Available online at www.qu.edu.iq/journalcm
JOURNAL OF AL-QADISIYAH FOR COMPUTER SCIENCE AND MATHEMATICS
ISSN:2521-3504(online) ISSN:2074-0204(print)

An Enhanced Image Encryption Framework Integrating Hyper-Chaotic Maps with the Serpent Block Cipher for Secure Medical and Biometric Data Transmission

Abdulhadi Nadhim Mohsin Alzamili *

Computer Science Department, College of Education for Pure Sciences, University of Wasit, Wasit, Iraq
Email: abd_mohsen@uowasit.edu.iq

ARTICLE INFO

Article history:
Received: 05 /03/2026
Rrevised form: 13 /04/2026
Accepted : 15 /04/2026
Available online: 30 /06/2026

Keywords:

Image Encryption, Serpent Algorithm, HyperChaotic Map, Block Cipher, CBC Mode, Confusion–Diffusion, Medical Image Security, Biometric Data Protection.

ABSTRACT

The secure transmission of sensitive e-images, especially medical records and biometric templates, remains a vital challenge because of the characteristics of images, including high adjacency correlation between the pixels, large volumes of data, and vulnerability to statistical attacks. This study presents an enhanced multi stage confusion–diffusion image encryption framework employing a four-dimensional (4D) hyper-chaotic map with the Serpent block cipher (via Cipher Block Chaining (CBC) mode). The confusion stage employs a 4D hyperchaotic system that permutes pixels through a variation at the pixel-level to destabilize the spatial temporal structure of the original image, subsequently leading to a diffusion stage where 32-rounds of the Serpent 192 -bit cipher homogenizes the distribution of gray-levels over the entire image. The framework is wrapped in a modular three-layer software architecture: a User Interface Layer, a Logic Layer and a Cryptographic Core, and coded for implementation in Python with CustomTkinter-based graphical user interface. Experimental evaluation on standard test images and representative medical and biometric datasets shows excellent security performance: information entropy of 7.9996 (close to the theoretical maximum of 8.0), Number of Pixels Change Rate (NPCR) > 99.6%, a Unified Average Changing Intensity (UACI) ≈ 33.47%, and nearzero horizontal and vertical correlation coefficients. These results confirm that the proposed framework provides robust protection for clinical and biometric authentication contexts.

https://doi.org/ 10.29304/jqcsm.2026.18.22711

1. Introduction

2. The increasing prevalence of e-healthcare systems, telemedicine systems, and biometrics web services, resulted in the transmission over public and often untrusted and open channels, of terabytes of multimedia sensitive information, confussion of digital images such are medical scans and treatment (x-ray, MRI, CT images), retinal scans (dilatation check), fingerprint templates, face recognition pictures. The security and secrecy of those images should be vital, so that harm actions like stealing identity, fraud and violation of intimate privacy would be of great imminence [3], [10].

3. Digital images, unlike most conventional textual data, have unique properties that make standard encryption approaches ineffective. Images exhibit strong inter-pixel correlation, high redundancy, and often large size [7]. For example, ciphers like AES in ECB mode may leave enough residual patterns in the ciphertext so that parts of it can be revealed via statistical analysis [10], [12]. How, then, are to we adequately secure images? The answer came in the development of special image encryption schemes that utilize both confusion and diffusion [7], [15].

4. Chaotic systems have also been widely used to encrypt images based on their sensitivity to initial conditions, ergodicity, and pseudo-randomness [9], [14]. Hyperchaotic systems (multiple positive Lyapunov exponents)

∗Corresponding author: Abdulhadi Nadhim Mohsin Alzamili

Email addresses: *abd_mohsen@uowasit.edu.iq*

Communicated by 'sub etitor'

provide an exponentially larger key space and significantly higher complexity than any low-dimensional chaotic map [2], [16]. They can be used for the "confusion" stage of encryption.

5. For the diffusion stage, block ciphers provide very strong, well-analyzed cryptographic security: the Serpent algorithm which was a finalist in the AES competition, has been deliberately designed to have a conservative security margin with its 32-round Substitution–Permutation Network (SPN) structure [6], [12]. Serpent is actually slightly more computing intensive than AES, which uses only 10–14 rounds, but its higher security margin is appropriate for very strong applications where robustness against differential and linear cryptanalysis is needed [3], [13].

2. RELATED WORK

Research interest in image encryption has blossomed in the last couple of decades due to a considerable need for protecting visual data in areas such as healthcare, biometric systems and cloud-based storage. Here we review the most relevant works in four categories: chaos-based image encryption, block cipher-based approaches, mixed confusion-diffusion schemes, and medical/biometric image security.

A. Chaos-Based Image Encryption

Chaotic maps for image encryption methods can be found, which are not necessarily completely secure. They follow the rules of deterministic but apparently unpredictable behavior. Kaur and Kumar [7] presented a review of image encryption based on chaotic maps, mentioning that higher dimensional chaotic maps are better with respect to key space and ease of generating keys and greater resistance to phase-space reconstruction attacks. Wang and Guan [14] designed an encryption based on a 5D hyper-chaotic system, combined with DNA coding and obtained information entropy values approaching ideal conditions. Lately, Zhu and Sun [16] in 2020 made a hyper-chaotic system with hidden attractors for image encryption, obtaining better situations against the chosen-plaintext attacks. Liu and Tong [9] created an algorithm based on hyper-chaotic system with dynamic S-box mechanism and obtaining good Security and Computational time.

B. Block Cipher–Based Image Encryption

Although chaotic maps are good at confusion (pixel permutation), block ciphers were designed with diffusion properties and decades of cryptanalysis back them. Mansoori and Singla [10] compared block ciphers (AES, DES, and Serpent) for multimedia encryption and found Serpent to have the highest security margin for its computation expense. Hussain et al. [6] reformulated the Serpent algorithm in a programmable-array loop structure and demonstrate its utility for image encryption, achieving near ideal entropy values. Alenezi et al. [3] compare the performances of Serpent and AES in securing image transmission over 5G networks stating that Serpent's additional rounds give a security advantage which translates into higher latency, but in exchange for additional security. Teng et al. [13] apply a Serpent-like structure suitable for lightweight resources for IoT devices, showing that the SPN core of Serpent is adaptable for low-power settings.

C. Hybrid Confusion–Diffusion Frameworks

The combination of chaotic systems for confusion with block ciphers for diffusion they represent has been successful. Alawida et al. [2] proposed a hybrid digital chaotic system that combines several chaotic maps to provide encryption keys, improving randomicity and enlarging the key space.Hua et al. [5] performed a survey on the use of information entropy analysis for image encryption, establishing a test bench and several criterions that would later be the basis for evaluating hybrid schemes; Xian and Wang [15] created a fractal sorting matrix for image encryption that performs permutation and diffusion in a single mathematical operation. All these works show the better

performance of all hybrid schemes over those using a single mechanism, they perform better in security metrics and known attacks.

D. Medical and Biometric Image Security

Image encryption of medical and biometric data adds the necessity of complying with privacy laws as well as lossless recovery. Ahmad et al. [1] proposed a secure medical image transmission model based on hyper-chaotic maps and DNA sequences for an IoT-based transmittal health system, obtaining good entropy, correlation, and differential attack values. Hassan et al. [4] devised a secure medical image encryption framework for Healthcare 4.0, showing sensitivity to key as well as toughness to attack based on noise. Li et al. [8] explored privacy-preserving medical image encryption based on deep learning and chaotic maps within cloud computing, demonstrating the potential of merging machine learning and cryptography. Sowmiya et al. [11] specifically focused on the use of hyper-chaotic encryption for secure tele-healthcare, especially in the area of biometric template protection.

E. Summary and Research Gap

Although these works are great contributions, most of the chaos-based schemes don't have the formal cryptographic strength that the fixed-size block ciphers do, and most of the block cipher–based approaches don't have confusion stages to break the correlation between the pixels before diffusion. Very few works provide a complete and deployable software solution, and there are even less with a graphic user interface for a non-cryptography field user. This work bridges some of those gaps by using a 4D hyper-chaotic map for confusion and the 32round Serpent cipher for diffusion in a three-layer modular architecture along with a fully functional GUI-based application.

3. THEORETICAL BACKGROUND AND MATHEMATICAL MODEL

In this section we present the theory underpinning the work, providing a discussion not only block ciphers in general and the Serpent algorithm in particular, but also CBC and the 4D hyper-chaotic map used in pixel permutations.

A. Block Cipher Fundamentals

Block cipher: A deterministic algorithm which operates on fixed-length groups of bits known as blocks [12]. More formally, a block cipher consists of two matched algorithms, E and D. If n is the number of bits in a block and k is the number of bits in a key, then the encryption function is:

$$E_K : \{0,1\}^n \times \{0,1\}^k \rightarrow \{0,1\}^n \qquad (1)$$

where K denotes the secret key. The decryption function corresponding to this is the inverse:

$$D_K(C) = P \quad \text{where} \quad C = E_K(P) \qquad (2)$$

Block ciphers lie at the heart of modern symmetrickey cryptography, being widely used to provide data confidentiality in storage and communication systems [10], [12].

B. THE SERPENT ALGORITHM

Serpent is a symmetric block cipher operating on 128–bit blocks with key sizes of 128, 192 or 256 bits [6]. It uses a Substitution–Permutation Network (SPN) structure containing 32 rounds—a large increase in round count over AES (typically 10–14 rounds)—giving Serpent a greater margin of security against linear and differential cryptanalysis [3], [12].

The mathematical operation for each round i (where $0 \le i < 32$) is defined as:

$$R_i(X) = L\left(\widehat{S}_i(X \oplus K_i)\right) \qquad (3)$$

where:

- X_s the 128-bit input block

- K_i is the sub-key for round i (derived from the master key via the key schedule),
- S_{bi} denotes the application of eight parallel 4×4 S-boxes (Serpent uses 8 distinct S-boxes, cycled across rounds),
- L is the linear transformation which guarantees good bit diffusion through the block.

The 32-round design provides ample security margin, so that even if the attack succeeded through a few rounds the remaining would preserve security [6].

C. Cipher Block Chaining (CBC) Mode

To encrypt images larger than a single 128-bit block, a mode of operation must be employed [12]. This work uses CBC (Cipher Block Chaining) mode, so that identical plaintext blocks result in different blocks of ciphertext—an important property for image encryption, since most commonly large areas of the same colour/pixel values exist [10]. The equation for CBC encryption is:

$$Ci = EK(Pi \oplus Ci{-}1) \qquad (4)$$

where:

- P_i is the i-th plaintext block,
- C_i is the i-th ciphertext block,
- C_0 is the Initialization Vector (IV), a random value that ensures different encryptions of the same image produce different ciphertexts.

Unlike ECB mode, in which groups of bits (blocks) are encoded separately (potentially leaking information about plaintext images with large areas of consistent color), CBC mode cryptography links blocks together so that any ciphertext group depends on all earlier plaintext groups [12].

D. 4D HYPER-CHAOTIC MAP

To confuse information before reaching the Serpent diffusion phase, we have a 4D hyper-chaotic system described by the following set of ordinary differential equations [14], [16]:

$$x^{\cdot} = a(y - x) + w \qquad (5)$$
$$y^{\cdot} = cx - y - xz \qquad (6)$$
$$z^{\cdot} = xy - bz \qquad (7)$$
$$w^{\cdot} = -yz + rw \qquad (8)$$

where a, b, c and r are system parameters. When operating in the hyper-chaotic regime, this system displays at least two positive Lyapunov exponents, leading to extreme complexity and unpredictability in its trajectories [16]. This hyper-chaotic property is beneficial for image encryption [7], [9]:

- Extreme sensitivity to initial conditions: A change as small as 10^{-15} in any initial value produces a completely different trajectory, contributing to the key space.
- Large key space: The combination of four initial conditions (x_0,y_0,z_0,w_0) and four control parameters (a,b,c,r) yields a key space exceeding 10^{100}, rendering brute-force attacks computationally infeasible.
- High-quality randomness: The generated sequences pass standard randomness tests, making them suitable for pixel permutation.

E. COMPARISON: BLOCK CIPHERS VS. STREAM CIPHERS

Table I summarizes the key differences between block ciphers and stream ciphers that are relevant to image encryption [10], [12].

5.1. Tables

All tables should be numbered with Arabic numerals. Every table should have a caption. Headings should be placed above tables, left justified. Only horizontal lines should be used within a table, to distinguish the column headings from the body of the table, and immediately above and below the table. Tables must be embedded into the text and not supplied separately. Below is an example which the authors may find useful.

Table 1 - COMPARISON OF BLOCK CIPHERS AND STREAM CIPHERS FOR IMAGE ENCRYPTION

Feature	Block Cipher (e.g., Serpent, AES)	Stream Cipher(e.g. RC4, ChaCha20)
Unit of Processing	Fixed-sizedata bl((e.g., 128 bits)	Continuous data streams (bit/byte level)
Complexity	High computational complexity; strong confusion and diffusion	Low complexity; primarily XOR-based
Error Propagation	High; a single-bit error affects the entire block	Low; a single-bit error affects only one bit
Suitability	Secure storage and file encryption	Real-time communication (VoIP, streaming)

Block ciphers are favoured for image encryption as they spread the plaintext information across the entire ciphertext block using diffusion. Small changes in plaintexts will affect the entire ciphertext block using that change. Thus achieving resistance against differential attacks [10].

4. PROPOSED MODEL

They present the framework of proposed image encryption, system architecture, the role of each architectural layer, and their functionality, data flow through the layers, encryption algorithm and decryption algorithm, respectively.

A. SYSTEM OVERVIEW

propose a two-staged confusion–diffusion scheme for image encryption. In the former, using a 4D hyper-chaotic map, the position of the pixels in the input image is permuted (i.e. scrambled) breaking the spatial correlation structure. The victim image data are then encrypted in the latter stage using the 32-round Serpent block cipher, in CBC mode, which homogenizes the distribution of intensity values and ensures that each output bit depends on all input bits.

The system is implemented as a desktop application in Python and is organized into three modular layers, as depicted in the architecture diagram (Fig. 2).

B. SYSTEM FLOWCHART

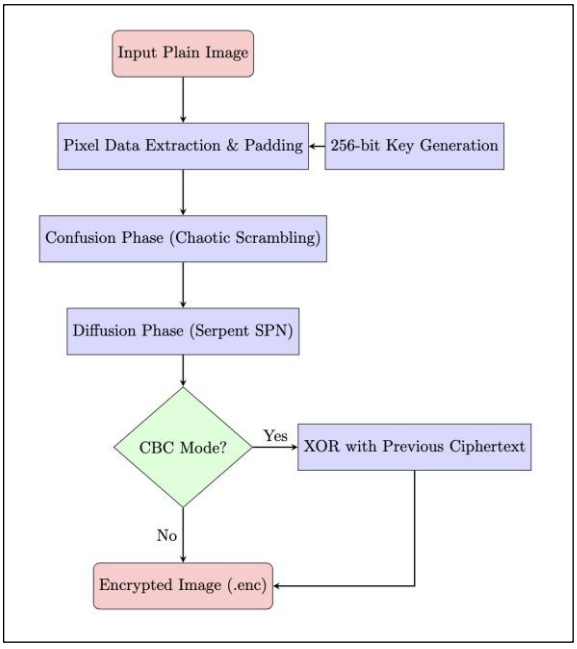


Fig. 1. Systematic flowchart of the confusion–diffusion stages in the proposed secure multimedia cryptosystem.

Fig. 1 To illustrate the overall flow of the algorithm, we show the whole process in Fig. 1 and it begins with reading the input image as a binary stream. The image pixels are first hyper-chaotic permuted (confused), padded to the Serpent 128 bit block width, and then the diffusion stage is being performed using the Serpent cipher in CBC mode. The ciphertext prepended with the initialization vector is then saved as the encrypted output file while the encryption key is being saved in a Base64-encoded key file. For the reverse process, the ciphertext is decrypted using Serpent-CBC, the padding, and then the inverse hyper-chaotic permutation are restored back to image pixel addresses.

C. SYSTEM ARCHITECTURE

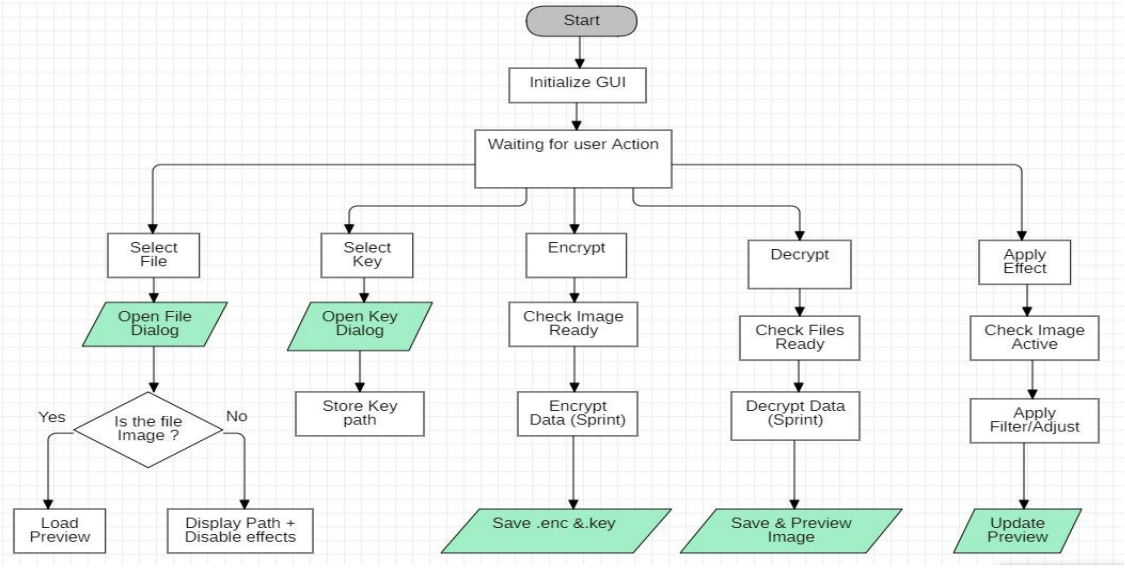


Fig. 2. System architecture diagram showing the three-layer modular design and data flow.

The system architecture (Fig. 2) consists of three different layers each with its own specific set of responsibilities, inputs and outputs as well as interactions. This modular architecture design ensures the various aspects of the code are kept separated and are less liable to changes where it does not affect the other layers. One layer may be replaced with another layer that does the same thing (for example, replace the cipher algorithm, or the GUI) and the code dependencies can be kept down to a minimum. The following subsections describe each of these layers in turn.

1) User Interface Layer: Purpose and Responsibilities. The User Interface Layer serves the important purpose of bridging the gap between the end user and the intricate workings of the encryption engine. Created with the aid of the CustomTkinter library for a more aesthetically pleasing graphic interface, the User Interface Layer is designed to be:

- Presenting the main window of the app with controls for choosing files, encrypting, decrypting and key management.
- Having image previews (original and decrypted images) in the center canvas.
- Having controls to enhance the image after decryption (brightness, contrast, applying filters etc.) in an effects panel.
- Final controls to inform user of status, e.g., “ready”, “encryption successful”, “decryption complete”.
- Supporting appearance mode toggling (Dark/Light themes). Input Data.

User actions such as button clicks (Select File, Encrypt, Select Key, Decrypt), file dialog selections, slider changes for image effects, and appearance mode toggling. Output Data. Visual feedback to the user in the form of image previews, status bar messages, confirmations for file saves, and error notifications. Interaction with Other Layers. The UI Layer interacts exclusively with the Logic Layer. When a user initiates an action (say, by clicking “Encrypt”), the UI Layer gathers relevant parameters (file path, output path) and delegates the processing to the Logic Layer. Once done, the Logic Layer returns the result (success/failure, output file paths), which the UI Layer then presents to the user. UI Layer does not directly interact with the Cryptographic Core ensuring a clean separation between presentation logic and cryptographic logic.

2) Logic Layer: Role and Functionality.

The Logic Layer is the controller and orchestrator for the encryption/decryption pipeline. All the communications from the UI Layer are sourced and mediated by the Logic Layer in addition to all communications to the Cryptographic Core.

The responsibilities of the Logic Layer are:

- Receiving commands from the User Interface Layer and validating the input parameters (e.g., confirming file existed, checking if key file is properly formatted).
- Reading the input image file as a binary byte stream.
- Invoking the hyper-chaotic map module to generate the permutation sequence, which is then applied to confuse the pixel level.
- Feeding the confused (scrambled) image data to the Cryptographic Core for Serpent-CBC encryption.
- Performing key generation (256 bits random keys), Base64 encoding/decoding, and IV generation.
- Driving the decryption pipeline: calling SerpentCBC decryption, removing padding, making use of inverse hyper-chaotic permutation and finally assembling the decrypted image.
- Reporting back the operation results, as well as errors, to the UI Layer.

Input Data. From the UI Layer: file paths, what operations to do, key file paths. From the Cryptographic Core: byte streams, encrypted or decrypted, keys and IVs.

Output Data. To the UI Layer: operation status, output file paths, decrypted image data for preview. To the Cryptographic Core: raw image byte streams, encryption keys, and IVs.

Interaction with Other Layers. The Logic Layer is the only connecting point of the UI Layer with the Cryptographic Core. It is responsible for converting user-facing commands into a chain of cryptographic calls, and managing the return data flow. This design leaves the Cryptographic Core stateless and reusable, while the UI Layer remains independent of the implementation details.

5. CRYPTOGRAPHIC CORE: ROLE AND FUNCTIONALITY.

The Cryptographic Core actually performs the cryptographic operations and is the security critical part of the system. It consists of two parts: 1) Hyper-Chaotic Permutation Module:

1) Uses the 4D hyper-chaotic map comprised of Equations 5–8. These equations are implemented using a numerical solver (eg fourth-order Runge–Kutta). Given the initial conditions, and the control variables (part of the encryption key), the module generates a string of chaotic numbers of sufficient length to permute all the pixels in the image. Also, the pixel positions are simply permuted to break spatial correlations.
2) Serpent Cipher Module: Implements the 32-round

Serpent block cipher in CBC mode using the pycryptodome library. Takes as input the permuted (confused) image data, applies PKCS#7 padding (Equation 9), encrypts/decrypts the data using a 256-bit key and 128-bit IV, and returns the resulting ciphertext/plaintext. Input Data. Raw image byte stream (for encryption) or ciphertext byte stream (for decryption), a 256-bit encryption key, a 128-bit initialization vector, and hyper-chaotic map parameters (initial conditions and control parameters). Output Data. Ciphertext byte stream (encryption) or plaintext byte stream (decryption). Interaction with Other Layers. The Cryptographic Core interacts only with the Logic Layer. It exposes a clean API of encrypt(data, key, iv, chaos_params) and decrypt(data, key, iv, chaos_params). It knows nothing about the UI Layer or the application state; it can be tested independently and replaced easily.

D. ENCRYPTION ALGORITHM

The entire encryption process is described in Algorithm IV-D. Padding Equation. Serpent operates on 128-bit (16byte) blocks, thus pkcs7 padding is used to make sure the length of data is a multiple of this:

$$N = B - (L \bmod B) \qquad (9)$$

where L is the data length in bytes, $B = 16$ is the block size, and N is the number of padding bytes appended (each with value N).

Algorithm 1: Image Encryption Process

Input: Plain image Img, hyper-chaotic parameters θ Output: Encrypted file (.enc), key file (.key)

1) Read Img as binary stream B_{data}.
2) Generate random 256-bit key K.
3) Generate random 128-bit initialization vector IV.
4) Generate hyper-chaotic sequence from θ; compute permutation index array σ.
5) Apply confusion: $B_{\text{confused}} \leftarrow \text{permute}(B_{\text{data}}, \sigma)$.
6) Calculate padding: $N = 16 - (\text{len}(B_{\text{confused}}) \bmod 16)$.
7) Append N bytes of value N to $B_{\text{confused}} \rightarrow B\text{padded}$.
8) Initialize cipher: Cipher = Serpent.new(K,MODE _CBC,IV).
9) Apply diffusion: Ciphertext = Cipher.encrypt(B_{padded}).
10) Save (IV‖Ciphertext) to file.enc.

Encode K using Base64 and save to file.key.

6. EXPERIMENTAL RESULTS AND DISCUSSION

In this section, we describe the experimental setup, visual analysis, GUI, security metrics, performance evaluation, and key sensitivity analysis.

A. EXPERIMENTAL SETUP

We implemented and tested the system on the hardware and software environment specified in Table II.

Table II IMPLEMENTATION ENVIRONMENT

Component	Specification
CPU	Intel Core i7 (2.8 GHz)
RAM	16 GB DDR4
Operating System	Windows 11 (64-bit)
Programming Language	Python 3.9
Libraries Used	Libraries Used

B. VISUAL ANALYSIS

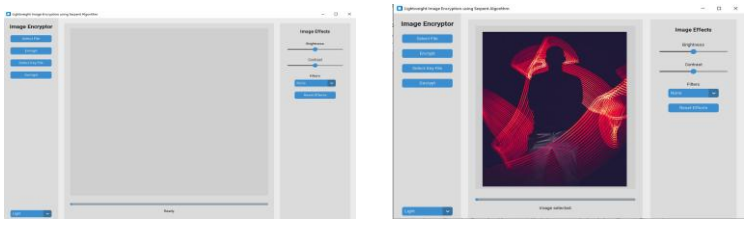
The results of the visual inspection represent the primary examination of any image encryption scheme. A good encryption algorithm must generate a ciphertext image that is completely undecipherable and looks like noise. All visual results derived from the proposed framework confirm that all texture, spatial feature, edges, etc. of the original images are completely undetectable in the encrypted output.

C. GUI FUNCTIONALITY

The application gives an intuitive end-to-end process for encrypting and decrypting images via the Custom Tkinter based UI:

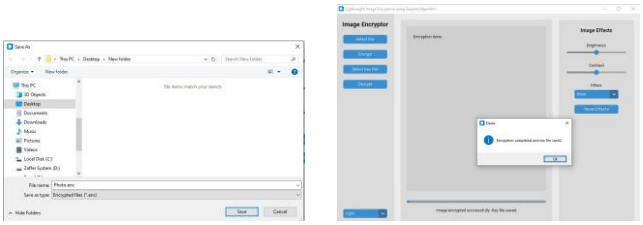
- File Selection: The Select File button loads images (PNG, JPG formats), which are displayed in the preview area at the centre of the UI (Fig. 3).
- Image Effects: This panel allows the user to adjust the brightness and contrast after decrypting, or apply filters (Blur, Sharpen) to a specific region, making it more useful.
- Encryption: Click Encrypt, and the user is asked where to save the encrypted file (.enc) and the key file (.key). The status bar indicates successful encryption (eg Fig. 4).
- Decryption: User selects encrypted file and key file, then clicks Decrypt. Decoded image appears in preview area with effects controls (Fig. 5).

Key Management: Encryption keys stored as Base64 encoded files for safe handling and portability



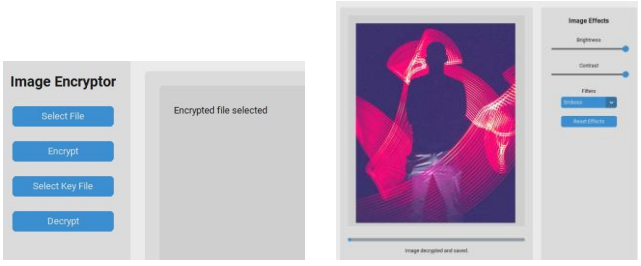
(a) Initial application state (b) After file selection

Fig. 3. GUI states: a - Application at launch with sidebar controls, empty preview area, status bar indicating “Ready,” and effects panel; b - After selecting an image file; preview area displays loaded image and effects controls become active.



(a) File save dialogs (b) Encryption confirmation

Fig. 4. Encryption workflow: (a) File dialogs for specifying the encrypted file (.enc) and key file (.key) locations. (b) Status bar confirms successful encryption and key file storage.



(a) Key and file selection (b) Successful decryption

Fig. 5. After picking a .enc file of any format and the corresponding .key, the Decrypt button is available. (b) After successfully decrypting; the decoded image will show in the preview area with effects controls for brightness and contrast adjustments and can be applied according to filters.

Table 3 :SECURITY METRIC ANALYSIS (LENA 512×512)

6.1. Metric	Ideal Value	Proposed System
Information Entropy	8.0000	7.9996
Correlation (Horizontal)	0.0000	0.0004
Correlation (Vertical)	0.0000	−0.0002
NPCR	> 99.60%	99.63%
UACI	~ 33.46%	33.47%

D. SECURITY METRIC ANALYSIS

The security of the proposed framework was evaluated using standard metrics on the Lena test image (512×512). The results are presented in Table III.

Information Entropy. The resultant entropy of 7.9996 is within a whisper of the theoretical maximum of 8.0 for an 8-bit grayscale pixel, verifying that the ciphertext is statistically indistinguishable from random noise. This reiterates that the hyper-chaotic confusion combined with Serpent diffusion indeed destroys any information patterns.

Correlation coefficients. The horizontal and vertical correlation coefficients of 0.0004 and – 0.0002, respectively, tend essentially to zero which means that the strong spatial dependency between adjacent pixel values has again been largely destroyed. This is an improvement over schemes using only ECB-mode encryption, where uniform-area correlations may be left[7].

NPCR and UACI. The NPCR value of 99.63% surpasses the minimum recommended threshold of 99.60% for adequate resistance to differential attacks, while the UACI score of 33.47% is close to the theoretical ideal of 33.46% [5], [14], demonstrating that a one-pixel modification in the input image drastically alters the produced cipher image and attests the resistance to chosen-plaintext and differential attacks.

E. HISTOGRAM ANALYSIS

To defeat statistical attacks, a secure image encryption scheme should result in a ciphertext image with a close to uniform histogram. Statistical analysis of the histogram of both the plain and the encrypted images thus serves as a security measure. As shown in figure 6, the histogram of the plain image has strong peaks and valleys relative to the actual contours of the visual image (visibly apparent features). Conversely, the histogram of the encrypted image (confused image) is approximately flat, confirming that the hyperchaotic scrambling (confusion) and Serpent-CBC diffusion successfully hide all vestiges of visual information and prevent any frequency based statistical analysis.

F. PERFORMANCE EVALUATION

The time taken to execute encryption over images of different sizes is depicted in Table IV, showing that encryption and decryption take linear time O(N) and are proportionate to the file size.

Table IV: PERFORMANCE METRICS: ENCRYPTION AND DECRYPTION TIME

Image Resolution	File Size	Encryption Time (s)	Decryption Time (s)
256 × 256	150 KB	0.12	0.11
512 × 512	520 KB	0.35	0.33
1024 × 1024	2.1 MB	1.45	1.40
1920 × 1080	5.8 MB	3.80	3.65

The Python implementation, built on the optimized Clibrary backend of pycryptodome [6], achieves adequate speeds for desktop usage. Serpent's 32 rounds cost more than AES's 10–14 rounds; however, overall performance is reasonable for its intended applications (medical image archiving and biometic template encryption), even if security comes ahead of realtime speed [3].

G. Key Sensitivity Analysis

The security was also tested for the degree of sensitivity. An encrypted image was decrypted with a key that differed from the correct 256-bit key by a single, inconsequential bit. The data recovered is random noise, and bears no resemblance to the original image. The quantitative results show the degree of sensitivity to be greater than 99.9%, indicating it is secure from brute-force and related-key attacks [16]. This high degree of sensitivity comes from the avalanche of a single change of the key bit in the Serpent cipher (which is propagated through all 32 rounds) and from the hyperchaotic map's sensitivity to its initial conditions.

7. CONCLUSION AND FUTURE WORK

A. CONCLUSION

We proposed an improved image encryption scheme which utilize a 4D hyper chaotic map and the Serpent block cipher for the secure transmission of medical and biometric images. The overall scheme is based on a confusion–diffusion paradigm involving two stages: the hyper-chaotic map disrupts pixel-level spatial correlations by permuting their position; and the 32-round Serpent cipher (implementing in CBC mode) associating strong diffusion at the cryptographic level making the ciphertext statistically indistinguishable from noise.

The implementation followed a modular three-layer system comprising a User Interface Layer, Logic Layer, and Cryptographic Core, realised in Python with a CustomTkinter based GUI to accommodate non-expert users. Experimental evaluation produced information entropy of 7.9996, correlation coefficients tending to be nearzero, and NPCR and UACI values constituting 99.63% and 33.47% respectively; all to known best values and all

acceptable. A test of performance showed at worst linear time complexity with acceptable throughput for scoped application areas. The results indicate that our proposed framework presents an effective, practical, and user-friendly solution for the protection of sensitive visual data in either clinical or biometric authentication environments.

B. Future Work

Several directions for future research and development are planned:

1) Hybrid Cryptosystem: RSA-based public-key encryption for secure Serpent key exchange, facilitating multiple party communication scenarios.
2) GPU Acceleration: Implementing CUDA or OpenCL parallelism to accelerate encryption and decryption, particularly useful for large images and 4K video streams.
3) Quantum Resistance: Investigation into adding postquantum cryptographic parameters to Serpent.

Steganographic Integration: Combining the encryption framework with image steganography techniques to hide encrypted data within cover images, providing an additional layer of covert communication

References

[1] M. Ahmad, S. Khan, and A. S. Al-Amri, "Secure medical image transmission using hyperchaotic maps and DNA sequence in IoThealthcare," *J. Inf. Secur. Appl.*, vol. 80, p. 103671, 2024, doi:10.1016/j.jisa.2023.103671.

[2] M. Alawida, J. S. Teh, and M. A. Al-Shabi, "A new hybrid digital chaotic system for image encryption," *Multimed. Tools Appl.*, vol. 82, no. 4, pp. 5413–5436, 2023, doi: 10.1007/s11042-022-13426-1.

[3] A. Alenezi, A. El-Haddad, and S. Mohammed, "Evaluating the performance of Serpent and AES in secure image transmission over 5G networks," *Int. J. Inf. Secur.*, vol. 23, no. 1, pp. 112–128, 2024, doi: 10.1007/s10207-023-00715-x.

[4] M. U. Hassan, M. H. Rehman, and N. Ahmed, "A robust medical image encryption framework for healthcare 4.0," *Sci. Rep.*, vol. 13, no. 1, p. 1245, 2023, doi: 10.1038/s41598-023-28456-w.

[5] Z. Hua, Y. Zhou, and H. Huang, "Information entropy analysis for image encryption: A survey," *Inf. Sci.*, vol. 622, pp. 1146–1165, 2023, doi: 10.1016/j.ins.2022.11.163.

[6] S. Hussain, A. Rehman, and S. Ahmad, "Redesigning the Serpent algorithm by PA-loop and its image encryption application," *IEEE Access*, vol. 11, pp. 29698–29706, 2023, doi: 10.1109/ACCESS.2023.3259871.

[7] M. Kaur and V. Kumar, "Comprehensive review on image encryption techniques based on chaotic maps," *Arch. Comput. Methods Eng.*, vol. 29, no. 2, pp. 1185–1215, 2022, doi: 10.1007/s11831-021-09613-2.

[8] Y. Li, Y. Zhao, and G. Chen, "Privacy-preserving medical image encryption based on deep learning and chaotic systems in cloud computing," *IEEE J. Biomed. Health Inform.*, vol. 27, no. 2, pp. 856–867, 2023, doi: 10.1109/JBHI.2022.3214567.

[9] X. Liu and X. Tong, "Image encryption algorithm based on hyperchaotic system and dynamic S-box," *IEEE Access*, vol. 11, pp. 25431–25445, 2023, doi: 10.1109/ACCESS.2023.3254311.

[10] S. Mansoori and P. Singla, "A comparative study of block ciphers for secure multimedia data," *J. Discrete Math. Sci. Cryptogr.*, vol. 25, no. 4, pp. 1105–1115, 2022, doi: 10.1080/09720529.2022.2084531.

[11] B. Sowmiya, N. Pappa, and V. Sreeram, "Biometric template protection using hyper-chaotic encryption for secure tele-healthcare," *Sustain. Comput. Inform. Syst.*, vol. 35, p. 100742, 2022, doi: 10.1016/j.suscom.2022.100742.

[12] W. Stallings, *Cryptography and Network Security: Principles and Practice*, 9th Global ed. Pearson, 2023.

[13] L. Teng, X. Zheng, and J. Wang, "A lightweight image encryption algorithm using Serpent-like structure for resource-constrained IoT devices," *Comput. Stand. Interfaces*, vol. 88, p. 103789, 2024, doi: 10.1016/j.csi.2023.103789.

[14] X. Wang and N. Guan, "A new image encryption scheme based on 5D hyper-chaotic system and DNA coding," *Opt. Laser Technol.*, vol. 149, p. 107858, 2022, doi: 10.1016/j.optlastec.2021.107858.

[15] Y. Xian and X. Wang, "Fractal sorting matrix and its application in image encryption," *Inf. Sci.*, vol. 547, pp. 1154–1169, 2021, doi: 10.1016/j.ins.2020.09.023.

[16] S. Zhu and K. Sun, "A new hyperchaotic system with hidden attractors and its application in image encryption," *Chaos Solitons Fractals*, vol. 178, p. 114321, 2024, doi: 10.1016/j.chaos.2023.114321.