, Block Frequency, Spectral, Cumulative Sums, Entropy, and Avalanche tests. These results show that Combined Blowfish Trilogy is a good technique to prevent data security from side-channel attack breaches and it shows significant resistance against them as evidenced by the performance improvements observed over standard Blowfish (an average of 133.8% for text and 117.5% in image datasets according to NIST tests).**

***Keywords*—Cyber security; cryptography; blowfish; countermeasure SCA attacks; combined blowfish trilogy**

I. Introduction

The cryptographic paradigm of today's world is that the statistical relationship between the plaintext, key material, and ciphertext should be concealed. It was Shannon's theory of secrecy systems which introduced the idea of two necessary properties for a secure cipher: confusion and diffusion. These properties have continued to shape the design and testing of symmetric encryption algorithms [1]. The efficient encryption of text, images and other multimedia objects is especially critical in practical digital systems as these objects are shared among cloud services, distributed applications and embedded devices.

Blowfish is a well-known symmetric block cipher which was at first proposed as a 64-bit Feistel network with variable key length up to 448 bits [2]. Its small size and effective software performance have ensured that it's been a great baseline for tweaked symmetric encryption schemes. But the present evaluation of security should not be restricted to speed or just apparent randomness of the ciphertext. The algorithmic strength, the maturity of implementations, key management, and mode of operation all contribute to the practical security of a cipher, as is evidenced by standardized block ciphers like AES [3], legacy DES [4] and TDEA/3DES [5].

Further, NIST-defined block-cipher modes demonstrate that the behavior of encryption is not solely dependent on the primitive being used, but also on how the primitive is being used in the data streams [6].

Randomness testing of statistical properties of the ciphertexts is useful for evaluating the ciphertexts, but it is NOT a replacement for cryptanalysis. The NIST Statistical Test Suite is a suite of tests to evaluate the randomness and pseudo randomness of sequences for cryptographic applications [7]. The nature and limitations of NIST SP 800-22 have been discussed in studies; they make it clear that p-values are not a measure or direct representation of cryptographic strength, but are a statistic particular to the test being used [8], [9]. Hence, this study considers NIST-style p-values as proof of randomness of the given ciphertext with respect to the tests employed, and supports this with the avalanche and performance tests.

Another way of examining diffusion is by avalanche analysis. The strict avalanche criterion was introduced to express the desirable behavior that a small change of input causes approximately half of the output bits to change [10]. Meanwhile, in case of resistance against differential cryptanalysis and linear cryptanalysis, it is required to adopt different interpretations with regards to difference propagation and statistical approximation [11], [12].

Implementation attacks introduce another level of attack as the timing leakage, power consumption, electromagnetic leakage, and profiling attacks potentially leak information even if the mathematical algorithm is sound [13], [14], [15], [16]. Masking and probing-secure circuit design offer a valuable set of buildouts for leakage reduction, but need to be carefully performed and verified [17], [18].

The Combined Blowfish Trilogy (CBT) framework examined in this study is an extension of the original Blowfish design that incorporates the following three different Blowfish-based encryption stages: the masking of plaintext, the derivation of stage keys, and the use of key whitening concepts along with session-dependent internal randomization. Key derivation relies on known recommendations for key-derivation of independent subkeys [19], [20] and the generation of random masks and session-parameters follows recommendations for deterministic random bit generators and entropy sources [21], [22]. The framework is thus checked as a cascaded and diversified Blowfish-based construction and not as an alternative to the standardized ciphers. This study has three contributions.

First, it gives a full design of text and image byte stream encryption and decryption using CBT.

Second, it gives a reproducible experimental evaluation consisting of NIST-style randomness interpretation, Hamming-distance-based avalanche analysis, computational benchmarking, and timing-variability checking.

Third, it offers the Python implementation, raw output files, and Excel results workbook as supplementary files as (https://doi.org/10.5281/zenodo.21079574), allowing the numerical results to be traced and independently reproduced.

II. Related Work and Technical Background

A. Feistel, Cascade, and Whitening-Based Constructions

Blowfish is a symmetric block cipher that is a member of the Feistel-type family. To understand the importance of repeated round transformations and key-dependent mixing in the design of block-cipher, it is helpful to have a background understanding of theoretical work on how to build pseudorandom permutations from pseudorandom functions [23]. Cascade constructions are based on this intuition of doing multiple encryption transformations in sequence.

The advantage of this cascade is based on the independence of the keys, the lack of structural weaknesses in the composition and proper treatment of intermediate values.

Key whitening: Additional key-dependent transformations around or within a block-cipher process. The Even-Mansour scheme and other minimal constructions show the effect of external key mixing on the security interpretation of a block-cipher-like construction [24].

Whitening is considered as another kind of diversification and is combined with stage-specific keys instead of being regarded as a stand-alone proof of security in CBT framework.

B. Image Encryption and Randomness Metrics

When encrypting images, it is important to consider statistical properties that may be visually or numerically noticed in the encrypted images. The primary requirements of basic cryptographic image-oriented cryptosystems are key or initial value sensitivity, resistance to leakage of statistical information and absence of visible residual structure [25]. Based on the classical works on image encryption using two-dimensional chaotic maps and subsequent three-dimensional chaotic systems, diffusion, pixel decorrelation and ciphertext unpredictability are important [26], [27].

The metrics like entropy, correlation, NPCR and UACI are widely mentioned in image-encryption studies to assess the degree of randomness and sensitivity at the pixel level [28]. To improve the statistical behavior, the following image-encryption schemes have been proposed recently: block scrambling [29], chaotic maps [30] and dynamic substitution [31] as well as S-box modification.

The following studies encourage a more comprehensive evaluation approach for CBT: the proposed framework is assessed not only in terms of NIST-style p-values, but also in terms of avalanche sensitivity and computational performance.

C. Lightweight and Modified Block-Cipher Designs

Low resource/low-cost cryptography has become significant for constrained devices and resource-sensitive applications. The standardization of Ascon for lightweight cryptography and previous lightweight block ciphers like PRESENT and SIMON/SPECK are examples of the need to take both computational cost and attack resistance into account during the security evaluation process [32], [33], [34]. A multi-stage framework like CBT should therefore report overheads and throughputs and not assume that an extra stage provides an added level of security without extra cost. Designs that have been modified to use Blowfish are still being seen in data, image and video encryption applications. Recently, researchers have investigated video encryption using a modified Blowfish, as well as a hybrid parallel image encryption for computational-complexity-aware video encryption [35], [36]. These studies demonstrate the applicability of Blowfish as a basis for research in applied encryption, but also indicate that any alternative Blowfish construction should be carefully studied in terms of randomness, sensitivity, performance and implementation assumptions.

III. Proposed Combined Blowfish Trilogy Framework

The suggested CBT approach consists of three stages in sequence implemented using Blowfish.

The plaintext is masked prior to the first of the three stages and session-specific keys are used, which are derived from a master key and session identifier. The framework also generates dynamically S-boxes and P-arrays to make the internal state in one encryption session different from another. The architecture overview is shown in Fig. 1.

Notation and Parameters (Table I)

TABLE I. Notations in CBT Framework

Symbol	Definition
P	Plaintext block or byte stream before encryption
MK	Master key supplied to the encryption session
SID	Unique session identifier or nonce
K_1, K_2, K_3	Stage-specific keys derived from MK and SID
M	Random mask used for plaintext masking
S_{dyn}, P_{dyn}	Session-dependent S-boxes and P-array
C_1, C_2, C_3	Intermediate and final ciphertext outputs
C	Final ciphertext

A. Encryption Process

The implementation generates or receives MK and SID, then computes K_1, K_2, and K_3, generates a random mask M, generates S_{dyn} and P_{dyn}, and masks the plaintext before using the three stages, all of which are based on Blowfish. Also, when encrypting text and image byte streams, the same block handling and padding policy is used to ensure consistency with the comparisons.

The masked plaintext is computed as follows:

$$P_m = P \oplus M \tag{1}$$

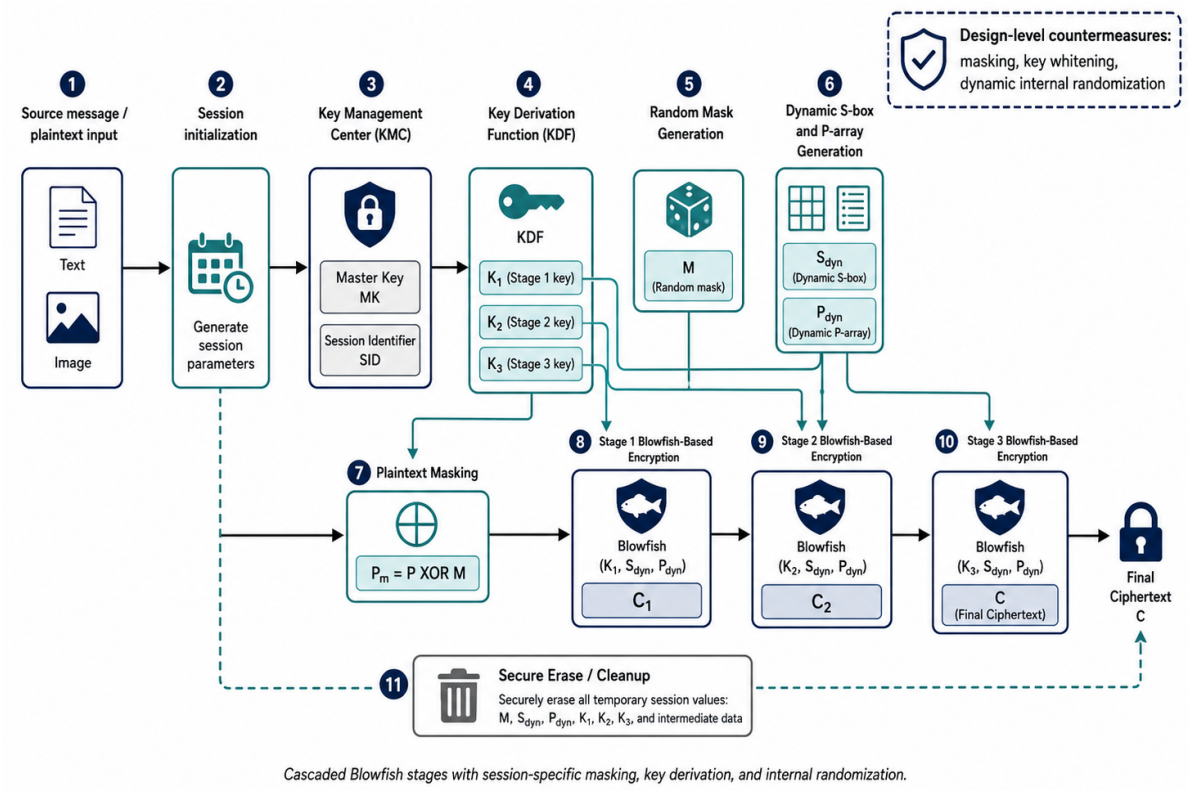


Fig. 1. CBT is a combined Blowfish Trilogy framework and its overall architecture. The framework involves session initiation, key generation, masking, internal dynamism and three cascaded Blowfish encryption stages to generate the final ciphertext.

The three-stage encryption can then be summarized as:

$$C_1 = \text{BlowfishEncrypt}(P_m,\ K_1,\ S_{dyn},\ P_{dyn}) \quad (2)$$

$$C_2 = \text{BlowfishEncrypt}(C_1,\ K_2,\ S_{dyn},\ P_{dyn}) \quad (3)$$

$$C_3 = \text{BlowfishEncrypt}(C_2,\ K_3,\ S_{dyn},\ P_{dyn}) \quad (4)$$

$$C = C_3$$

Fig. 2 shows the encryption flow used in the framework.

B. Decryption Process

Decryption reverses the order of the three encryption stages. The same session-specific parameters are used to recover the masked plaintext and then remove the mask (Fig. 3).

$$C_2 = \text{BlowfishDecrypt}(C,\ K_3,\ S_{dyn},\ P_{dyn}) \quad (5)$$

$$C_1 = \text{BlowfishDecrypt}(C_2,\ K_2,\ S_{dyn},\ P_{dyn}) \quad (6)$$

$$P_m = \text{BlowfishDecrypt}(C_1,\ K_1,\ S_{dyn},\ P_{dyn}) \quad (7)$$

$$P = P_m \oplus M$$

C. Key Management and Dynamic Internal Randomization

The framework does not contain the direct re-use of a single stage key, as K_1, K_2, and K_3 are derived from MK and SID. Each encryption session has its own session identifier, and the decryption operation has to build up or securely access the same elements of the session. In the reference implementation, intermediate values, temporary keys, masks, and dynamically created internal structures are wiped out after use. Dynamic S-box and P-array randomization to diversify internal state associated with current session. These structures have a security impact that is dependent on the quality of the random source, and the maintenance of the necessary properties of the cipher. Thus, they are introduced as mechanisms for internal diversification, which serve as supplementation, not substitution, for formal cryptanalytic evaluation (Fig. 4 and 5).

D. Security Interpretation of the Design

The CBT framework applies more transformations to the plaintext and adds session diversification. But it is not only by visual ciphertext appearance nor by NIST p-values that the design is evaluated. Randomness, avalanche sensitivity,

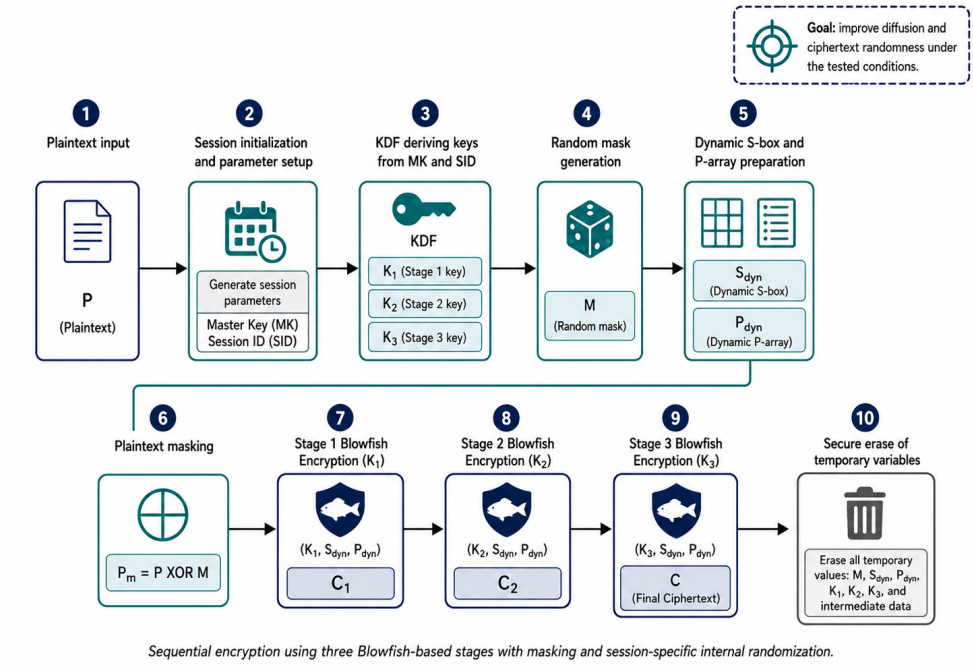


Fig. 2. Encryption flow of the proposed CBT framework. The plaintext is masked using a session-specific random mask and processed through three sequential Blowfish-based encryption stages using stage-specific keys and dynamic internal structures.

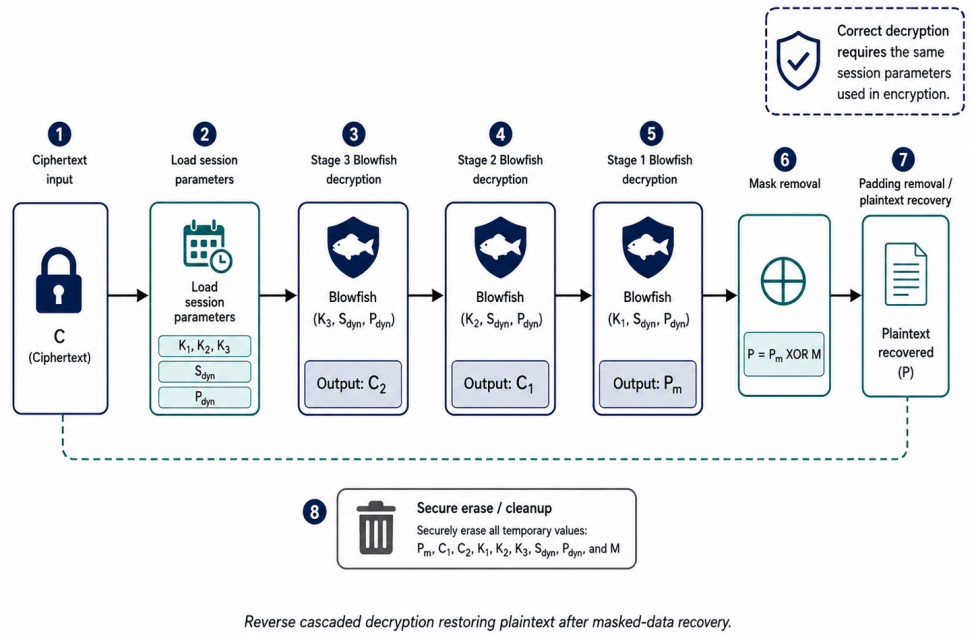


Fig. 3. Decryption flow of the proposed CBT framework. The corresponding session parameters are used to decrypt the ciphertext in the opposite order, removing the mask and recovering the plaintext.

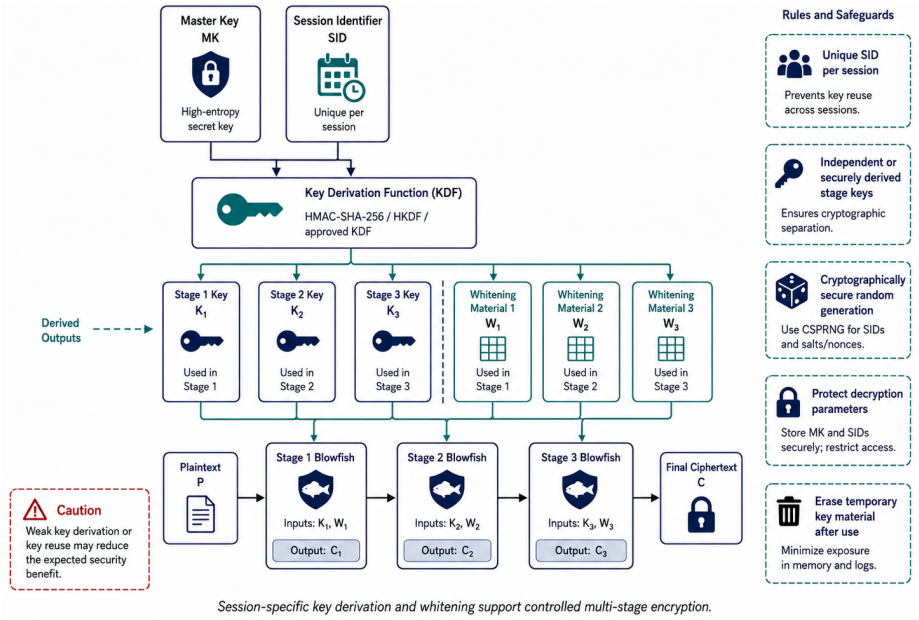


Fig. 4. Key management and whitening flow in the CBT framework. Stage-specific keys and whitening materials are either generated independently or securely derived from a master key and unique session identifier.

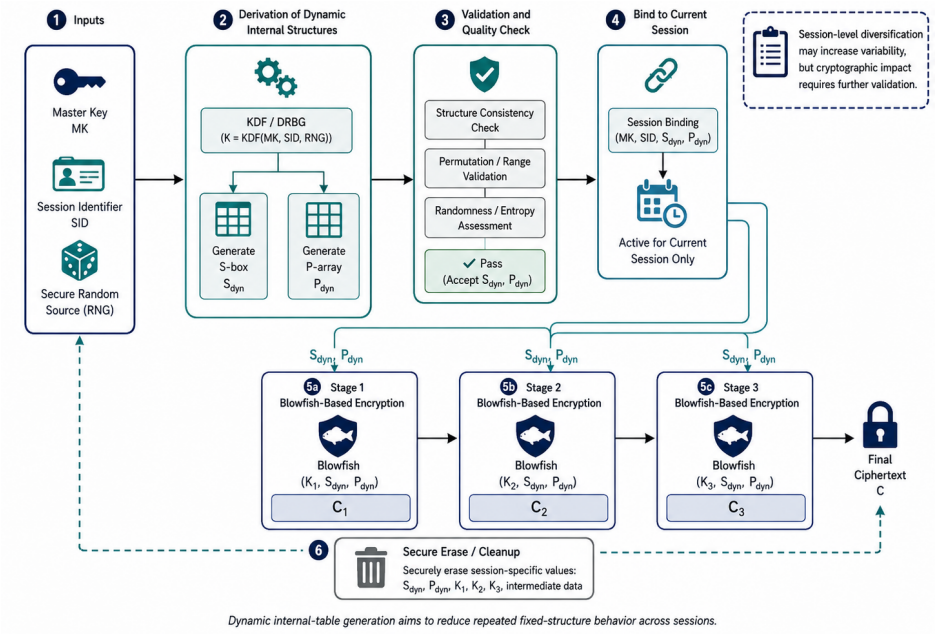


Fig. 5. Dynamic S-box and P-array randomization in the CBT framework. Session-specific internal structures are created, verified, attached to the session and discarded afterwards, to provide internal diversification.

timing variability and performance overhead are reported independently. The resistance to differential cryptanalysis, linear cryptanalysis, chosen-plaintext attack, related-key attack, fault injection, timing leakage, power leakage and electromagnetic attack is not evaluated in the present software prototype and would need dedicated attack evaluation.

IV. Experimental Methodology

The experimental methodology consists of four different parts: NIST-style statistical randomness interpretation, Hamming-distance-based avalanche analysis, computational performance comparison and timing-variability testing.

Pass/fail results have been included in the re-arranged p-values from the original experiment at alpha = 0.01. Avalanche, performance and timing are based on the Python reference implementation, which is included as supplementary material (Fig. 6).

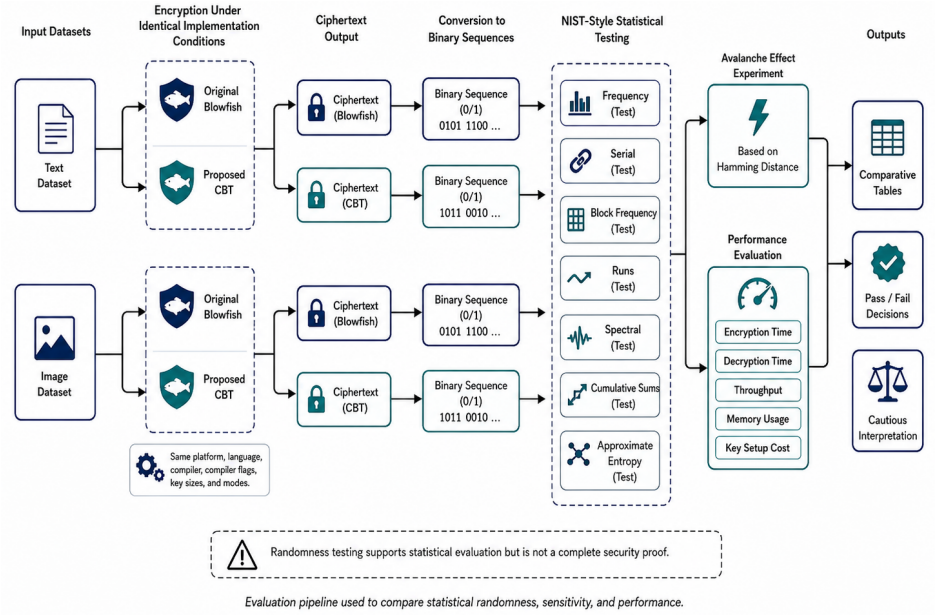


Fig. 6. Experimental evaluation pipeline. Both text and image datasets are encrypted under the same implementation conditions, turned into the binary sequence, and tested statistically, using NIST-style statistical tests, avalanche analysis and performance measurement.

Experimental Setup (Table II)

TABLE II. EXPERIMENTAL SETUP USED FOR THE REVISED EVALUATION

Parameter	Value
Implementation language	Python reference implementation
Compared algorithms	Original Blowfish and proposed CBT framework
Text workload	65,536-byte reproducible byte stream
Image workload	49,152-byte reproducible image-like byte stream
NIST significance level	alpha = 0.01
Avalanche trials	1,000 trials per algorithm, dataset, and sensitivity type
Performance repetitions	50 repetitions per algorithm and dataset
Timing repetitions	200 repetitions per algorithm and input type
Supplementary result workbook	Supplementary File S1
Python code package	Supplementary Files S2 and S3

A. NIST-Style Randomness Interpretation

The NIST type results are analyzed according to the fixed threshold alpha = 0.01. A test result is considered to be pass if $p \geq 0.01$ or fail if $p < 0.01$. The p-value is just a statistical measure of the corresponding randomness test and should not be interpreted as a measure of encryption strength.

$$\text{Decision} = \begin{cases} \text{Pass} & \text{if } p \geq 0.01 \\ \text{Fail} & \text{otherwise} \end{cases}$$

B. Avalanche Effect Analysis

The avalanche experiment evaluates sensitivity to one-bit changes in the plaintext and in the key. For plaintext-bit sensitivity, a plaintext block P and a modified block P' differing in exactly one bit are encrypted under the same key and configuration. For key-bit sensitivity, the same plaintext is encrypted using keys K and K' differing in exactly one bit. The resulting ciphertexts are compared using Hamming distance.

$$\text{Avalanche}(\%) = \frac{HD(C, C')}{N} \times 100 \tag{8}$$

where, $HD(C, C')$ is the Hamming distance between the two ciphertext outputs and N is the ciphertext length in bits.

The reported values summarize the mean, standard deviation, minimum, and maximum over 1,000 trials for each condition.

C. Computational Performance and Timing Variability

Performance is measured using encryption time, decryption time, throughput, memory usage, key setup time, mask-generation cost, and overhead relative to Original Blowfish. The measurements are based on the same reproducible workloads used for the avalanche experiment.

$$\text{Throughput} = \frac{\text{Data size}}{\text{Execution time}} \tag{9}$$

$$\text{Overhead}(\%) = \frac{\text{CBT time} - \text{Blowfish time}}{\text{Blowfish time}} \times 100 \tag{10}$$

The timing-variability check is a comparison of the repeated encryption of fixed and random plaintexts. This check is documented as a software-level timing observation and is not intended as a full-fledged practical side-channel assessment.

V. Results and Discussion

A. NIST-Style Test Outcomes

Table III: 7 of 9 reported results for text and image data were good with the original Blowfish. All 9 outcomes reported in both datasets passed the CBT. The improvement is primarily for the Serial Test parts, where the original Blowfish outputs failed and CBT passed (Table IV). These results are indicative of good ciphertext randomness properties for the chosen tests, but do not constitute full cryptographic security (Fig. 7).

B. Avalanche Effect Results

The mean avalanche value is very close to 50%, which is the expected diffusion property of the output of a block cipher to a one-bit change in the input or key. The CBT results are still near to the original Blowfish baseline and include the masking, cascading, and session-level diversification (Table V). Thus, the avalanche results should be taken more as an indication of the diffusion sensitivity than as a standalone proof of resistance to differential cryptanalysis (Fig. 8).

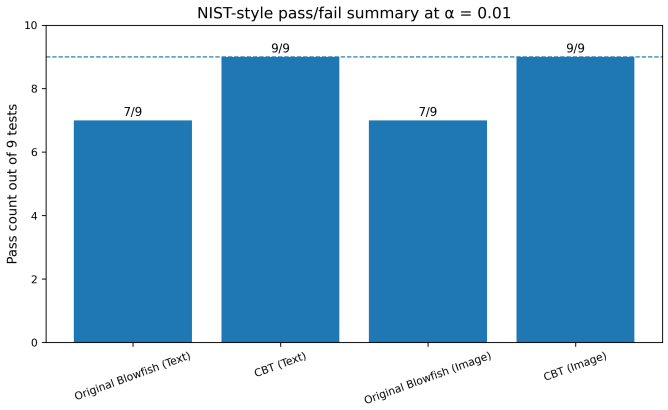


Fig. 7. NIST-style pass/fail summary at alpha = 0.01. The proposed CBT framework achieved 9/9 outcomes for both text and image data, compared to 7/9 outcomes for both text and image data for Original Blowfish.

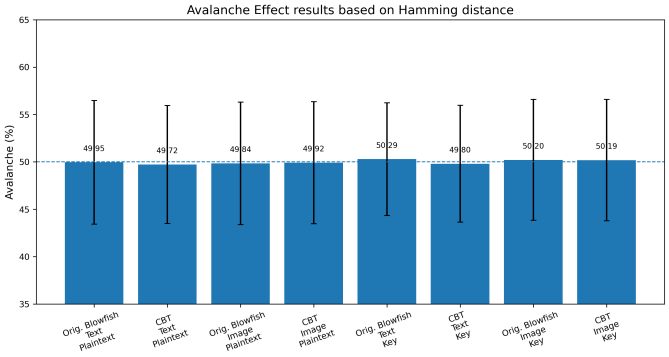


Fig. 8. Avalanche-effect comparison between Original Blowfish and CBT. This involved plaintext-bit and key-bit sensitivity experiments with Hamming-distance values being computed over 1,000 trials per condition.

C. Performance Benchmarking

The CBT prototype is heavier on calculation as it involves performing three consecutive stages of Blowfish and also involves mask generation and session-dependent processing. The measured overhead is that of the Python reference implementation and should not be used to extrapolate to optimized native-code, hardware-assisted or parallel implementations (Table VI). The goal of the benchmark is to transparently measure the prototype cost and not to state anything about the superiority of the speed (Fig. 9).

D. Timing-Variability Check

The timing-variability check gives a partial software-level view of the fixed and random plaintext runtimes (Table VII). These measurements are not a substitute for hardware leakage testing. A complete side-channel evaluation would need a defined leakage model, power or electromagnetic traces, a measurement platform and statistical leakage analysis.

E. Consolidated Interpretation

When the results of the two are combined, it is seen that CBT enhances the NIST-style pass-rate behavior without sacrificing the avalanche values, which are in or close to the desirable 50% diffusion region. This increase in structural complexity is borne out by the performance benchmark, which shows that there is a computational cost in the reference implementation.

So, the new assessment provides a balanced view of the framework: while CBT can enhance selected statistical-randomness metrics and a measure of diversification, p-values alone are not complete evidence of cryptographic security.

VI. Supplementary Materials and Reproducibility

The numerical results presented in this study are supplemented by file(s). A results workbook in Excel format is included as Supplementary File S1 and contains NIST pass/fail tables, raw avalanche trials, avalanche summaries, raw performance measurements, performance summaries, timing-variability data and experimental setup details. The Python experimental package, which was used to produce the avalanche, performance, timing and CSV outputs, is provided in Supplementary File S2. All of the Python scripts have been made available in a Word-readable format in Supplementary File S3 for convenient inspection on devices lacking source-code viewers. The raw CSV outputs from the experimental scripts are provided in Supplementary File S4 (Table VIII).

VII. Conclusion

In this study, a modified and well-documented assessment of Combined Blowfish Trilogy framework for text and image encryption was proposed. The framework involves three stages of Blowfish-based encryption along with key derivation, key whitening ideas, masking of plaintext and session-specific internal randomization. The updated assessment categorizes the different aspects of the experiment into four parts: randomness of the ciphertext, avalanche sensitivity, computational performance, and timing variability.

The NIST-style analysis reveals that the CBT passed all 9 test results for both the text and image datasets at alpha = 0.01, while Original Blowfish passed 7 of 9 for both datasets. The

TABLE III. NIST-STYLE P-VALUES AND PASS/FAIL INTERPRETATION AT ALPHA = 0.01

Test	BF Text	Decision	CBT Text	Decision	BF Image	Decision	CBT Image	Decision
Frequency Test	0.187	Pass	0.233366	Pass	0.596	Pass	0.800278	Pass
Serial Test p1	3.94×10^{-24}	Fail	0.538113	Pass	8.68×10^{-54}	Fail	0.718264	Pass
Serial Test p2	2.63×10^{-12}	Fail	0.250261	Pass	6.55×10^{-27}	Fail	0.726591	Pass
Block Frequency Test	0.0467	Pass	0.834676	Pass	0.228	Pass	0.272722	Pass
Runs Test	0.0557	Pass	0.204289	Pass	0.141	Pass	0.865479	Pass
Spectral Test	0.872	Pass	0.935652	Pass	0.121	Pass	0.037693	Pass
Cumulative Sums Forward	0.224	Pass	0.309039	Pass	0.964	Pass	0.990496	Pass
Cumulative Sums Backward	0.243	Pass	0.365787	Pass	0.525	Pass	0.853891	Pass
Approximate Entropy Test	0.143	Pass	0.469485	Pass	0.0696	Pass	0.490191	Pass

TABLE IV. SUMMARY OF NIST-STYLE PASS RATES

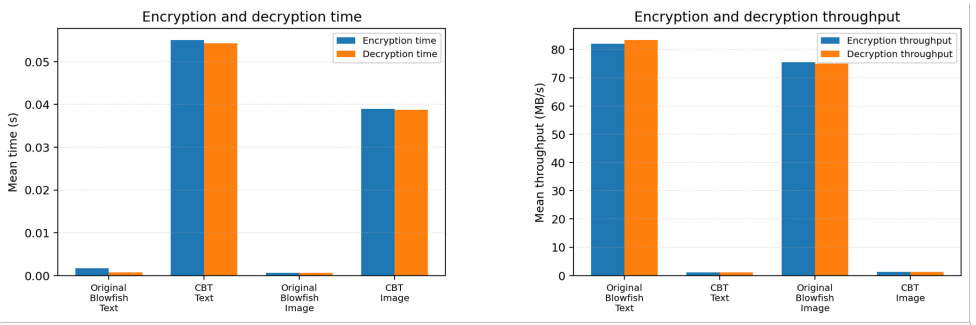
Configuration	Tests	Pass	Pass Rate (%)	Mean p
Original Blowfish (Text)	9	7	77.78	0.196744
CBT (Text)	9	9	100.00	0.460074
Original Blowfish (Image)	9	7	77.78	0.293978
CBT (Image)	9	9	100.00	0.639512

TABLE V. HAMMING-DISTANCE-BASED AVALANCHE EFFECT RESULTS

Algorithm	Dataset	Sensitivity	Trials	Mean (%)	Std. Dev.	Min (%)	Max (%)
Original Blowfish	Image	Key-bit	1000	50.205	6.378	32.812	73.438
Original Blowfish	Image	Plaintext-bit	1000	49.836	6.459	29.688	68.750
CBT	Image	Key-bit	1000	50.188	6.393	32.812	67.188
CBT	Image	Plaintext-bit	1000	49.917	6.435	28.125	71.875
Original Blowfish	Text	Key-bit	1000	50.289	5.945	32.812	68.750
Original Blowfish	Text	Plaintext-bit	1000	49.953	6.524	28.125	70.312
CBT	Text	Key-bit	1000	49.797	6.162	31.250	71.875
CBT	Text	Plaintext-bit	1000	49.722	6.227	31.250	70.312

TABLE VI. PERFORMANCE BENCHMARK RESULTS FOR THE PYTHON REFERENCE IMPLEMENTATION

Algorithm	Dataset	Size (B)	Enc. (s)	Dec. (s)	Enc. TP (MB/s)	Dec. TP (MB/s)	Mem. (MB)	Overhead (%)
Original Blowfish	Image	49152	0.000635	0.000645	75.548	74.940	0.094	0.00
CBT	Image	49152	0.038926	0.038800	1.207	1.210	0.430	6031.24
Original Blowfish	Text	65536	0.001747	0.000754	82.021	83.308	0.138	0.00
CBT	Text	65536	0.055050	0.054325	1.140	1.154	0.568	3051.75



Configuration	Enc. time (s)	Dec. time (s)	Enc. throughput (MB/s)	Dec. throughput (MB/s)	Overhead vs. Blowfish (%)
Original Blowfish (Text)	0.001747	0.000754	82.02	83.31	Baseline
CBT (Text)	0.055050	0.054325	1.14	1.15	3051.8
Original Blowfish (Image)	0.000635	0.000645	75.55	74.94	Baseline
CBT (Image)	0.038926	0.038800	1.21	1.21	6031.2

Performance comparison based on repeated Python benchmark runs under identical experimental conditions.

Fig. 9. Performance evaluation results for the Python reference implementation. The figure shows the summary of encryption/decryption time and throughput for Original Blowfish and CBT for the text workloads and image workloads.

TABLE VII. SOFTWARE TIMING-VARIABILITY CHECK

Algorithm	Input type	Reps	Mean (s)	Std. dev. (s)	Min (s)	Max (s)
CBT	Fixed plaintext	200	0.00245414	0.00061309	0.00201252	0.00561595
CBT	Random plaintext	200	0.00224973	0.00025428	0.00197839	0.00504541
Original Blowfish	Fixed plaintext	200	0.00021473	0.00149891	0.00009377	0.02130122
Original Blowfish	Random plaintext	200	0.00011441	0.00002633	0.00009251	0.00032433

TABLE VIII. SUPPLEMENTARY FILES SUPPLIED WITH THE MANUSCRIPT

File	Contents
S1	`CBT_Supplementary_Results_Workbook.xlsx`: Excel workbook containing all final result sheets. https://doi.org/10.5281/zenodo.21079051
S2	`CBT_Python_Code_Package.zip`: executable Python package and requirements file. https://doi.org/10.5281/zenodo.21079574
S3	`CBT_Python_Codes_Readable.docx`: Word-readable copy of all Python scripts. https://doi.org/10.5281/zenodo.21079681
S4	`CBT_Raw_CSV_Outputs.zip`: raw CSV files exported from the experimental run. https://doi.org/10.5281/zenodo.21079770

Hamming-distance-based avalanche analysis gave the mean values close to 50%, which showed suitable sensitivity for one-bit change in plaintext and key. The cost of the prototype with the three stages was quantified and it was demonstrated that the extra transformations incur measurable overhead in the Python implementation.

Results indicate that, under the tested conditions, the CBT is a diversified framework based on Blowfish with better selected statistical-randomness behavior. Meanwhile, the study does not claim that the NIST p-values are full security proofs, and clarifies what constitutes software-level timing checks from actual side-channel resistance. Future implementation studies can optimize the prototype in native code and/or hardware and do dedicate leakage measurement, but the present manuscript offers a complete reproducible baseline for statistical, avalanche, and performance evaluation.

REFERENCES

[1] C. E. Shannon, "Communication theory of secrecy systems," *Bell System Technical Journal*, vol. 28, no. 4, pp. 656–715, 1949. https://doi.org/10.1002/j.1538-7305.1949.tb00928.x

[2] B. Schneier, "Description of a new variable-length key, 64-bit block cipher (Blowfish)," in *Fast Software Encryption*, LNCS 809, pp. 191–204, Springer, 1994. https://doi.org/10.1007/3-540-58108-1_24

[3] National Institute of Standards and Technology, "Advanced Encryption Standard (AES)," FIPS PUB 197, 2001; updated 2023. https://doi.org/10.6028/NIST.FIPS.197-upd1

[4] National Institute of Standards and Technology, "Data Encryption Standard (DES)," FIPS PUB 46-3, 1999, withdrawn 2005. https://csrc.nist.gov/pubs/fips/46-3/final

[5] E. Barker and N. Mouha, "Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher," NIST SP 800-67 Rev. 2, 2017. https://doi.org/10.6028/NIST.SP.800-67r2

[6] M. Dworkin, "Recommendation for Block Cipher Modes of Operation: Methods and Techniques," NIST SP 800-38A, 2001. https://doi.org/10.6028/NIST.SP.800-38A

[7] A. Rukhin, J. Soto, J. Nechvatal *et al.*, "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications," NIST SP 800-22 Rev. 1a, 2010. https://doi.org/10.6028/NIST.SP.800-22r1a

[8] A. Iwasaki, "Analysis of NIST SP800-22 focusing on randomness of each sequence," arXiv:1710.01441, 2017. https://arxiv.org/abs/1710.01441

[9] H. Haramoto *et al.*, "Study on upper limit of sample sizes for a two-level test in NIST SP800-22," arXiv:1912.10602, 2019. https://arxiv.org/abs/1912.10602

[10] A. F. Webster and S. E. Tavares, "On the design of S-boxes," in *Advances in Cryptology — CRYPTO '85*, LNCS 218, pp. 523–534, Springer, 1985. https://doi.org/10.1007/3-540-39799-X_41

[11] E. Biham and A. Shamir, "Differential cryptanalysis of DES-like cryptosystems," *Journal of Cryptology*, vol. 4, no. 1, pp. 3–72, 1991. https://doi.org/10.1007/BF00630563

[12] M. Matsui, "Linear cryptanalysis method for DES cipher," in *Advances in Cryptology — EUROCRYPT '93*, LNCS 765, pp. 386–397, Springer, 1994. https://doi.org/10.1007/3-540-48285-7_33

[13] P. C. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," in *Advances in Cryptology — CRYPTO '96*, LNCS 1109, pp. 104–113, Springer, 1996. https://doi.org/10.1007/3-540-68697-5_9

[14] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology — CRYPTO '99*, LNCS 1666, pp. 388–397, Springer, 1999. https://doi.org/10.1007/3-540-48405-1_25

[15] S. Chari, J. R. Rao, and P. Rohatgi, "Template attacks," in *Cryptographic Hardware and Embedded Systems — CHES 2002*, LNCS 2523, pp. 13–28, Springer, 2003. https://doi.org/10.1007/3-540-36400-5_3

[16] S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Springer, 2007. https://doi.org/10.1007/978-0-387-38162-6

[17] M. Rivain and E. Prouff, "Provably secure higher-order masking of AES," in *Cryptographic Hardware and Embedded Systems — CHES 2010*, LNCS 6225, pp. 413–427, Springer, 2010. https://doi.org/10.1007/978-3-642-15031-9_28

[18] Y. Ishai, A. Sahai, and D. Wagner, "Private circuits: Securing hardware against probing attacks," in *Advances in Cryptology — CRYPTO 2003*, LNCS 2729, pp. 463–481, Springer, 2003. https://doi.org/10.1007/978-3-540-45146-4_27

[19] L. Chen, "Recommendation for Key Derivation Using Pseudorandom Functions," NIST SP 800-108 Rev. 1, 2022. https://doi.org/10.6028/NIST.SP.800-108r1

[20] H. Krawczyk and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)," RFC 5869, IETF, 2010. https://www.rfc-editor.org/rfc/rfc5869

[21] E. Barker and J. Kelsey, "Recommendation for Random Number Generation Using Deterministic Random Bit Generators," NIST SP 800-90A Rev. 1, 2015. https://doi.org/10.6028/NIST.SP.800-90Ar1

[22] M. S. Turan, E. Barker, J. Kelsey *et al.*, "Recommendation for the Entropy Sources Used for Random Bit Generation," NIST SP 800-90B, 2018. https://doi.org/10.6028/NIST.SP.800-90B

[23] M. Luby and C. Rackoff, "How to construct pseudorandom permutations from pseudorandom functions," *SIAM Journal on Computing*, vol. 17, no. 2, pp. 373–386, 1988. https://doi.org/10.1137/0217022

[24] O. Dunkelman, N. Keller, and A. Shamir, "Minimalism in cryptography: The Even-Mansour scheme revisited," in *Advances in Cryptology — EUROCRYPT 2012*, LNCS 7237, pp. 336–354, Springer, 2012. https://doi.org/10.1007/978-3-642-29011-4_21

[25] G. Álvarez and S. Li, "Some basic cryptographic requirements for chaos-based cryptosystems," *International Journal of Bifurcation and Chaos*, vol. 16, no. 8, pp. 2129–2151, 2006. https://doi.org/10.1142/S0218127406015970

[26] J. Fridrich, "Symmetric ciphers based on two-dimensional chaotic maps," *International Journal of Bifurcation and Chaos*, vol. 8, no. 6, pp. 1259–1284, 1998. https://doi.org/10.1142/S021812749800098X

[27] G. Chen, Y. Mao, and C. K. Chui, "A symmetric image encryption scheme based on 3D chaotic cat maps," *Chaos, Solitons & Fractals*, vol. 21, no. 3, pp. 749–761, 2004. https://doi.org/10.1016/j.chaos.2003.12.022

[28] Y. Wu, J. P. Noonan, and S. Agaian, "NPCR and UACI randomness tests for image encryption," *Journal of Selected Areas in Telecommunications*, vol. 1, no. 2, pp. 31–38, 2011.

[29] P. Ramasamy, V. Ranganathan, S. Kadry, R. Damaševičius, and T. Blažauskas, "An image encryption scheme based on block scrambling, modified zigzag transformation and key generation using enhanced logistic-tent map," *Entropy*, vol. 21, no. 7, p. 656, 2019. https://doi.org/10.3390/e21070656

[30] M. Ahmad and A. Chopra, "Chaotic dynamic S-boxes based substitution approach for digital images," arXiv:1709.07620, 2017. https://arxiv.org/abs/1709.07620

[31] M. Gholamzadeh and B. Khadem, "A hybrid image encryption scheme based on chaos and a DPA-resistant S-box," arXiv:2312.15280, 2023. https://arxiv.org/abs/2312.15280

[32] M. S. Turan *et al.*, "Ascon-Based Lightweight Cryptography Standards for Constrained Devices," NIST SP 800-232, 2025. https://doi.org/10.6028/NIST.SP.800-232

[33] A. Bogdanov, L. R. Knudsen, G. Leander *et al.*, "PRESENT: An ultra-lightweight block cipher," in *Cryptographic Hardware and Embedded Systems — CHES 2007*, LNCS 4727, pp. 450–466, Springer, 2007. https://doi.org/10.1007/978-3-540-74735-2_31

[34] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers, "The SIMON and SPECK families of lightweight block ciphers," IACR ePrint 2013/404, 2013. https://eprint.iacr.org/2013/404

[35] A. E. Adeniyi *et al.*, "Computational complexity of modified Blowfish cryptographic algorithm for video encryption," *Algorithms*, vol. 15, no. 10, p. 373, 2022. https://doi.org/10.3390/a15100373

[36] A. A. A. Abdulkadhim and A. Lakizadeh, "Hybrid parallelism image encryption algorithm based on a modified Blowfish algorithm," *International Journal of Intelligent Engineering and Systems*, vol. 17, no. 6, pp. 934–946, 2024. https://doi.org/10.22266/ijies2024.1231.70