

Biometric-Based Secure Encryption Key Generation Using Convolutional Neural Networks and Particle Swarm Optimization

Sahera A. S. Almola, Raidah S. Khudayer, Hameed Abdulkareem Younis

Department of Computer Information Systems, College of Computer Science and Information Technology, University of Basrah, Basrah, Iraq

E-mail: sahera.sead@uobasrah.edu.iq, raidah.khudayer@uobasrah.edu.iq, hameed.younis@uobasrah.edu.iq

*Corresponding author

Keywords: biometric verification, fingerprints, deep learning, particle swarm optimization (ps) algorithm, encryption key generation

Received: December 7, 2024

With the rapid expansion of computer networks and information technology, ensuring secure data transmission is increasingly vital—especially for image data, which often contains sensitive information. This research presents a biometric-based encryption system that uses fingerprint recognition and deep learning to generate strong, random encryption keys. Two convolutional neural networks (CNNs) are employed: one to verify identity based on a user's ID and another to extract fingerprint features for key generation. These keys are optimized using Particle Swarm Optimization (PSO), enhancing their randomness and resistance to brute-force attacks.

The system generates keys in real-time, eliminating the need for storage and minimizing the risk of theft or leakage. To further improve security, encryption keys are automatically updated after every ten messages, with different keys generated from multiple fingerprints of the same individual. Testing with the SOCOFing dataset (6,000 original and 49,270 synthetic images) achieved 99.75% identity verification and 99.83% classification accuracy. Performance metrics—entropy of 7.89, correlation factor of 0.00628, and zero repetition—demonstrate high robustness. This approach offers a secure, adaptive, and personalized encryption method ideal for sensitive domains like finance and healthcare.

Povzetek: Opisana je izvirna metoda za generiranje varnih šifrirnih ključev z uporabo prstnih odtisov, CNN modelov in optimizacije roja delcev (PSO)

1 Introduction

Internet and network users share millions of color images daily, which are utilized in various applications such as telemedicine, remote learning, business, and military operations. Color images, in particular, often contain sensitive and detailed information, making them prime targets for unauthorized access and cyberattacks. Securing these images is crucial not only to prevent data loss during transmission but also to protect sensitive information from attackers. Various techniques are employed to secure digital images, such as watermarking, steganography, and image encryption. Encryption operates in two main stages: encryption and decryption. During encryption, the input image is transformed into an unreadable form using a secret key, while in decryption, the content is restored using the same key [1]. The encryption key is a fundamental element in the encryption and decryption processes, and it significantly determines the security system's strength. However, a critical challenge faced by encryption systems lies in managing the encryption key itself [2]. Traditional encryption methods require transmitting the encryption key to the recipient to decrypt the data. This approach introduces vulnerabilities, as any exposure of the key during transmission could lead to the compromise of the

encrypted data. Consequently, there is an increasing need for systems that dynamically generate encryption keys on-demand at the user's end, eliminating the need for key transmission over networks [3]. This innovative approach ensures that the encryption key is generated locally each time data is decrypted, significantly reducing risks associated with key interception. It also eliminates the need for key exchange, adding an extra layer of security since unauthorized parties cannot generate the key even if communication is intercepted.

The keyless exchange method, when combined with biometric verification, offers a highly secure solution by minimizing the risk of key theft. This approach aligns with the methodology presented in this research. However, implementing such a solution poses significant challenges in the fields of secure computing and key management, as it requires a robust system to ensure the consistent and accurate generation of keys [4]. The importance of this research lies in emphasizing the generation of encryption keys locally at the user's end to safeguard data and mitigate risks associated with key transmission over networks. This is particularly critical for securing color images, as their high information content often correlates with increased sensitivity, making them especially vulnerable to sophisticated attacks.

To address these challenges, advanced techniques based on artificial intelligence and machine learning,

particularly deep learning, have emerged. One notable technique involves using deep learning to generate encryption keys from fingerprints. This method leverages the extraction of unique features from fingerprints, converting them into robust, non-repetitive encryption keys to ensure high data security [5]. This method addresses limitations in traditional encryption systems, such as the need for key transmission over networks. Since a fingerprint is a unique biometric identifier that cannot be easily copied or mimicked, it serves as an ideal source for generating encryption keys. Moreover, deep learning enhances the accuracy and strength of the generated keys by utilizing deep neural networks to analyze biometric images and extract unique features for each fingerprint [6]. This approach also resists advanced threats, including brute-force and quantum encryption attacks, by dynamically generating encryption keys in real-time. The added layer of complexity and secrecy prevents unauthorized parties from accessing the keys, even if communication data is partially intercepted [7]. The integration of deep learning in generating encryption keys from fingerprints represents a significant advancement in information security. This approach combines robust security measures with individual privacy, paving the way for building encryption systems that are highly resistant to breaches and better equipped to address modern security challenges. The remainder of this paper is organized as follows: Section 2 reviews related works, while Section 3 provides background on the key techniques utilized in this research. Section 4 explains the management of secret keys. Section 5 details the proposed method. Section 6 focuses on experimental results and performance analysis. Section 7 discusses the results, and Section 8 concludes this study.

2 Related works

The integration of biometric data, chaotic systems, and deep learning in encryption key generation has been a prominent research area. Various studies have explored innovative approaches to enhance the security and robustness of encryption systems. Hashem and Kuban (2023) [8] introduced a system that leverages fingerprint biometrics to generate long, random encryption keys. The approach involves preprocessing fingerprint images to remove noise, utilizing a modified VGG-16 convolutional neural network (CNN) to extract unique features, and employing transfer learning to build a key cryptographic key management in systems with increasing users and applications. They evaluated five key management systems, including Hashicorp Vault and Pinterest Knox, focusing on features such as security, scalability, and access control. The study concluded that

generation model without the need for retraining. Erkan et al. (2024) [9] proposed a secure image encryption framework that combines a chaotic logarithmic map with a deep CNN for key generation. Their system incorporates advanced operations such as permutation, DNA encoding, diffusion, and bit-reversal to ensure security. The robustness of this framework was validated through comprehensive analyses, including key sensitivity and resistance to various attacks, demonstrating superior performance compared to traditional encryption methods.

Quinga Socasi, Zhinin-Vera, and Chang (2020) [10] developed a method for generating encryption keys from alphanumeric passwords using an autoencoder neural network. Their experiments revealed that this method outperforms conventional algorithms, particularly when encrypting small text files, making it highly resistant to cracking attempts. Wu et al. (2022) [11] presented a biometric key generation framework that uses fingerprints to achieve over 1024-bit key strength and 98% accuracy. However, their method depends on a predefined pipeline and fuzzy extractors for key stabilization. In contrast, the method proposed in this research dynamically extracts high-resolution fingerprint features using deep learning models, ensuring greater adaptability across datasets. These features are combined with chaotic encryption systems to enhance randomness and security. Furthermore, Particle Swarm Optimization (PSO) is employed to optimize the generated keys, achieving over 99% accuracy and producing 1024-byte keys without requiring stabilization layers. This approach demonstrates superior flexibility and security for real-world IoT applications. Alesawy and Muniyandi (2016) [12] investigated data security in cloud environments using random encryption keys. Their study analyzed the impact of incorporating Elliptic Curve Diffie-Hellman (ECDH) keys and demonstrated significant improvements in efficiency and performance by integrating Artificial Neural Networks (ANNs) with ECDH and genetic algorithms, despite increased processing times for larger datasets. Saini and Sehrawat (2024) [13] proposed a technique for generating unique encryption keys by combining an autoencoder network with hashing techniques and prime numbers derived from the MNIST dataset. To enhance security, the system incorporates XOR operations and Blum-Blum-Shub (BBS) generators. Extensive testing confirmed the robustness of this approach against attacks. Kurtynykh, Ghita, and Shiaeles (2021) [14] addressed the complexities of Hashicorp Vault is particularly suitable for small businesses due to its superior security features. A summary of the related studies is provided in Table 1. for further reference.

Table 1: Previous works on key generation

Metrics Used	Method Limitations	Dynamic Key Generation and Non-Transferability	Key Space	Accuracy	Technique(s) Used	Researcher(s)
Confusion matrix, Precision, Recall, F1-Score, ROC Curve	Reliance on specific datasets, weak attack analysis	Stable, Transferable, Permanent	2^{1024}	98%	Key generation method from fingerprint image based on deep Convolutional neural network model	Hashem, M., & Kuban, K.H (2023)
Keyspace analysis, Key sensitivity, Information entropy, Histogram, Correlation, Differential attack, Noisy attack, Cropping attack	Chaos effects may be unreliable in other scenarios	Stable, Transferable, Permanent	2^{398}	Not specified	An Image Encryption Scheme Based on Chaotic Key Generation using Deep CNN	Erkate, U., Toktas, A., Enginoglu, S., et al (2024)
Resistance to brute-force attacks	Performance declines with large files and lacks adaptability and attack analysis	Stable, Transferable, Permanent	2^{2527}	Outperforms traditional algorithms with small text files	A Deep Learning Approach for Symmetric-Key Cryptography System	Quinga-Socasi, F., Zhinin-Vera, L., & Chang, (2020)
Misrecognition rate, Generation intensity, Accuracy	Weak analysis of pattern diversity and resistance	Stable, Transferable, Permanent.	2^{1024}	98%	Fingerprint bio-key generation based on deep neural networks	Wu, Z., Lv, Z., Kang, J., et al (2022)
Encryption effectiveness and data security in cloud environments	Complexity in implementation, lack of attack evaluation	Stable, Transferable, Permanent.	Not specified	Improved time efficiency and performance	Hybrid Encryption using ECCDH with ANN for key exchange in cloud computing environments	Alesawiy, O., & Muniyandi, R.C (2016)
Keyspace analysis, Binary entropy analysis, Resistance to brute-force attacks	The system is complex, and its reliance on limited technologies may affect performance	Stable, Transferable, Permanent.	2^{512}	Final training loss: 0.0974, Validation loss: 0.0953	Enhancing Data Security through Machine Learning-based Key Generation and Encryption	Saini, A., & Sehrawat, R (2024)
Performance metrics, Key management systems, Implementation complexity, Vulnerabilities	The variability of systems and the limited evaluation	Stable, Transferable, Permanent.	Not specified	Not specified	Comparative Analysis of Cryptographic Key Management Systems	Kurtiniyih, I., Ghita, B., & Shiaeles, S (2021)
Confusion matrix, Precision, Recall, F1-Score, Entropy Repetition, Chi-Squared, Pearson Correlation, Stability, Range, Autocorrelation Test False Positive Rate (FPR)	The limitation of the method lies in the data type.	Dynamic, Non-Transferable, Ephemeral	2^{8192}	99.73 - 99.83	Biometric-Based Secure Encryption Key Generation Using Convolutional Neural Networks and Particle Swarm Optimization	Proposed Method (2025)

This research builds upon the foundations laid by these studies, emphasizing the dynamic generation of encryption keys using deep learning and chaotic systems to address challenges in key management and enhance security. The comparison Table 1. clearly demonstrates the superiority of our proposed method over all previous approaches. The proposed method utilizes dynamic keys generated by deep learning networks, which significantly enhance randomness and security. Moreover, the key is non-portable, non-persistent, and achieves the largest size and highest accuracy compared to other methods.

3 Background

This paragraph addresses two main techniques: CNNs and PSO, which form the foundation of the methodology proposed in this research. In the following paragraphs, we will provide a summary of each technique and explain its significance in the study.

A. CNNs are advanced models in the field of deep learning, specifically designed to handle grid-like data, such as images. In this research, two CNN models were used to generate an encryption key based on fingerprint

images. Table 2. summarize the components of each model used in the work.

Table 2: Components of CNN models used

Layer (Type)	Output Shape	Parameters (#)
Conv2D (conv2d_1)	(None, 92, 92, 32)	832
BatchNormalization(batch_normalization_1)	(None, 92, 92, 32)	128
MaxPooling2D(max_pooling2d_1)	(None, 46, 46, 32)	0
Conv2D (conv2d_2)	(None, 42, 42, 64)	51,264
BatchNormalization batch_normalization_2)	(None, 42, 42, 64)	256
MaxPooling2D (max_pooling2d_2)	(None, 21, 21, 64)	0
Conv2D (conv2d_3)	(None, 19, 19, 128)	73,856
BatchNormalization (batch_normalization_3)	(None, 19, 19, 128)	512
MaxPooling2D (max_pooling2d_3)	(None, 9, 9, 128)	0
Dropout (dropout_1)	(None, 9, 9, 128)	0
Flatten (flatten_1)	(None, 10368)	0
Dense (dense_1)	(None, 1024)	10,617,856
Dropout (dropout_2)	(None, 1024)	0
Dense (dense_2)	(None, 600)	615,000

The first model was designed to identify a person's identity based on their ID number. After confirming the person's identity, the second model identifies the selected fingerprint and extracts its features. Both models rely on convolutional layers to automatically and progressively extract important features from the input data, making them effective in performing tasks, which, in turn, aids in generating strong encryption keys by analyzing fine patterns in the images.

The two models were trained using the backpropagation technique with a suitable loss function for each task. This architectural design was chosen to achieve accurate performance in recognizing the identity of the fingerprint owner through the identifier number in the file name, and then generate an encryption key based on the unique features of the fingerprint using two convolutional neural networks.

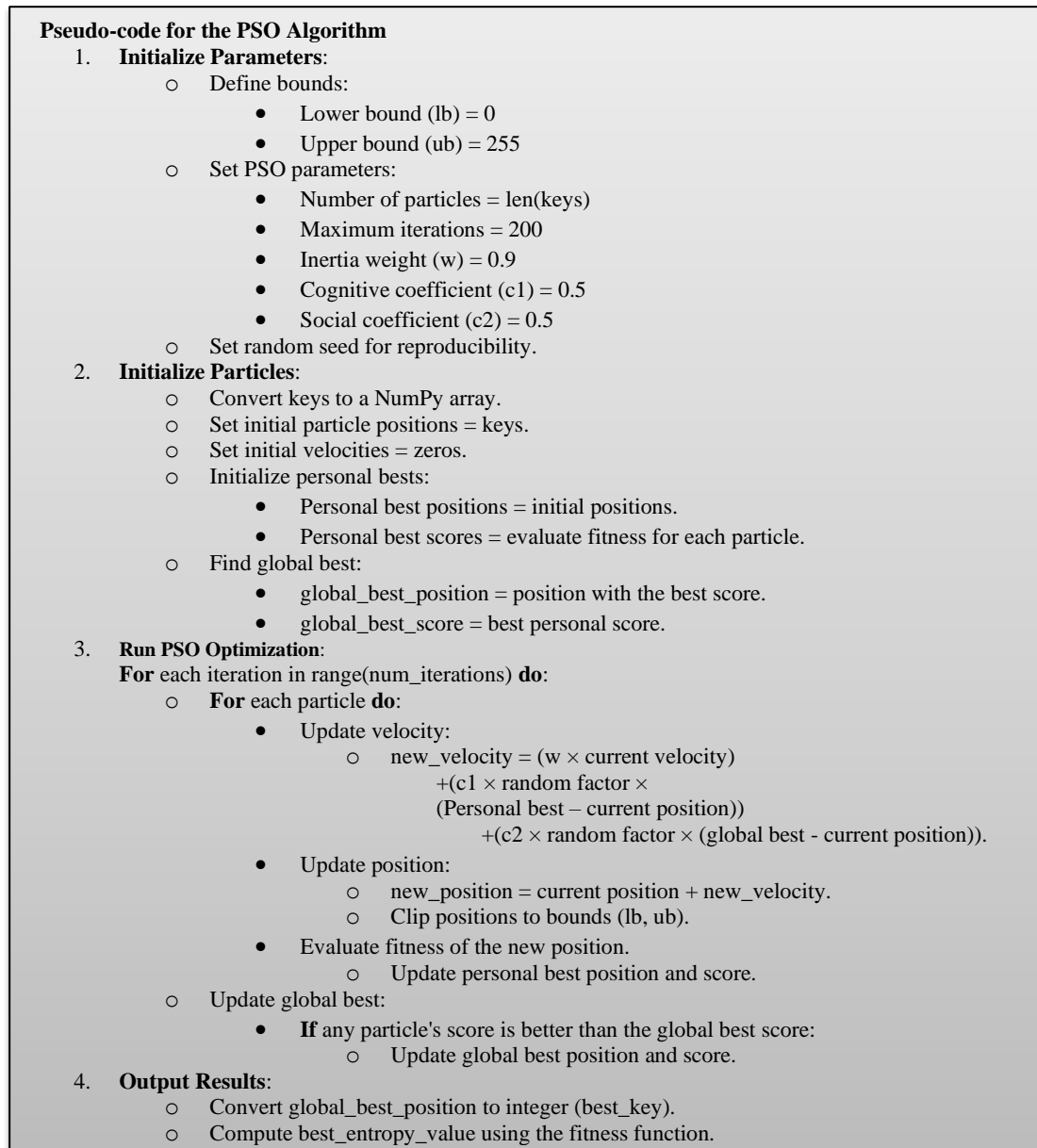


Figure 1: PSO algorithm

This approach adds an extra layer of security, ensuring that the keys are not only unique and non-repetitive but also resilient to various forms of attacks. The use of PSO ensures that the final encryption keys are both optimized for security and

B. PSO (Particle swarm optimization) is an optimization algorithm inspired by the collective behavior of birds or fish. It involves a group of particles, each representing a potential solution in the solution space. Each particle adjusts its movement based on its own experience and the experiences of neighboring particles, with the aim of reaching the optimal solution. PSO is known for its efficiency and ability to find optimal solutions in multi-dimensional spaces. In this research, PSO is applied to optimize the process of encryption key generation.

The algorithm enhances the randomness and strength of the generated keys, ensuring that they are both secure and resistant to attacks. PSO improves the key generation process by fine-tuning the key parameters in real-time, making it more robust against potential security threats. This approach adds an extra layer of security, ensuring that the keys are not only unique and non-repetitive but also resilient to various forms of attacks. The use of PSO ensures that the final encryption keys are both optimized for security and generated dynamically, without the need for permanent storage, thus reducing the risks of key leakage or unauthorized access. **Key enhancement using PSO:** The Particle swarm optimization (PSO) algorithm is used to enhance the quality of the initial key, making it stronger and more secure. Figure 1. illustrates the detailed steps of the Particle swarm optimization (PSO) algorithm using pseudo-code. This pseudocode reflects the essence of the PSO algorithm applied to optimize encryption keys

based on the fitness function (such as randomness or security). The process iteratively adjusts the position (key) and velocity of the particles to find the optimal encryption key with high security.

4 Secure key management

Secure key management is a critical process to ensure the protection of encrypted data across encryption systems. In the proposed methodology, the focus is on generating cryptographic keys in real-time without permanent storage, thus reducing the risks associated with key leakage. However, temporary handling and protection of keys during their lifecycle remain essential. Below is a detailed explanation of the steps and importance of secure key management, updated to reflect the real-time generation approach:

1. Key generation: In the proposed system, keys are generated dynamically and in real-time using advanced techniques such as artificial neural networks, particularly convolutional neural networks (CNN). This approach ensures that the keys are both highly secure and non-repetitive, avoiding the need for long-term storage. These keys are designed to be sufficiently random and robust, minimizing the possibility of guessing or tampering.

2. Temporary key handling: While keys are not stored permanently, they are managed securely during their temporary existence within the system. During encryption or decryption processes, the keys are stored in memory with strict safeguards, such as memory encryption or secure enclaves, to prevent unauthorized access. Once the operation is complete, the keys are securely erased from the system to eliminate any residual risk.

3. Key distribution: Since the system eliminates the need for traditional key exchange, the reliance on secure protocols like SSL/TLS or Diffie-Hellman for key distribution is significantly reduced. Instead, the generated key remains local to the system, mitigating risks associated with interception during transmission [16].

4. Key rotation: In systems where keys are reused for multiple sessions or extended periods, regular key rotation is critical. However, in the proposed system, each key is uniquely generated for a specific session or operation, inherently providing the benefits of key rotation by design.

5. Key revocation: Although the system minimizes the use of persistent keys, mechanisms for immediate key invalidation are essential for scenarios involving session-based or temporarily stored keys. These mechanisms ensure that any exposed or misused keys are rendered unusable promptly [17].

6. Importance of key management in real-time systems: The proposed approach emphasizes the importance of secure key handling during the active

lifecycle of keys. By avoiding permanent storage and focusing on real-time generation and temporary protection, the system significantly reduces the risks associated with key leakage or unauthorized access. This approach aligns with best practices in modern cybersecurity by combining the advantages of real-time key generation with robust temporary key management to ensure the highest level of data protection throughout the encryption process [16].

5 Proposed method

Figure 2: presents the diagram for the proposed encryption key management and generation.

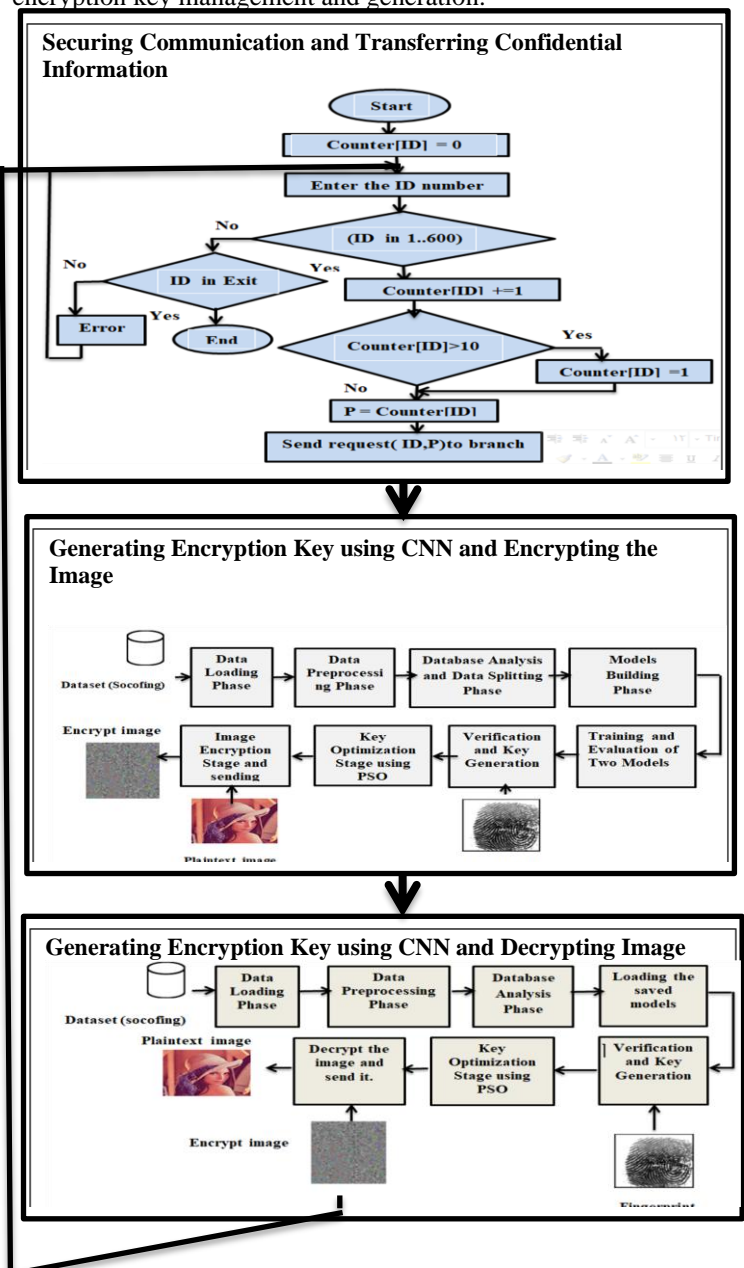


Figure 2: Proposed method diagram

proposed method consists of three main parts. The first part begins with an algorithm for securing communication and managing encryption keys. This is followed by the second part, which involves the process of generating the encryption key and encrypting the image. Finally, the third part focuses on decrypting the image after the key has been generated. Each of these parts will be explained in detail later.

Part One: Securing communication and managing confidential information transfer

The first part of Figure 2 illustrates an algorithm designed to ensure secure communication and reliable key management between branches and the main branch. When a branch requests access to sensitive information (such as encrypted images), the main branch fulfills this request by sending the requested information after encrypting it with a secure key, ensuring data protection during transmission. User ID is used to control access.

Algorithm execution steps

The algorithm is executed in cooperation with the following two parts in the diagram as follows:

- 1. Starting the process (start):** The process begins by initializing the user's counter Counter [ID] to zero.
- 2. Entering the ID number:** The system prompts the user to input their identification number to verify their identity.
- 3. Verifying the ID range (ID in 1..600):** The system checks whether the entered ID number falls within the allowed range (1 to 600).

- If the number is outside the range, an error message is displayed, and the user is asked to re-enter the ID.
- If the number is valid, the process moves to the next step.

- 4. Checking the match with the exit indicator (ID in exit):** The system compares the entered ID with the exit indicator list.

- If a match is found, the process is terminated.
- If no match is found, the process continues to the next step.

- 5. Incrementing the message counter (counter [ID] += 1):**

If the ID is valid and not listed in the exit indicator, the user's message counter is incremented by 1.

- 6. Managing the number of sent messages (dynamic key management):** The system checks whether the number of messages sent by the user has exceeded the allowed limit (10 messages).

- If the limit is exceeded, the counter is reset to 1.
- If the limit is not exceeded, the current counter is used as an index for generating the encryption key. This mechanism ensures unique encryption keys for each set of messages, enhancing data security. Additionally, it raises a critical question: "Can biometric fingerprint data generate dynamic encryption keys resistant to quantum attacks?"

This approach aims to strengthen the security of biometric keys against advanced threats such as quantum attacks.

- 7. Sending the request to the branch (send request to branch):** The request containing the ID (ID) and the fingerprint index (P) is sent to the second branch for processing.

- **In the second part:** A key is generated for image encryption, and the encryption process is executed. After encryption, the encrypted image is sent back to the first part.
- **In the third part:** A new key is generated to decrypt the image. Once decryption is completed, the data is returned to the first part for the remaining steps.

Note: The details of the second and third parts will be explained in the following sections of the document for a precise and comprehensive understanding. In this way, the three parts form an integrated system that ensures secure communication and the safe transmission of sensitive information effectively.

Algorithm features

- **Biometric security:** Fingerprints are used as a means to verify user identities, which reduces the risks of unauthorized access.
- **Synchronization:** The system relies on concurrent processing, enhancing performance efficiency and reducing response times for requests.
- **Dynamic key management:** Each key is generated uniquely for each user based on their fingerprint, increasing the difficulty of breaching the system. This algorithm ensures effective protection of encrypted data and enhances the security of communications between branches, making it an excellent choice for systems that require a high level of security and privacy.

Part Two: Encryption key generation using CNN and image encryption

The encryption key is generated using CNN based on the fingerprint. This process is carried out as specified in Part 2 of the diagram, which includes the following operations:

- 1. Database loading phase:** This step is considered one of the main preparatory phases in the system to ensure the readiness of the data and models required to achieve accuracy and security in encryption key generation. In this research, the SOCOFing database was used, which contains fingerprints from 600 people of African descent, with each person having 10 fingerprints, resulting in a total of 6,000 original fingerprints. Additionally, synthetic groups were created with three levels of variation in the fingerprints: minor changes (Easy), medium changes (Medium), and significant changes (Hard). The total number of synthetic fingerprints used in training was approximately 49,270. The variation fingerprints were used for training the model, while the original fingerprints were used solely for testing.
- 2. Data preprocessing phase:** The following processes are included:
- 3. Image size standardization:** To ensure that all images in the database are compatible with the model requirements, the dimensions of all images are standardized. A common size, such as 96×96 pixels, is often chosen to prepare the images for efficient model processing. The formula for resizing the images can be expressed mathematically as shown in Equation (1) below:

$$I'(x',y') = I(y/Sy, x/Sx) \quad (1)$$

Where:

$I(x,y)$ is the original image, and $I'(x',y')$ is the image after resizing, with Sx and Sy representing the scaling factors in the image dimensions[18].

A. Image enhancement using histogram equalization:

The histogram equalization technique was applied to enhance contrast in fingerprint images and highlight fine details. This technique is one of the fundamental methods in image processing and quality enhancement, aiming to improve the distribution of grayscale levels in the image to make fine details more visible. In images with low contrast, gray values may cluster within a narrow range, leading to the loss of fine details in dark or bright areas. Histogram equalization is used to address this issue by improving the distribution of these gray values over a broader range of available colors, enhancing contrast and making details easier to detect. The process of adjusting the tonal gradients in the image is carried out using the following equation (2)[19]:

$$H'(I) = \frac{CDF_{min} - CDF(I)}{(NXM) - CDF_{min}}(1-L) \quad (2)$$

The histogram equalization process involves several key parameters that affect the final outcome of the operation:

1. Cumulative distribution function (CDF): This is the primary factor that determines how grayscale values are redistributed in the image. The CDF accumulates grayscale values progressively from the lowest to the highest and is used to adjust the distribution. Through this function, the grayscale value distribution in the image is calculated, and adjustments are made to spread these values evenly across the color range.

2. Minimum non-zero value (CDFmin): This refers to the smallest non-zero value in the cumulative distribution function. It is used to determine how grayscale values in the image will be adjusted to achieve a more balanced distribution. For example, if the grayscale values in the image are concentrated around a particular value, utilizing this minimum helps improve the distribution of those values without significantly affecting the overall contrast of the image.

3. Image size (N×M): This refers to the number of pixels in the image. The larger the image (i.e., a greater N×M), the more opportunities there are for accurately redistributing grayscale values. However, it is important to note that image size can impact processing speed, as larger images require more computations.

4. Number of gray levels (L): Typically, $L=256$ in grayscale images (meaning there are 256 possible tonal levels ranging from 0 to 255). The number of gray levels defines the range of colors that can be distributed across the image. In images with a high number of gray levels, tonal gradations can be distributed more evenly, leading to better contrast enhancement.

When applying this technique, the range of grayscale values in the image is expanded, and these values are evenly distributed across the color range, leading to increased contrast. This enhanced contrast reveals fine details in the image, such as the minutiae in fingerprints, which might be poorly visible in low-contrast images. In the case of fingerprints, fine details such as ridges and patterns are often crucial for analysis and classification.

By using histogram equalization, the clarity of these fine details can be improved, aiding in better feature extraction of the fingerprint and achieving higher performance in systems that use fingerprint recognition. Figure 3. shows an example of fingerprints before and after contrast enhancement using histogram equalization. Notice how the enhanced images display finer and clearer details compared to the original images.

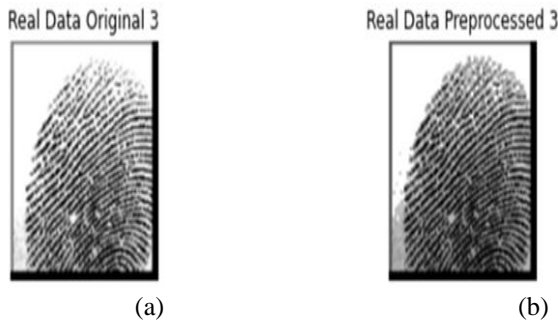


Figure 3: (a) Original image (b) Image after histogram equalization

5. Database analysis phase

After image enhancement, each fingerprint is analyzed to identify distinctive features, such as patterns and key regions within the fingerprint. This helps prepare the data for the model to understand the unique elements in each fingerprint. The goal of this process is to efficiently analyze the fingerprint database to extract the necessary information for feeding two different models. This is achieved by parsing the file name to extract the individual's identity, finger type, and hand (right or left). Additionally, these steps prepare the data for the two models, allowing the first model to recognize the individual's identity, while the second model identifies the finger type based on fingerprint information. After applying these processes to the database, the data is divided into training and testing sets. Artificial fingerprint data is used for training, while original data is used for testing.

6. Model building phase: After preparing the database, two models are built using CNNs. The first model aims to identify the person's identity (SubjectID), while the second model aims to determine the finger number (FingerNum) and extract distinctive features of the finger. Each model consists of the layers shown in Table 2. The hyperparameters used in the models are illustrated in Table 3.

Table 3: CNN hyperparameters configuration

Hyperparameter	Value
Number of Convolutional Layers	3
Batch Size	64
Filter Size	3x3, 5x5
Filters per Layer	32-64-128
Activation Function	ReLU, Softmax, tanh
Optimization	Adam
Learning Rate	0.0001
Epochs	20
L2 Regularizer	regularizers.l2(0.001)
Loss Function	Categorical_crossentropy
Dropout	0.3, 0.4
MaxPooling	MaxPooling2D(pool_size=(2, 2))

7. Training and evaluation phase of the two models

The performance of both models is evaluated using standard performance metrics such as accuracy, validation, and error rate calculation. This ensures that the first model is capable of accurately verifying the identity of authorized individuals when requesting the encryption key. Similarly, the second model's performance is assessed to determine its ability to correctly identify the fingerprint belonging to the individual whose identity has been verified. This evaluation is done using the test set.

8. Identity verification and key generation

The identity of the individual and the fingerprint match with the registered name are verified using two deep learning models. This is a key step in generating the encryption key from fingerprints, as illustrated in Figure 4.

9. Key optimization stage using PSO

To enhance the quality of the initial key and obtain a stronger, more secure key, PSO algorithm is applied. This algorithm aims to improve the random distribution and security properties of the key. The goal of this algorithm is to increase the randomness of the key and ensure its difficulty in being guessed or broken. The use of the PSO algorithm to optimize encryption keys relies on updating the positions and velocities of particles based on the individual's fingerprints, as illustrated in Figure 1. This continuous update of the keys, leveraging the best personal and global positions, results in generating an encryption key that is more secure and complex. This also raises the question: "How does the proposed system perform against statistical attacks?" This approach aims to reduce the likelihood of the keys being exposed to any repetitive patterns that could be exploited in statistical attacks. Table 4. outlines the hyperparameters used in the optimization algorithm, selected based on a series of experimental trials.

Pseudo-Code for verification and encryption key generation

1. **Initialize finger name function:**
 - Define show_fingernum(fingernum):
 - If fingernum >= 5:
 - Set hand = "right" and subtract 5 from fingernum.
 - Otherwise:
 - Set hand = "left".
 - Map fingernum to finger names (e.g., little, ring, middle, index, thumb).
 - Return the full finger name (hand + finger).
2. **Verify fingerprint information:**
 - Predict subject ID and finger number for a random fingerprint (rand_fp_num) from the test set using models:
 - Id_pred = predicted subject ID.
 - Id_real = actual subject ID.
 - fingerNum_pred = predicted finger number.
 - fingerNum_real = actual finger number.
 - Check predictions:
 - If both IDs and finger numbers match:
 - Print "Information confirmed" with subject ID and call show_fingernum(fingerNum_pred) to get the finger name.
 - Otherwise:
 - Print "Prediction is wrong."
3. **Extract candidate fingerprints:**
 - Initialize lists keys1 (for original fingerprints) and keys2 (for dense layer outputs).
 - For each index i in the prediction range:
 - Get Id_check = predicted subject ID.
 - If Id_check == Id_pred:
 - Append the fingerprint to keys1.
 - Append the dense layer output to keys2.
 - Convert keys1 and keys2 to arrays.
4. **Select target fingerprint:**
 - Use index p1 to select:
 - original_fp = keys1[p1].
 - dense_output_finger_selected = keys2[p1].
5. **Apply data augmentation:**
 - Define an image data generator (datagen) with +transformations:
 - Rotation, width/height shift, shear, zoom, and horizontal flip.
 - Reshape original_fp to fit the generator's input format.
6. **Generate augmented fingerprints and keys:**
 - Use datagen to create 20 augmented fingerprints:
 - For each augmented fingerprint:
 - Generate a new fingerprint.
 - Predict the dense layer output.
 - Take absolute values of the output to create a key.
 - Append the key to the keys list.
7. **Return results:**
 - Output Keys // To be used as input for PSO algorithm to find optimal key the list keys for use in encryption.

Unit/Explanation	Description/Value	Parameter
Lower bound of the range	0	lb
Upper bound of the range	255	ub
Number of particles in the swarm	len(keys)	num_particles
Maximum number of iterations	200	num_iterations
Inertia weight	0.9	w
Cognitive coefficient (personal best focus)	0.5	c1
Social coefficient (global best focus)	0.5	c2
Random factor for cognitive component; varies per particle	0.5	r1
Random factor for social component; varies per particle	0.5	r2
Initial positions of the particles	keys	particle positions
Initial velocities of the particles	zeros	particle velocities
Personal best positions	Updated based on the best performance of each particle	personal_best_positions
Best scores achieved by each particle	Calculated through fitness evaluation	personal best scores
Global best position	Updated based on the global best performance	global best position
Global best performance	Updated based on the global best value	global best score

10. Image encryption stage and sending encrypted image

Chen's chaotic system is a three-dimensional dynamic system that exhibits chaotic behavior and is based on nonlinear differential equations to represent the evolution of the state over time. It can be used to generate a chaotic encryption key based on the system's state. The Chen chaotic system relies on the following equations that describe the changes in the variables x, y, and z:

$$\frac{dx}{dt} = a \cdot (x - y) \tag{3}$$

$$\frac{dy}{dt} = (a - c) \cdot x - x \cdot z + c \cdot y \tag{4}$$

$$\frac{dz}{dt} = x \cdot y - b \cdot z \tag{5}$$

Figure 4: Pseudocode for the identity verification and key generation process

Where: x , y , and z are the variables that determine the state of the chaotic system at time t .

- a , b , and c are the parameters that control the behavior of t the system. Steps followed:
 - **Initial conditions:** The process starts by defining the initial values for x , y , and z , which represent the state of the system at the beginning of the simulation. These values are set in the code as [1.0,1.0,1.0].
 - **Numerical integration:** The odeint function is used for numerical integration to solve the differential equations over time. Through this process, the values of x , y , and z are updated at each time step, based on the parameters a , b , and c that influence the system's behavior.
 - **Generating a chaotic sequence:** A chaotic sequence is generated by solving the differential equations of the Chen chaotic system over multiple time steps. This sequence is then used to generate a chaotic encryption key.
 - **Encryption key generation:** The resulting chaotic sequence is converted into integer values ranging from 0 to 255 to represent color values in an RGB image. This is done by multiplying each value in the sequence by 255 and converting it to the uint8 data type.
 - **Combining the chaotic key with the generated key:** The chaotic key is combined with the key generated using CNN through an XOR operation. This step increases the complexity of the final key used for image encryption.
 - **Encrypting the image:** The XOR operation is applied between the original image and the final key to generate the encrypted image. This operation transforms the pixel values in the image into new values based on the chaotic key.

Part three: key generation using CNN and image decryption:

This part is similar to the stages in Part 2, with the only difference being that the models are not built and trained again; instead, the previously saved models are loaded. Additionally, there is a decryption stage instead of encryption. The stages in this part are as follows:

- **Data loading phase:** Only the test set is loaded (i.e., 600 genuine fingerprints from SOCOFing).
- **Data preprocessing phase:** The raw data is processed to prepare it for the next stage.
- **Database analysis phase:** The data is analyzed to extract the necessary information.
- **Loading the saved models:** The previously trained models are loaded.
- **Verification and key generation:** The data is verified, and the key is generated.

- **Key optimization stage using PSO:** The key is optimized using the PSO algorithm.
- **Decrypt the image and send it:** The image is decrypted and sent. Decryption: To decrypt the image, the same key (the chaotic key and the key generated from CNN) is used to perform an XOR operation on the encrypted image, restoring the original image.

6 Results and analysis

This section of the research addressed four main axes: evaluating the CNN, assessing the generated key, evaluating the PSO algorithm, and finally comparing the results of the proposed method with similar methods. As follows:

A. Results of CNN models and performance analysis

At this stage, the data was divided into training and testing sets to ensure the accuracy of the models in predicting and distinguishing between different categories. 80% of the data was allocated for training, and 20% for testing, ensuring an equal distribution of categories in both sets to avoid bias. After training the models on the training set, their performance was tested using the test data.

The accuracy of the models was calculated using the accuracy function available in the TensorFlow library, which represents the ratio of correct predictions to the total number of predictions. To continuously monitor performance, TensorBoard was used, which helped track various metrics such as accuracy and loss throughout the training and testing phases, allowing for ongoing improvements to the models based on these indicators. The results obtained showed varying performance across the different models, with these results summarized in Tables 5, 6, and 7, which illustrate accuracy and loss across different generations. Additionally, the graphs in Figures 5, 6, and 7 show the evolution of the models' performance over time, highlighting the models' ability to learn and improve progressively. **Where the false positive rate was 0.000185.**

Table 5: Model performance comparison: accuracy and loss.



ID Name Fingerprint example	Model No	Training Accuracy(%)	Testing Accuracy (%)	Training Loss	Testing Loss
331 right little 	1	99.733334	0.9943	0.0563811	0.0947
331 right little 	2	99.833333	0.9856	0.1004451	0.1387

Table 6: Classification report for finger recognition

No_ SubjectID	Precision	recall	f1-score	support
0	1.00	1.00	1.00	10
1	1.00	1.00	1.00	10
2	1.00	1.00	1.00	10
3	1.00	1.00	1.00	10
4	1.00	1.00	1.00	10
5	1.00	1.00	1.00	10
.
597	1.00	1.00	1.00	10
598	1.00	1.00	1.00	10
599	1.00	1.00	1.00	10
accuracy			1.00	6000
macro avg	1.00	1.00	1.00	6000
weighted avg	1.00	1.00	1.00	6000

Table 7: Classification report for subjectID recognition

No_ Finger	Precision	recall	f1-score	Support
0	1.00	0.99	1.00	600
1	1.00	1.00	1.00	600
2	1.00	1.00	1.00	600
3	1.00	1.00	1.00	600
4	1.00	1.00	1.00	600
5	1.00	1.00	1.00	600
6	1.00	1.00	1.00	600
7	1.00	1.00	1.00	600
8	1.00	1.00	1.00	600
9	1.00	1.00	1.00	600
accuracy			1.00	6000
macro avg	1.00	1.00	1.00	6000
weighted avg	1.00	1.00	1.00	6000

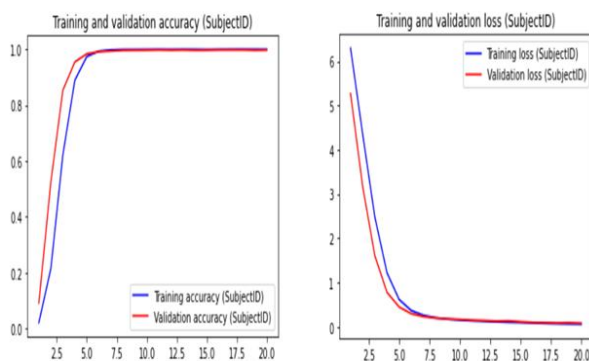


Figure 5: Accuracy and loss of the identity model.

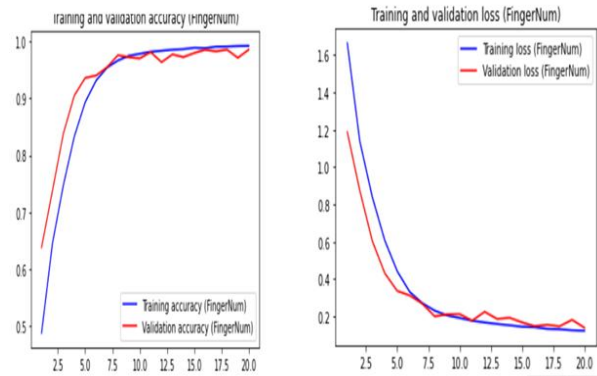


Figure 6: Accuracy and loss of the fingerprint model.

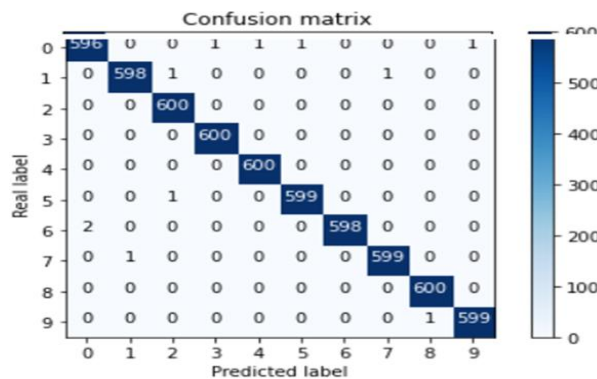


Figure 7: Confusion matrix and accuracy metric

B. Encryption key evaluation results and metrics: The generated encryption key was evaluated using a set of specialized metrics to ensure its quality and effectiveness in resisting cyberattacks. The experiments were conducted using fingerprint images sized 96×96 in a Kaggle environment with Python, on a workstation equipped with an Xeon(R) I (R) processor, 64 GB of RAM, and a GPU P100. The metrics used included evaluations such as key size and various randomization tests (such as entropy test, repetition test, etc.) to assess the randomness of the key and its predictability. These tests help ensure that the system remains unaffected when used in live applications.

6.1 Key space analysis

Brute-force attack is a type of cyber-attack that relies on guessing the key by attempting a large number of possible passwords or secret phrases. An encrypted image with a short key is highly vulnerable to this attack over time. However, if the key is longer, it will remain resistant for a longer period. Therefore, it becomes impossible to guess the key if it has an adequate length. Key space analysis is used to assess the strength against brute-force attacks.

According to this analysis, a key with a length greater than 2^{100} is considered suitable for high-security encryption [26]. In our system, we propose an approach based on deep neural networks (CNN) and PSO to generate this key.

The key has a size of 1024 values, with each value ranging between 0 and 255. This means the key consists of 1024 bytes (since each value requires one byte, and 8 bits are enough to represent values from 0 to 255). Given that each value in the key ranges from 0 to 255, we have 256 possibilities for each value. With 1024 values, the total key space will be: 256^{1024} or, in other words, $2^{8 \times 1024} = 2^{8192}$. This represents an extremely large key space, which is sufficiently large to be highly resistant to brute-force attacks. A key size of 2^{8192} offers a very high level of security, making it practically impossible to crack using brute-force methods, even with fast computing devices. Nonetheless, the question remains: **How does the proposed system perform against brute-force attacks?**

Comparison of key space (2^{8192}) with traditional systems

- Comparison with AES-256:**

The proposed key space (2^{8192}) is significantly larger compared to AES-256, where the key space is approximately 2^{256} . This substantial difference makes our key space more resistant to brute-force attacks. Traditional systems like AES-256 rely on efficient algorithms to compensate for the smaller key space compared to the vast proposed space.

- Comparison with RSA-2048:**

The proposed key space is also significantly larger compared to RSA-2048, where the key space is approximately 2^{2048} . RSA relies on computational complexity for large numerical factorization, whereas in our system, the security strength depends on the key length derived from biometric features processed through deep networks.

- Comparison with ECC-384 (Elliptic curve cryptography):**

The traditional key space for ECC-384 is approximately 2^{384} , which is much smaller compared to our proposed key space (2^{8192}). ECC relies on elliptic curves to compensate for shorter keys, but in contrast, we provide much longer keys derived from neural networks, enhancing their unpredictability.

- Comparison with DES (Data encryption standard):**

The key space in DES is 2^{56} , which is extremely small compared to our proposed key space. DES is considered outdated and vulnerable to brute-force attacks, whereas our proposed key space vastly surpasses it in terms of length and complexity.

6.2 Significance of results in cryptographic key management

- The results, such as randomness tests and high entropy, demonstrate that the generated key exhibits a high degree of randomness, making it ideal for high-security applications.
- High entropy indicates that the keys have a uniform distribution of values, reducing
- the likelihood of predicting any part of the key, which is a critical feature in key management.

6.3 Encryption key tests

In this study, six fingerprint samples were used as the basis to generate six encryption keys. Each key underwent comprehensive testing using eight different metrics to determine the quality and randomness of the generated keys. The results of these eight tests were systematically presented in a table, reflecting the effectiveness of the proposed method. The results showed the success of the keys in all eight tests, confirming that the keys generated from the fingerprints meet the required security standards. These tests demonstrate the randomness and unpredictability of the keys, making the approach suitable for secure encryption applications. The core encryption key tests include the following:







- Entropy test:**

The entropy measures the distribution of information in the key and reflects the level of randomness. The entropy is calculated using the following equation (6):

$$H(X) = -\sum_{i=1}^n p(x_i) \log_2 p(x_i) \quad (6)$$

Where $p(x_i)$ is the probability distribution of the value x_i in the key. If the entropy equals 8 bits, it means the key is completely random [26].

Table 8: Results of the entropy test

ID	5	3	1	173	174	175	
Finger Print							Ideal Value
Entropy Test	7.782	7.763	7.748	7.778	7.774	7.87	8

- Repetition test:** The repetition test generally aims to ensure that the key does not contain any repeated sections within its sequence, whether these sections are adjacent or non-adjacent. If parts of the key are repeated, it weakens the randomness and increases the likelihood of discovering a pattern that can be exploited in an attack. This test involves checking all parts of the key to detect any repetition that might impact its security level. The repetition test addresses repetition in the key overall, whether in adjacent or non-adjacent parts [27].

Table 9: Results of the repetition test

ID	5	3	1	173	174	175	
Finger Print							Ideal Value
Repetition Test	0	0	0	0	0	0	0





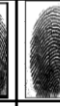
• **Uniformity test using the chi-squared test:**

This test aims to check whether the values in the encryption key are evenly distributed across the full range of possible values. The **chi-squared test** is used to compare the actual distribution of values in the key with the expected ideal distribution. If the values are evenly distributed, the key is considered to have a uniform distribution. Equation (7) illustrates the test:

$$\chi^2 = \sum_{i=0}^{255} \left(\frac{\frac{n}{256} - ni}{n/256} \right)^2 \quad (7)$$




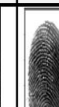
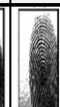

- **ni:** The frequency of occurrence of value i in the key.
- **n/256:** The expected frequency for each value i assuming a uniform distribution.
- **n:** The total number of values in the key.
- If the chi-square (χ^2) value is low, it indicates that the actual distribution of values is close to the ideal distribution, meaning the key is evenly distributed.
- At a significance level of 0.05, if the chi-square value is less than 293.25, the key is considered to have passed the test and has a uniform distribution [28].

Table 10: Results of the uniformity test

ID	5	3	1	173	174	175	
Finger Print							Ideal Value
Uniformity Test (Chi-Squared)	262.0	215.0	271.5	295.5	226.5	327.0	293

- **Repetition test(adjacent)** focuses specifically on identifying repetition in adjacent parts of the key. This test checks for any repeated consecutive or sequential sections that might indicate a fixed pattern or excessive repetition, which could weaken the effectiveness of encryption. Repetition of adjacent parts is considered a sign of poor randomness, thus reducing the strength of the key. The closer the value is to 0, the less repetition there is, which means the key has a higher level of randomness [29].

Table 11: Results of the repetition test(adjacent)

ID	5	3	1	173	174	175	
Finger Print							Ideal Value
Repetition Test(Adjacent)	0.59	0.2932	0.6842	0.293	0.586	0.977	0







- **Pearson correlation test** is a statistical test used to measure the relationship between two variables. This relationship is expressed by a coefficient called the "Pearson correlation coefficient," which ranges from -1 to 1. If the correlation coefficient is close to 0, it indicates no correlation (high randomness), making the encryption key strong and hard to predict. The purpose is to determine the extent of the correlation between values in the encryption key. If the correlation coefficient is close to 0, it indicates that the key is sufficiently random, thus making it strong against analytical attacks. The equation (8) represents the Pearson correlation equation:

$$r = \frac{\sum(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum(X_i - \bar{X})^2 \sum(Y_i - \bar{Y})^2}} \quad (8)$$

Where:

- **r:** Pearson correlation coefficient.
- **Xi:** Individual values in the first series.
- **Yi:** Individual values in the second series (e.g., lagged values in time series).
- \bar{X} : Mean of the Xi values.
- \bar{Y} : Mean of the Yi values [30].

Table 12: Results of the pearson correlation test

ID	5	3	1	173	174	175	
Finger Print							Ideal Value
Average Pearson Correlation	0.0062	0.0062	0.0062	0.0062	0.0062	0.0062	0.05







• **Stability test**

The key must remain stable if the input data is stable. This means that if the same inputs are used to generate the key multiple times, the resulting key should always be identical. However, slight changes in the inputs should result in a significant change in the key, which enhances encryption strength against attacks.

6.3 Consistency of fixed inputs

- If the input is fixed I, the encryption system should produce the same key K every time: $F(I)=K$.
- Repeat key generation multiple times using the same I, and the result should be consistent K in all attempts: $K1=K2=\dots=Kn$.

Table 13: Results of the stability test (1)

The key generated from attempt 5	The key generated from attempt 4	The key generated from attempt 3	The key generated from attempt 2	The key generated from attempt 1	Finger print	ID
[95 39 243 220 42...]	[95 39 243 220 42...]	[95 39 243 220 42...]	[95 39 243 220 42...]	[95 39 243 220 42...]		5
[143 24 107 86 110...]	[143 24 107 86 110...]	[143 24 107 86 110...]	[143 24 107 86 110...]	[143 24 107 86 110...]		3
[127 6 38 182 127...]	[127 6 38 182 127...]	[127 6 38 182 127...]	[127 6 38 182 127...]	[127 6 38 182 127...]		1
[188 224 208 191 161...]	[188 224 208 191 161...]	[188 224 208 191 161...]	[188 224 208 191 161...]	[188 224 208 191 161...]		173
[77 115 56 243 251...]	[77 115 56 243 251...]	[77 115 56 243 251...]	[77 115 56 243 251...]	[77 115 56 243 251...]		174
[106 30 245 143 52...]	[106 30 245 143 52...]	[106 30 245 143 52...]	[106 30 245 143 52...]	[106 30 245 143 52...]		175

6.4 Sensitivity to minor changes (inclusivity effect)

We make a slight change in the input I to create I'. A new key K' is generated using I': $F(I') = K'$. We measure the difference between K and K' using the Bit Change Rate:

$$\text{Bit change rate} = \frac{\text{Bit difference between } K \text{ and } K'}{1024} \times 100\%$$

The change rate should be higher than 50% to ensure the system's sensitivity to changes [31].

Table 14: Results of the stability test (1)







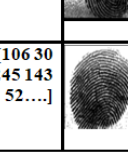
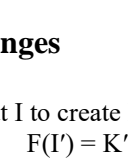
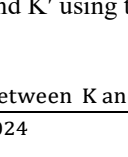
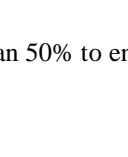
ID	5	3	1	173	174	175	
Finger print							Ideal Value
The key difference resulting from the input change	0.068	0.093	0.060	0.057	0.049	0.039	<0.1

Table 15: Encryption results with the original key and the modified key

Original image	Encryption using the original key	Encryption using the modified key	Difference between the encrypted images
			

6.5 Range test

The **range test** aims to evaluate the distribution of encryption key values within a specific range to ensure its randomness.







Steps of the range test

1. **Calculate the range:** Determine the difference between the maximum value max and the minimum value min. $\text{Range} = \text{max} - \text{min}$
2. **Range splitting:** divide the range into buckets.
3. **Frequency calculation:** count the values in each bucket.
4. **Distribution analysis:** if the frequencies are approximately equal, the key is considered random. expected frequency equation (9):

$$E_i = \frac{N}{M} \tag{9}$$

where N is the total number of values, and M is the number of buckets[27].

Table 16: Results of the range test

ID	5	3	1	173	174	175	
Finger Print							Ideal Value
Range Test	passed	passed	passed	passed	passed	passed	0.255

• **Autocorrelation test**

The Autocorrelation Test is used to determine the randomness of a sequence of values in an encryption key. If the key is sufficiently random, the autocorrelation values should be small or close to zero, indicating no clear pattern or dependency in the sequence. To calculate the autocorrelation at a lag d, equation (10) is used:

$$R(d) = \frac{1}{n-d} \sum_{i=1}^{n-d} (x_i \cdot x_{d+i}) \quad (10)$$

Where: R(d) is the autocorrelation coefficient for lag d.

- xi is the value at position i in the sequence.
- X_{d+i} is the value at position d+i in the sequence.
- n is the length of the sequence.

A value of R(d) close to zero for different values of d indicates a high level of randomness in the encryption key, The Figure 8. shows the distribution of autocorrelation test results, highlighting successful and failed values based on the specified critical value (0.05) [28].

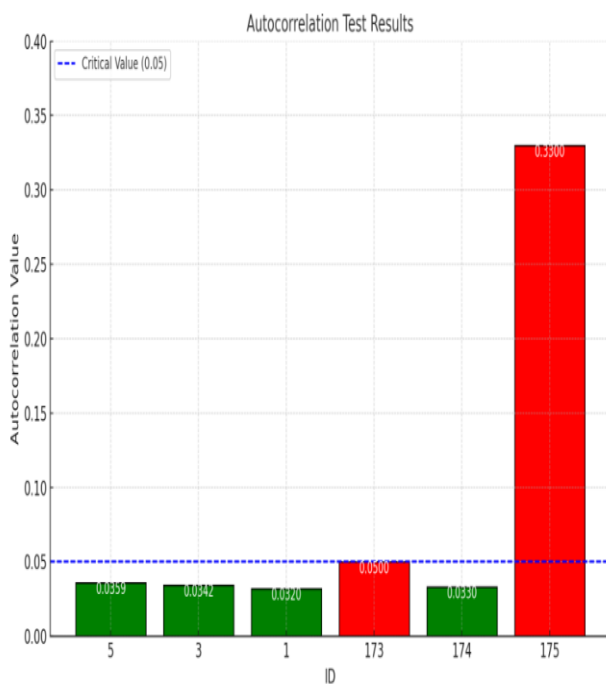








Figure 8: Distribution of autocorrelation test results with success and failure indication based on critical value







Table 17: Results of the autocorrelation test

Result Explanation	Critical Value	Ideal Value	Autocorrelation Test	Finger print	ID
Pass (close to 0)	0.05	Close to 0	0.0359		5
Pass(close to 0)	0.05	Close to 0	0.0342		3
Pass(close to 0)	0.05	Close to 0	0.0320		1
Fail (pattern detected)	0.05	Close to 0	Failed		173
Pass (close to 0)	0.05	Close to 0	0.0330		174
Fail (deviation from ideal value)	0.05	Close to 0	0.3300		175

C. Results of using PSO in enhancing the encryption key

After completing the specified number of iterations, the best encryption key is obtained, which is the key that achieved the highest fitness during the optimization process. Table 18. illustrates the effect of using PSO on the generated encryption key.

Table 18: The impact of the PSO algorithm in improving the encryption key.

No(ID)	5	3	1	173	174	175
Finger print						
Key strength without PSO	4.682	4.804	3.975	5.093	4.700	4.843
Key strength with the use of PSO	7.782	7.707	7.687	7.762	7.774	7.703

D. Comparison of the accuracy of the proposed system with other systems

This section evaluates the accuracy of our proposed system in comparison to other systems reported in recent years, based on their respective sources. Experimental results from our proposed model demonstrated an accuracy exceeding 99%. Table 19. presents a detailed comparison between our system and other existing systems.

Table 19: Accuracy comparison between our system and recent approaches.

Epochs	Data Type	%ACC	Method Using	Reference
100	Fingerprint	99.98	CNN model (VGG-16 Tuned)	2023[8]
60	Fingerprint	98.89	DeepFKTNet model	2022[29]
55	Fingerprint	99.10	Modified–LetNet(tested on FVC2004-DB1)	2022[30]
50	Fingerprint	98.00	MCP-FP model	2022[11]
--	Natural image	95.30	B-DCNNs	2022[13]
30	Fingerprint	96.10	CNN(3 classes of FVC2004)	2019[31]
20	Fingerprint	98.70	Modified HOG descriptor	2018b[34]
30	Fingerprint	97.40	Statistics of D-SIFT descriptor	2018a[33]
20	Fingerprint	99.73-99.83	CNN and PSO	proposed system

The aim of this comparison is to evaluate the effectiveness of the proposed system in the context of recent advancements in deep learning technology, providing insight into how cybersecurity can be enhanced through the application of advanced techniques. The table also reflects the ongoing progress in fingerprint data processing, showing that modern systems achieve higher accuracy than traditional systems, supporting the idea that using deep learning can improve the effectiveness and security of encryption systems.

7 Discussion

In this section, we will discuss the results of the proposed system in comparison to modern methods presented in Table 19, focusing on accuracy, randomness tests (such as entropy), and robustness. Additionally, we will address potential trade-offs associated with using CNNs, such as computational overhead. Below is a detailed comparison of key results.

7.1 Comparison of results

Accuracy: The proposed system (using CNN and PSO) achieved an accuracy between **99.73% and 99.83%** within just 20 epochs, outperforming most models in the table. For example: The **enhanced VGG-16 model** achieved **99.98% accuracy** in 100 epochs, the highest in the table, but required five times more epochs than the proposed system. The **Modified-LeNet model** achieved **99.10% accuracy** in 55 epochs, which is lower than the proposed system. The **DeepFKTNet model** achieved **98.89% accuracy** in 60 epochs. Thus, the proposed system stands out as a strong option, delivering high accuracy in less training time, thanks to the combination of CNN and PSO, which enhances feature extraction and

generates robust keys.

Randomness tests (e.g., entropy): The use of PSO in the proposed system significantly contributed to enhancing randomness, which strengthens the generated keys. When comparing randomness tests (e.g., entropy) with other models, the proposed system showed remarkable superiority. The combination of CNN and PSO enabled the generation of keys with excellent randomness levels, providing a higher degree of security compared to traditional models. PSO helps optimize the quality of the keys by searching for the optimal combination of hidden parameters, making them more random and harder to break.

It is important to note that the model only retains the predictions generated during its operation, which are values devoid of any sensitive information. This enhances the system's security against various types of attacks, such as mixed replacement attacks, crossover attacks, and exhaustive search attacks. In such attacks, the attacker has no knowledge of the key generation mechanism or the supporting data, making the number of attempts required to crack the key increase proportionally with its length. For example, if the key length is 1024, the number of possible combinations would reach 2^{8192} . The proposed system focuses on enhancing data security by avoiding key storage, improving the randomness of key generation, and protecting sensitive information from various attacks, while ensuring high efficiency in user fingerprint recognition.

Robustness: Biometric key (encryption key for security):

The biometric key is generated based on the parameters learned during the training of the CNN model. During training, the model learns unique representations or features extracted from fingerprints. These representations are numerical weights that are not easily interpretable.

The parameters are converted into an encryption key that relies on the unique properties of each fingerprint, making the key:

- Unique and tamper-proof.
- More secure and resistant to duplication.

Role of PSO in key enhancement:

PSO improves the key by identifying the optimal values of the parameters used in key generation. This enhances randomness and independence among keys, making them more resistant to attacks.

7.2 Advantages of the proposed system

The proposed system combines CNN and PSO to achieve:

- High classification accuracy in less time.
- High-quality encryption keys with excellent levels of randomness and security.
- Strong protection of users' biometric data against exploitation or breaches.
- Improved biometric key performance using PSO to generate stronger and more random keys, increasing the system's resilience to cyber threats. Thus, the proposed system leverages the multiple features of CNN and PSO, making it more robust in addressing security challenges such as resistance to adversarial attacks. While other models primarily focus on classification and accuracy, the proposed system demonstrates additional strength in encryption applications.

Potential trade-offs:

Although the use of CNN in the proposed system results in a slight increase in computational overhead compared to simpler models like the modified LeNet, this does not pose a significant obstacle. The model is designed to operate efficiently on modern systems supported by Graphics Processing Units (GPUs), ensuring accelerated training and reduced execution time.




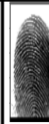


8 Conclusions

- The results of this research show that integrating biometric techniques with deep learning provides an innovative and effective solution for generating secure and robust encryption keys based on fingerprints. The proposed system enhances the security of data transmitted over the internet, making it more resistant to theft and tampering. The use of two convolutional neural network models is a significant step, where the first model contributes to identity recognition and the second focuses on fingerprint detail recognition, ensuring the extraction of unique and reliable biometric features.
- One of the main conclusions of this research is that the tanh activation function plays a crucial role in neural networks for generating encryption keys. This function is known for its ability to transform outputs into the range of $(-1, 1)$ non-linearly, which contributes to improving the quality of the generated keys. Increased complexity and randomness:

The tanh function ensures a more balanced distribution of values across the range $(-1, 1)$, reducing value concentration and enhancing the randomness of the key, leading to the generation of secure and robust encryption keys. Better stability during training: The tanh function helps avoid issues such as vanishing

gradients, resulting in better stability during the training process and improved model performance in generating encryption keys. Table 20. illustrates the key strength (entropy measure) when using the Tanh activation function compared to using the ReLU activation function.

Table 20: Compares the key strength (entropy) of the Tanh and ReLU activation functions.

No(ID)	5	3	1	173	174	175
Finger print						
Using Tanh Activation Function	7.782	7.763	7.748	7.748	7.732	7.719
Using ReLU Activation Function	3.648	4.439	4.597	5.328	4.844	4.541

The batch normalization layer plays a significant role in stabilizing and accelerating the learning process in deep models by normalizing the outputs to have a mean of 0 and a standard deviation of 1. While this stabilization is beneficial in many applications, such as image classification, it may negatively impact the strength of the generated encryption key.

- The results indicate that the generated keys exhibit high levels of randomness, making them more challenging to breach. Additionally, the use of PSO algorithm is considered an effective technique for enhancing the randomness of the keys, as it allows for generating different keys for each transmission, thereby reducing the risk of key theft and increasing security. A comprehensive analysis of the performance of the models used in this research was conducted, showing a significant improvement in encryption effectiveness and the reliability of the generated keys, underscoring the efficiency of these models in the context of cybersecurity.
- The proposed approach enhances system security compared to traditional systems by reducing reliance on static keys, which are a vulnerability in many encryption systems. Instead, biometric verification is used to generate unique keys for each user based on their fingerprints, thereby increasing the level of security. This research provides a significant contribution to systems that require high levels of protection, such as financial systems and medical data, by facilitating biometric verification for encryption without the need to exchange keys, thereby reducing associated risks. Additionally, the automatic key change feature adds an extra layer of security,

reflecting the effectiveness of this system in providing advanced protection. Ultimately, the research highlights the importance of integrating biometrics and deep learning in developing effective security solutions that address contemporary challenges in data protection.

- The method presented in the research has wide potential for application in various fields. In addition to securing fingerprints and using them to generate encryption keys, the method can be applied to secure internet of things (IoT) devices by generating strong encryption keys that protect communication between devices. It can also be used to secure data stored in the cloud by generating high-security encryption keys based on unique user attributes, such as fingerprints. These applications highlight the flexibility and efficiency of the method in addressing modern cybersecurity challenges and enhance its appeal in various practical scenarios.

References

- [1] Hosny, K. M., Darwish, M. M., & Fouda, M. M. (2021). Robust color images watermarking using new fractional-order exponent moments. *IEEE Access*, 9, 47425–47435. <https://doi.org/10.1109/ACCESS.2021.3069317>
- [2] Kuzior, A., Tiutiunyk, I., Zielińska, A., & Kelemen, R. (2024). Cybersecurity and cybercrime: Current trends and threats. *Journal of International Studies*, 17(2). <https://doi.org/10.14254/2071-8330.2024/17-2/5>
- [3] Saran, D. G., & Jain, K. (2023). An improvised algorithm for a dynamic key generation model. In *Inventive Computation and Information Technologies: Proceedings of ICICIT 2022* (pp. 607–627). Springer Nature Singapore. https://doi.org/10.1007/978-981-19-5048-5_44
- [4] Rahman, Z., Yi, X., Billah, M., Sumi, M., & Anwar, A. (2022). Enhancing AES using chaos and logistic map-based key generation technique for securing IoT-based smart home. *Electronics*, 11(7), 1083. <https://doi.org/10.3390/electronics11071083>
- [5] Kuznetsov, O., Zakharov, D., & Frontoni, E. (2024). Deep learning-based biometric cryptographic key generation with post-quantum security. *Multimedia Tools and Applications*, 83(19), 56909–56938. <https://doi.org/10.1007/s11042-023-15265-6>
- [6] Yang, W., Wang, S., Cui, H., Tang, Z., & Li, Y. (2023). A review of homomorphic encryption for privacy-preserving biometrics. *Sensors*, 23(7), 3566. <https://doi.org/10.3390/s23073566>
- [7] Rana, M., Mamun, Q., & Islam, R. (2023). Enhancing IoT security: An innovative key management system for lightweight block ciphers. *Sensors*, 23(18), 7678. <https://doi.org/10.3390/s23187678>
- [8] Hashem, M. I., & Kuban, K. H. (2023). Key generation method from fingerprint image based on deep convolutional neural network model. *Nexo Revista Científica*, 36(6), 906–925. <https://doi.org/10.5377/nexo.vXXiXX.XXXX>
- [9] Erkate, U., Toktas, A., Enginoglu, S., Karabacak, E., & Thanh, D. N. H. (2024). An image encryption scheme based on chaotic logarithmic map and key generation using deep CNN. *Expert Systems with Applications*, 237, 121452. <https://doi.org/10.1016/j.eswa.2023.121452>
- [10] Quinga Socasi, F., Zhinin-Vera, L., & Chang, O. (2020). A deep learning approach for symmetric key cryptography system. In *Proceedings of the Future Technologies Conference* (pp. 41). https://link.springer.com/chapter/10.1007/978-3-030-63128-4_41
- [11] Wu, Z., Lv, Z., Kang, J., Ding, W., & Zhang, J. (2022). Fingerprint bio-key generation based on a deep neural network. *International Journal of Intelligent Systems*, 37(7), 4329–4358. <https://doi.org/10.1002/int.22782>
- [12] Alesawy, O., & Muniyandi, R. C. (2016). Elliptic Curve Diffie-Hellman random keys using artificial neural network and genetic algorithm for secure data over private cloud. *Information Technology Journal*, 15(2), 77–83. <https://doi.org/10.3923/itj.2016.77.83>
- [13] Saini, A., & Sehrawat, R. (2024). Enhancing data security through machine learning-based key generation and encryption. *Engineering, Technology & Applied Science Research*, 14(3), 14148–14154. <https://doi.org/10.48084/etasr.7181>
- [14] Kurtynykh, I., Ghita, B., & Shiaeles, S. (2021). Comparative analysis of cryptographic key management systems. King's College London, Strand, London, WC2R 2LS, UK. <https://doi.org/10.48550/arXiv.2109.09905>
- [15] SSL Support Team. (2024, May 3). Key management best practices: A practical guide. Retrieved from [SSL Support Team Website] <https://www.ssl.com/article/key-management-best-practices-a-practical-guide/>
- [16] Wang, L., & Lv, Y. (2024). Differential privacy-based data mining in distributed scenarios using decision trees. *Informatica*, 48(2), 145–158. <https://doi.org/10.31449/inf.v48i23.6918>
- [17] Tu, Z., Milanfar, P., & Talebi, H. (2023). MULLER: Multilayer Laplacian Resizer for Vision. ResearchGate. Retrieved from https://www.researchgate.net/publication/369855623_MULLER_Multilayer_Laplacian_Resizer_for_Vision
- [18] Saifullah, S., Pranolo, A., & Dreżewski, R. (2024). Comparative analysis of image enhancement techniques for brain tumor segmentation: Contrast, histogram, and hybrid approaches. *Journal Name*, Volume (Issue), Page range. <https://doi.org/10.48550/arXiv.2404.05341>

- [19] Singh, P., Dutta, S., & Pranav, P. (2024). Optimizing GANs for Cryptography: The Role and Impact of Activation Functions in Neural Layers Assessing the Cryptographic Strength. *Applied Sciences*, 14(6), 2379. <https://doi.org/10.3390/app14062379>.
- [20] Zhang, B., & Liu, L. (2023). Chaos-Based Image Encryption: Review, Application, and Challenges. *Mathematics*, 11(11), 2585. <https://doi.org/10.3390/math11112585>
- [21] Taylor, O. E., & Igiri, C. G. (2024). Enhancing image encryption using histogram analysis, adjacent pixel autocorrelation test in chaos-based framework. *International Journal of Computer Applications*, 186(22). <https://doi.org/10.5120/ijca202492338>
- [22] Munshi, N. H., Das, P., & Maitra, S. (2022). Chi-Squared Test Analysis on Hybrid Cryptosystem. Volume 14, Issue 1, 34-40. <https://doi.org/10.2174/1876402913666210508235706>.
- [23] Rasheed, A. F., Zarkoosh, M., & Abbas, S. (2023, October). Comprehensive Evaluation of Encryption Algorithms: A Study of 22 Performance Tests. 2023 Sixth International Conference on Vocational Education and Electrical Engineering (ICVEE), Surabaya, France, 191-194. <https://doi.org/10.1109/ICVEE59738.2023.10348240>.
- [24] Feng, L., Du, J., & Fu, C. (2023). Double graph correlation encryption based on hyperchaos. *PLOS ONE*, 18(9), e0291759. <https://doi.org/10.1371/journal.pone.0291759>.
- [25] Barker, E., & Roginsky, A. (2024). NIST SP 800-131A Rev. 3: Transitioning the use of cryptographic algorithms and key lengths (Initial Public Draft). National Institute of Standards and Technology. <https://doi.org/10.6028/NIST.SP.800-131Ar3.ipd>.
- [26] Avaroğlu, E., Kahveci, S., & Akkurt, R. (2024). Optimization of Acoustic Entropy Source for Random Sequence Generation Using an Improved Grey Wolf Algorithm. *Computer Engineering Department, Faculty of Engineering, Mersin University*. <https://doi.org/10.18280/ts.410220>
- [27] Foreman, C., Yeung, R., & Curchod, F. J. (2024). Statistical testing of random number generators and their improvement using randomness extraction. *Cryptology ePrint Archive*, Paper 2024/492. Retrieved from <https://doi.org/10.3390/e26121053>
- [28] Taylor, O. E., & Igiri, C. G. (2024). Enhancing image encryption using histogram analysis, adjacent pixel autocorrelation test in chaos-based framework. *International Journal of Computer Applications*, 186(22). <https://doi.org/10.5120/ijca2024923653>
- [29] Saeed, F., Hussain, M., & Aboalsamh, H. A. (2022). Automatic fingerprint classification using deep learning technology (DeepFKTNet). *Mathematics*, 10(8), 1285. <https://doi.org/10.3390/math10081285>
- [30] Nahar, P., Chaudhari, N. S., & Tanwani, S. K. (2022). Fingerprint classification system using CNN. *Multimedia Tools and Applications*, 81(17), 24515–24527. <https://doi.org/10.1007/s11042-022-13494-6>
- [31] Nguyen, H. T., & Nguyen, L. T. (2019). Fingerprints classification through image analysis and machine learning method. *Algorithms*, 12(11), 241. <https://doi.org/10.3390/a12110241>
- [32] Ang, L.-M., Seng, K. P., Ijamaru, G. K., & Zungeru, A. M. (2018). Deployment of IoV for smart cities: Applications, architecture, and challenges. *IEEE Access*, 7, 6473–6492. <https://doi.org/10.1109/ACCESS.2018.2886575>
- [33] Saeed, F., Hussain, M., & Aboalsamh, H. A. (2018a). Classification of live scanned fingerprints using dense SIFT based ridge orientation features. 2018 1st International Conference on Computer Applications & Information Security (ICCAIS), 1–4. <https://doi.org/10.1109/CAIS.2018.8441995>
- [34] Saeed, F., Hussain, M., & Aboalsamh, H. A. (2018b). Classification of live scanned fingerprints using histogram of gradient descriptor. 2018 21st Saudi Computer Society National Computer Conference (NCC), 1–5. <https://doi.org/10.1109/NCC.2018.8682629>

