AN ENHANCED CYBER SECURITY FOR DORSAL FINGER KNUCKLE PATTERNS AND FINGER NAILS RECOGNITION SYSTEM BASED ON A CONCATENATED DEEP LEARNING MODEL WITH GRAIN-128A ENCRYPTION ALGORITHM

Haitham Salman Chyad[1*], and Tarek Abbes[2]

[1*,2] NTS'COM, National School of Electronics and Telecommunications of Sfax, University of Sfax, Tunisia

[1*] Department of Computer Science, College of Education, Mustansiriyah University, Baghdad, Iraq

Emails: dr.haitham@uomustansiriyah.edu.iq[1*] (Corresponding Author), tarek.abbes@enetcom.usf.tn[2]

ABSTRACT

The shift to biometric systems where unique qualities of a user used in place of standard passwords has brought a significant change in cybersecurity. The biometrics which consist of patterns of finger knuckles and fingernails give accurate and reliable identity recognition, and therefore minimize chances of unauthorized access and impersonation. The massive advancement of deep learning models allows the extraction of accurate biometric information in the images with the help of convolutional neural networks (CNNs), which are after that integrated to highly discriminative and consistent feature space. This high level of integration can help to build greater recognition strength and the safety of cryptographic keys that are produced by biometric features which is why this strategy can be chosen as one of the cornerstones in developing the safe biometric recognition systems that can help to build cybersecurity in the contemporary settings. This research introduces an enhanced cybersecurity for a dorsal finger knuckle pattern and finger nail recognition system, namely (ECS-DFKPFNRS), depending on an integrated feature deep learning (DL) with the cryptography algorithm. This ECS-DFKPFNRS utilizes nineteen key components of the dorsal finger knuckle pattern of ten fingers for concatenated fusion recognition of all the features extracted. This system applies a concatenated DL model to four pre-trained models: InceptionNetV3, ResNet50, DenseNet201, and MobileNet-V3. To conduct this, the ECS-DFKPFNRS utilizes a new segmentation method that relies on the Hands Landmark Module (MediaPipe Module) to detect key knuckle hand components (Five-Basic, Five-Major, Four-Minor, and the fingernails) for both hands. After that, the concatenated fusion feature vectors extracted from the concatenated DL model will be utilized to generate a symmetric key (secret key) by the Rubik's cube technique. This technique will be improved by incorporating logistic scrambling, permutations, and SHA-512 hashing; subsequently, these keys to generated after the enhanced Rubik's cube technique and will be utilized to encrypt biometric data by employing the lightweight Grain-128a algorithm. This is an integrated deep feature for FKP and FN and encryption, aiming to guarantee confidentiality and integrity against statistical attacks. 11,076K Hands and the 'Poly U HD' datasets were utilized to analyze and evaluate this system. The system which was proposed was more effective on these datasets with outstanding security measures: NPCR (99.95% and 99.63%), UACI (30.23% and 30.21%), and resistance to statistical attacks. It is demonstrated in this study that the combination of DL and biometric encryption improves the reliability of biometric data and biometric robustness, which improves cybersecurity and digital biometric identity protection.

Keywords: Concatenated DL model; InceptionNetV3 model; ResNet50 model; DenseNet201 model; MobileNet-V3 model;Rubik's Cube technique; Grain-128a encryption algorithm.

1.0 INTRODUCTION

In a modern digitalized society, the need to protect sensitive information and digital resources is becoming a necessity in the context of increasing the number of cyberattacks. Traditional methods of recognition used in cybersecurity in the past have been passwords and personal identification numbers (PINs). With regard to automated biometric recognition, the first paper on the research publication was by Mitchell Trauring and published in the journal Nature in 1963, which dealt with fingerprint matching. In the 1960s, automated biometric recognition systems based on alternative characteristics started to develop like voice, face and signatures. Later on, biometric recognition systems were invented using iris, hand geometry, finger knuckle and fingernail patterns as the characteristics [1].

Hand-based identification systems identify fingerprints, palm prints, hand geometry, hand shape, and hand veins [2], [3]. The hand-based systems have been a subject of considerable focus over a long duration. The performance of the systems can be explained by the fact that they have reliable features that ensure stability,

acceptance, simplicity, and robustness [4]. In the present day, hand-based technology finds application by many organizations, industries and government agencies due to a number of reasons, which comprises security [5]. Recently, it has been found out that the finger knuckle pattern (FKP) along with the fingernail knuckle patterns (FNP) can be regarded as distinct biometric characteristics in secure recognition systems, in that they are fine anatomical structures, which include fine lines, surface texture, and layers structure [6], [7]. The knuckle structure of the skin of a finger knuckle is composed of specific transverse folds and knuckle lines, while the knuckle structure of the fingernail is made up of finer radial knuckle lines, gradient color texture, and clear surface ridges that can be seen as an individual, non-duplicable fingerprint. The inclusion of the finger knuckle and FN knuckle is a significant process towards the improvement of the accuracy of the recognition system. The finger knuckle supports deep skin features, whereas FN knuckle adds reflectivity and color contributing to resistance to spoofing attacks and enhanced strength of recognition [8], [9]. The finger knuckle patterns (FKP) are an original, universal, and unchangeable biometric pattern that is used to identify the person with high accuracy.

Recent FKP studies have been based on correct feature extraction, non-constrained and contactless capture methods, and multi-level fusion methods. Nevertheless, there are still few works in the literature that cover the process of integrating the finger knuckle with FN knuckle into a single system that can improve the performance and increase the efficiency of the intelligent recognition systems [10]. The field of deep learning (DL), and particularly convolutional neural networks (CNNs), has transformed the quality of the analysis and removal of biometric data in the most precise way. The progress of the pre-trained models, such as FaceNet that can recognize faces and networks, which can identify finger veins and palm prints can also be applied to FKP and FNP to obtain precise and profound representations of anatomical and tactile characteristics [11], [12]. The strategy enables one to incorporate features of both knucks with concatenated deep networks (CDNs), and generate a rich, coherent feature space that builds on the recognition system accuracy and helps to produce specific and secure cryptographic keys.

DL combined with biometrics marks an important step in achieving systems capable of achieving high-level cybersecurity and resistance to attacks. Biometrics use in cybersecurity expands the ability of the systems to overcome the limitations of identification in traditional systems and effectively counter the new threats. The systems provide precise and reliable user identification, which minimizes chances of unauthorized access and fraudulent attacks. The fact that they are easily integrated into the smart devices such as smartphones and laptops has increased the level of consumer acceptance and utilization making them a very critical component of the present-day security systems. The use of those technologies raises the issue of biometric data privacy and security, the possibility of spoofing and forgery, and the problem of compliance with the regulations. This would require more intelligent and secure biometric recognition systems to be developed. In the given situation, embedded deep learning (Deep Feature Fusion) is crucial in the extraction of detailed features on Dorsal-FKP and FN and their integration into a cohesive feature space. This strengthens the recognition system and makes it easier to develop a secure and strong key to cryptography. According to it, the current work is devoted to the development of an integrated biometric recognition system on the deep integration and encryption and the purpose of this work is to increase the cybersecurity level through high reliability and attack resistance.

In our study, we suggest an improved version of cybersecurity of the Dorsal-FKP and FN recognition system, which is named ECS-DFKPFNRS, which utilizes the concatenated DL model through the use of four strong architectures with coherent representations to extract the feature vectors of all D-FKP and FN of the fingers (little, ring, middle, index, and thumb) of both hands. To achieve this, the ECS-DFKPFNRS utilizes various processing procedures in an attempt to extract the significant parts of DFKP and FN of each of the ten fingers of both hands. Moreover, the ECS-DFKPFNRS use these features to produce a secret key (SK) based on the Rubik cube method that improves the process. The method involves the use of a logistic chaotic system (LCS) to generate chaotic sequences, which are used in the logistic scrambling process to safely and irregularly rearrange image data so as to increase the encryption strength and complicate reverse analysis. The encrypted properties of each concatenated DL model are encrypted with the help of the lightweight Grain-128a encryption algorithm that is a symmetric stream cipher. The suggested lightweight Grain-128a encryption, which uses the Rubik’s cube method relies on concatenation of four DL models, is of high value as compared to the current encryption schemes. The main value of the proposed system may be stated in the following way:

1. Proposing an effective concatenated DL model for extracting optimal feature vectors. The vectors extracted from each one of the models (e.g., InceptionV3, ResNet 50, DenseNet201, and MobileNetV2) have been stacked into one large, high-dimensional vector, therefore, allowing the system to use the complementary representations provided by each model. This concatenation enhances speed and accuracy and thus it is applicable in real-time. In this way, it ensures effective processing and performance is maintained.
2. Proposing the Rubik's Cube technique for generating a symmetric key (secret key) for every of the ten fingers on both hands—the thumb-knuckle (ThK), the base-knuckle (BaK), the major-knuckle (MaK), the minor-knuckle (MiK), and the FNs—that make up the DFKP and FN fusion key components. The permutations, logistic scrambling, and SHA-512 hashing of the encrypted biometric data are added by this Rubik's Cube technology, and processed on all its faces all at once.

3. Proposing the concatenated fusion of DFKP and FN features for all fingers (little, ring, middle, index, and thumb) for both hands, along with extracting them individually to distinguish between them and evaluate the effectiveness of concatenated fusion features. The concatenated DFKP and FN features enhance recognition accuracy by integrating patterns from several fingers. Combined feature vectors from several knuckle areas offer more discriminative data to improve resilience against hand orientation, lighting conditions, and textural noise. This integrated representation enhances biometric recognition by diversifying features, reducing false matches, and facilitating the creation of strong and secure encryption keys.
4. Proposing a Grain-128a lightweight encryption algorithm for the encryption of fusion nineteen key components for DFKP and FN for both hands, utilizing features generated by a concatenated DL model.
5. Proposing additional analysis and simulation testing to validate the effective performance of the proposed ECS-DFKPFNRS and the system's resilience against different statistical attacks. It has been demonstrated that the proposed system is dependable, offering reliability, adequate security, and resilience for diverse biometric templates.

The following sections of this research are organized as outlined below: Section 2 reviews the state-of-the-art. Section 3 presents the procedures that are defined. Section 4 highlights the analysis and evaluation results for the concatenated DL model, as well as the analysis encryption metric and statistical attack. Section 5 covers the comparison and discussion. Section 6 addresses the conclusion.

2.0 RESEARCH BACKGROUND

In this section, the state of the art in DFKP and FN is reviewed. The following subsection summarizes the state-of-the-art in the field of combining deep learning models and cryptography in biometric systems.

2.1 Deep Learning Models for DFKP and FN Recognition

Thapar, Jaswal, and Nigam [13] suggested a “convolutional neural network” (CNN) that generates a 128-dimensional feature embedding of every dorsal finger image. This embedding is essential for the recognition system, as it captures the distinctive features of the dorsal finger knuckle images. A triplet loss function, which aims to maximize the distance between distinct subjects and minimize the distance between embeddings of the same subject, is used to train the network. This strategy ensures that the model effectively learns to distinguish between multiple users. The experiment was conducted on the performance of each finger utilizing two publicly accessible dorsal finger knuckle image datasets, namely the PolyU FKP and the PolyU Contactless FKI datasets. A weighted sum of scores for the major knuckle, minor knuckle, and nail modalities was computed, substantiating our hypothesis to consider the entire finger as a biometric entity rather than its individual components. Although CNNs achieve the best results, deep learning models, especially CNNs, can be computationally demanding, which could be a drawback for real-time applications on resource-limited devices. An innovative deep learning model called NailNet was introduced by Fan et al.[14] for highly accurate fingernail plate and crescent segmentation and classification. The DeepLabv3+ model, with its encoder-decoder architecture, has achieved the best result on the publicly accessible dataset, comprising 6250 images. But this study has its drawbacks. The different finger parts in hand images should be kept separate from one another to distinguish the nail area. The VGG-19 model was suggested by Tarawneh et al. [15] to extract important deep features from FKP images. Deep features are extracted from the fully connected layer 6 (Feature6) and fully connected layer 7 (Feature7) of the VGG-19 model. Following the utilization of various preprocessing methods, including the integration of features from diverse layers and the execution of dimensionality reduction utilizing “principal component analysis” (PCA), the derived deep features are subjected to evaluation. Several methods of machine learning (ML), such as the “Artificial Neural Network” (ANN), the “K-Nearest Neighbors” (KNN), “Naive Bayes” (NB), and the “Random Forest” (RF), are used for classifiers based on deep features extracted from FKP images. The experiments achieved the best results by utilizing the Delhi Finger Knuckle dataset. Although the study primarily looks at the VGG-19 feature extraction model, its restricted analysis of other DL models limits the scope of its findings.

A hybrid VGG-16 model and RF classifier was created by Sharma et al. [16] to extract features using the VGG-16 model from images of a person's fingernails for disease diagnosis. RF was employed for classification and proved to be highly effective in classifying fingernail images into distinct disease classes. The model's performance achieved good results, using datasets from Kaggle27 and Roboflow29. This hybrid model might not be appropriate for real-time applications or devices with limited resources due to its high computational complexity. A CNN was presented by M. Afifi [17] to extract important deep features relying on images of their hands to determine an individual's gender. “Support Vector Machine” (SVM) is utilized for the classification of deep features obtained from CNN. The experiments achieved satisfactory results for both dorsal hand images and palm hand images by using the 11-K hand datasets. This puts a challenge on the process of training the model where CNN is being integrated with SVM classifiers, which is an obstacle to real time applications. In their study Altaher et al. [18] used AlexNet, DenseNet, EfficientNet, GoogleNet, Shallow Convolutional Neural Networks (SCNNs), ResNet50, and Vision Transformer (VT) models to extract features and classify Finger Knuckle Prints (FKP) images. The

experiment on the training and testing was organized with the help of the Hong Kong Polytechnic University (PolyU), with which good results were attained. The paper does not address various criterion among them being the cost of computing, training time, complexity of the model, and ability to survive in different environmental conditions, which are important issues when it comes to real use.

2.2 Cryptographic Key Generation Techniques for DFKP and FN

Hang, and Yang[19] proposed a new biometric to create a key using nail images called NailKey where information on the edge of sanded fingernails is recorded by the key, using existing capacitive fingerprint sensors. In order to extract features of the biometric data, a tree based algorithm was applied to accurately and efficiently match and compare features. The NailKey demonstrates certain diversity and sustainability in the analysis, which proves its effectiveness and possibilities of use. Compared to more conventional biometrics like fingerprint and facial recognition, NailKey still suffers from weaknesses in both FRR (usability) and FAR (security), despite efforts to enhance it. A key was generated using the Bio-Hashing technique introduced by Singh et al. [20] to create a secure network for generating a "Cancellable finger dorsal" (CDF) template (learning domain-specific features) using this technique. Extracting important features is based on the "Finger Dorsal Feature Extraction Network" (FDFNet), which trains features without using any pre-trained architecture. These features are employed with a hash by specifying a distinct random number to each user to generate a key. The test performed effectively achieved a high level of security using two public datasets (PolyU FKP and PolyU Contactless FKI). The key generated based on the trained feature network lacks comparability with other deep learning networks. This comparison highlights the strength or weakness of the key. Biometric key components were suggested by Arunachalam, and Subramanian [21], and are based on key generation in FKP components with the use of the k-means algorithm and the Hash Algorithm (SHA) function to generate a secret hash value. The FKP key elements were extracted through the use of the Scale Invariant Feature Transform (SIFT) which was encrypted using the Symmetric Advanced Encryption Standard (AES) algorithm. Consequently, the secret value and the biometric data are both encrypted by the secret encrypted hash value. Besides giving the ability to check errors, the hash function secures the biometric data against any form of malicious modification that they might be subjected to. The study did not cover the performance of the proposed methodology in different real-world conditions and how the method can be extended to a great number of users, which might result in variations in FKP images.

3.0 MATERIALS AND METHODS

This section discusses the Grain-128a encryption algorithm, which has been applied to improve cybersecurity by outlining the DFKP and FN datasets for extracting knuckle characteristics order to encrypt these characteristics.

3.1 Dataset Description

These datasets refer to the “11-k Hands” and the “Hong Kong Polytechnic University contactless hand dorsal Images” datasets which are used in the current work in the context of testing concatenated deep learning models and Rubiks Cube method of cryptographic key generation. The set of 11k Hands is 11,076 hand images of 190 users, including both palmar and dorsal perspectives. The images show hands in different postures, that is, closed, partial closed, and fully open. Each image in this dataset has been captured using a USB document camera with a resolution of 1600 x 1200 pixels and the hands have been positioned at roughly the same distance of the camera [17]. This dataset consists of 4,650 dorsal images of right hands of 501 users all lying flat with open fingers and this is known as the Poly UHD dataset. The camera of the handheld nature was used to capture images of 1600 × 1200 pixels, and the indoor and outdoor lighting conditions were managed [22]. In addition, since the subject of this study is DFKP and FN, 2738 left and 2755 right images were selected, which gave 52,022 left and 52,345 right segmented images. From the Poly UHD dataset, 4,650 right hands were chosen, obtaining 88,350 segmented right images. To prevent overfitting and to facilitate the training process, the users allocated 70% of the data for training, 30% for validation, and 30% for testing the concatenated DL model on both datasets.

3.2 Grain-128a Lightweight Encryption Algorithm

The latest version of the Grain family of encryption algorithms, which was developed by Hell et al. in 2011, is Grain-128a [23], [24], a lightweight encryption algorithm that is highly secure [25]. The Grain-128a supports a 128-bit secret key and a 96-bit initial vector (IV). The Grain-128a system is comprised of a "128-level linear feedback shift register" (LFSR), a "128-level non-linear feedback shift register" (NFSR), and a non-linear filter function. For the LFSR and NFSR, the feedback functions are f and g, respectively. As seen in Fig.1, certain bits of the LFSR and NFSR are utilized to generate the keystream [26].

Step 1: The 256-bit binary sequence $\{k_0, k_1, \ldots, k_{255}\}$ is the result of decomposing into the first part of the secret key, k32. Assign the initial states of LFSR and NFSR to $\{k_0, k_1, \ldots, k_{127}\}$ and $\{k_{128}, k_{129}, \ldots, k_{255}\}$, respectively. Next, assume that the contents of the LFSR are $\{p_i, p_{i+1}, \ldots, p_{i+127}\}$, and the contents of the NFSR are $\{p_i, p_{i+1}, \ldots, p_{i+127}\}$.
Step 2: These three functions—update, filter, and output—all produce a new binary sequence.
In contrast to the LFSR, the NFSR's update function includes the LFSR's state bits, as seen in equation 1.

$$p_{i+128} = p_i + p_{i+7} + p_{i+38} + p_{i+70} + p_{i+81} + p_{i+96}. \quad (1)$$

In contrast to the LFSR, the NFSR's update function includes the LFSR's state bits, as seen in equation 2.

$$\begin{aligned} q_{i+128} = p_i + q_i + q_{i+26} + q_{i+56} + q_{i+91} + q_{i+96} + q_{i+3}q_{i+67} + q_{i+11}q_{i+13} + \\ q_{i+18} + q_{i+27}q_{i+59} + q_{i+40}q_{i+48} + q_{i+61}q_{i+65} + q_{i+68}q_{i+84} + \\ q_{i+22}q_{i+24}q_{i+25} + q_{i+70}q_{i+78}q_{i+82} + q_{i+88}q_{i+92}q_{i+93}q_{i+95}. \end{aligned} \quad (2)$$

The non-linear filter function is shown as follows, before the final output

$$\begin{aligned} h(x) = q_{i+12}p_{i+8} + p_{i+13}p_{i+20} + q_{i+95}p_{i+42} \\ +p_{i+60}p_{i+79} + q_{i+12}q_{i+95}p_{i+94}. \end{aligned} \quad (3)$$

Lastly, binary sequence update, $H_i (i \in [0,255] \in N)$, is derived from

$$\begin{aligned} H_i = h(x) + p_{i+93} + q_{i+2} + q_{i+15} + q_{i+36} \\ +q_{i+45} + q_{i+64} + q_{i+73} + q_{i+89}. \end{aligned} \quad (4)$$

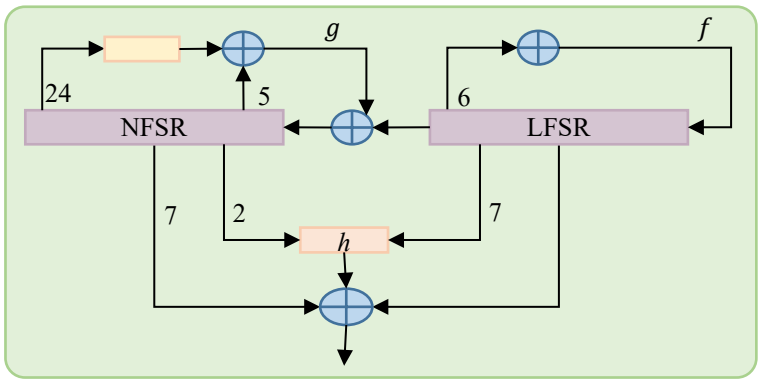


Fig. 1: Grain-128a stream cipher

3.3 Proposed System

In our system, named ECS-DFKPFNRS, this work uses four effective models: The InceptionV3 model, the ResNet50 model, the DenseNet201 model, and the MobileNetV2 model. These models are concatenated and enhanced using fine-tuning feature extraction (FE). Our system framework consists of the following main stages: preprocessing, feature extraction (FE), feature-level fusion of DFKP and FN at the concatenated, the Rubik's Cube technique for key generation, and the Grain-128a lightweight encryption algorithm. This framework provides details of each stage the user goes through in the proposed overall framework. Fig. 2 depicts the framework of the proposed system.

3.3.1 Pre-processing Stage

In this stage, the DFKP and FN for five fingers are identified utilizing the hand image. This study employs a hand posture estimate using the hand's module (MediaPipe Module) to determine the location of a hand area, such as DFKP and FN. The main components of the hand are identified using the main points of this model. While the proposed method was used to obtain DFKP and FN images in this work, it was recently recorded and published in our research [27], where it was employed to obtain inner knuckle patterns (IKP). The best localization key points result was obtained by resizing the original manual image to 224 x 224. The DFKP and FN detection method is implemented using a number of processing steps, which consist of:

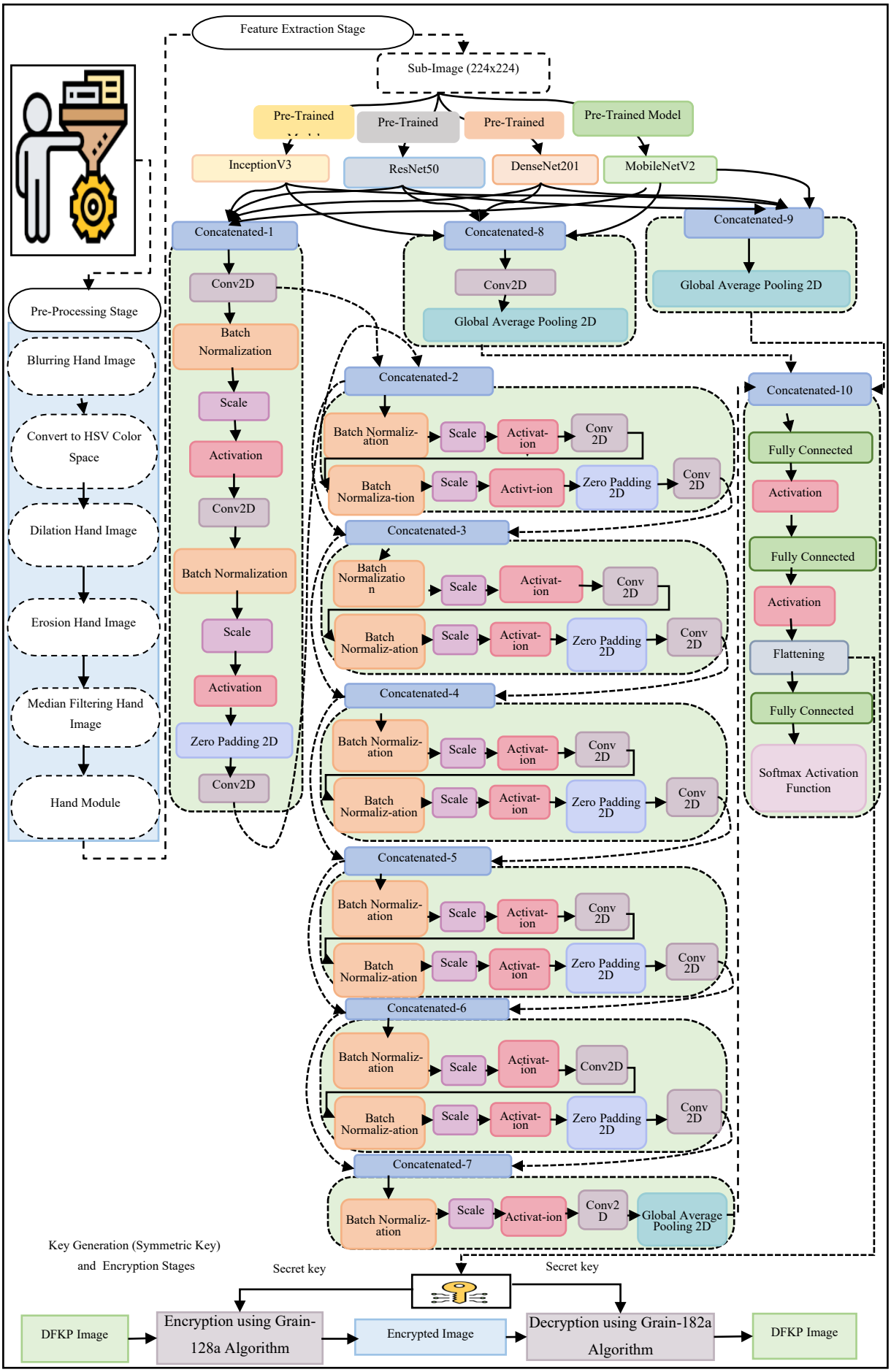


Fig. 2: The Proposed Framework for ECS-DFKPFNRS

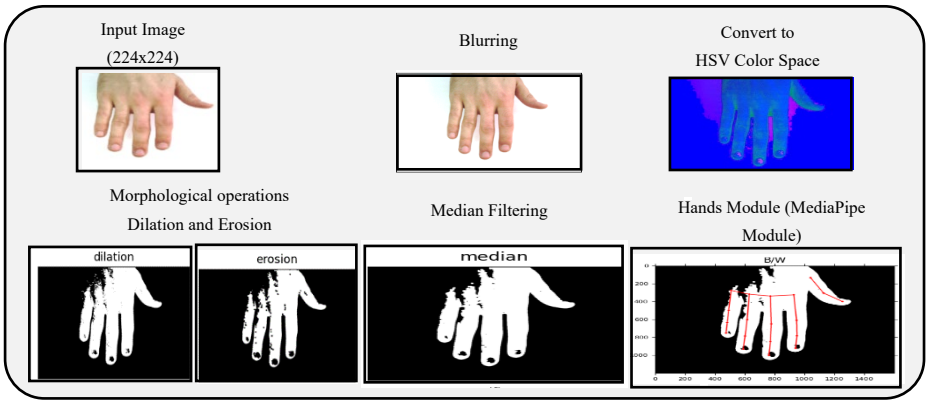


Fig. 3: The processing stages for detecting the DFPK using 11K Hands dataset

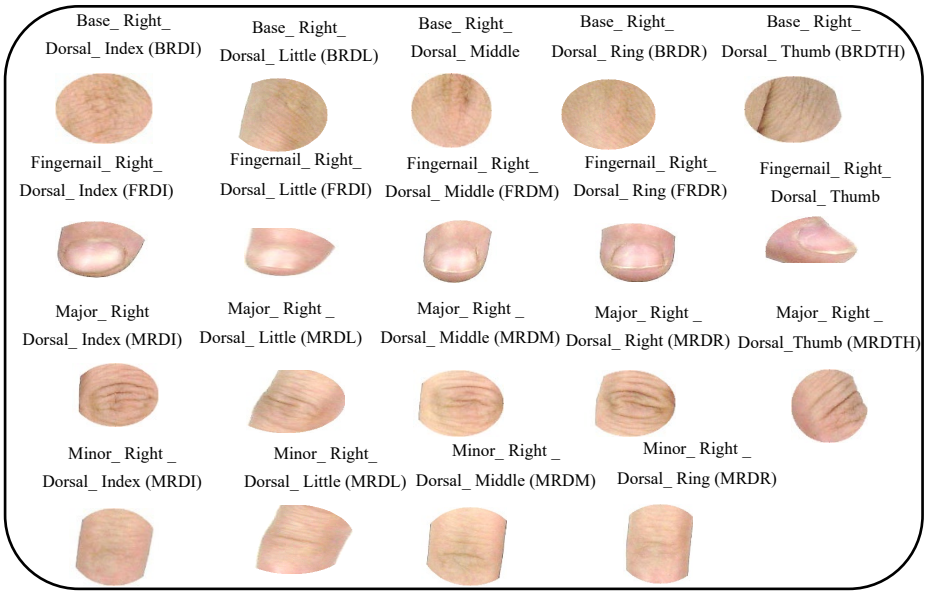


Fig. 4: Sample of the sub-images of the hand's key components for DFPK using 11K Hands dataset

1. Blurring step: An image can be made blurrier by using a filter, which also eliminates more noise. An important component of image processing is image blurring [28], [29]
2. HSV color space step: Using techniques for identifying the selected skin region after converting the RGB color space to HSV, this step aims to identify the skin region in the hand image [30], [31], [32].
3. Morphological operations step: These are a set of steps that use shapes to manipulate images. Adding a structural element to an input image results in an output image. Erosion and Dilation are two of the most fundamental morphological processes [33], [34], [35].
4. Median filtering step: This is an excellent first step in decreasing this kind of noise. Using a small matrix, such as the 3 x 3, the filtering process scans the entire image. After that, it utilizes the median of each of the values inside the matrix to compute the value of the center pixel [36], [37], [38].
5. MediaPipe Module step: This strategy tracks hands and fingers with a high degree of accuracy. Machine learning (ML) is used to guess nine 2D hand landmarks from a single shot [39], [40].

As shown in Fig. 3, the hand's key points will be cropped into sub-images by the execution of four processing steps. Both hands have 19 distinct components, as shown in Fig. 4.

3.3.2 Feature Extraction (FE) Stage

Extracting features is a crucial stage for analyzing the dorsal surface of the DKFP and FN, due to the distinctive and non-repetitive patterns present in these regions. The DKFP exhibit fine lines and wrinkles that are consistently stable throughout time, whereas FNs provide additional morphological and textural features (shape and surface

features) that enhance biometric identification. By extracting these features, raw images are transformed into the concise numerical form that captures the most stable and distinguishable features, and diminishes the effect of noises and externalities (e.g. lighting or position of hands). The integration of the two sources of features significantly increases the uniqueness and reliability of biometric systems of identification [41], [42]. To improve the efficiency and accuracy of the biometric recognition systems using DFKP and FN images a deep model fusion method was used. This strategy combines four powerful architectures, including InceptionNetV3, ResNet50, DenseNet201, and MobileNet-V3. Each of these models was pre-trained using the ImageNet to ensure the extraction of rich and relevant features and then refined using the target dataset. Following feature-level concatenation, the concatenated features were inputted into fully connected layers with a Softmax classifier to do the full classification, which leads to the coherent classification exploiting the merits of the four models.

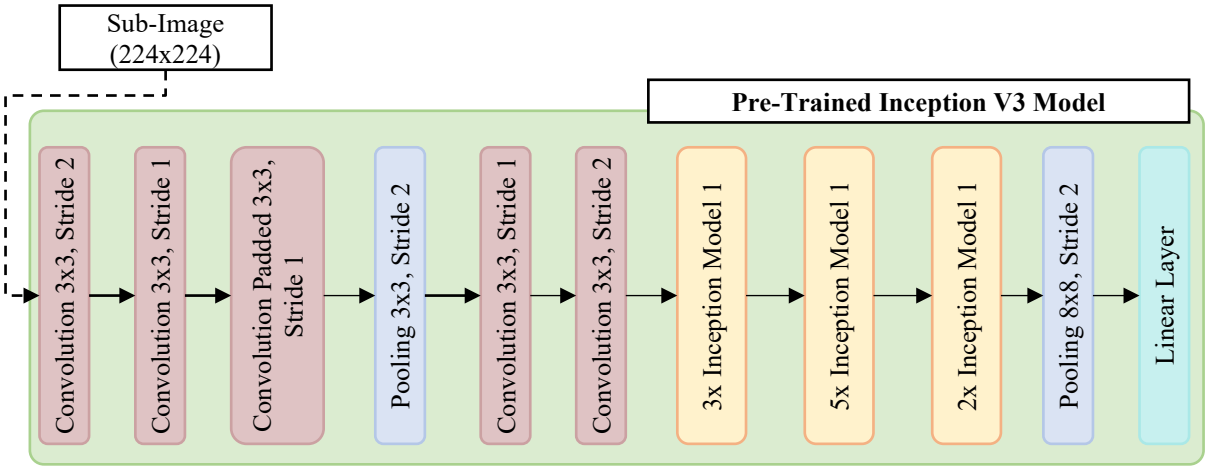


Fig. 5: The InceptionV3 model

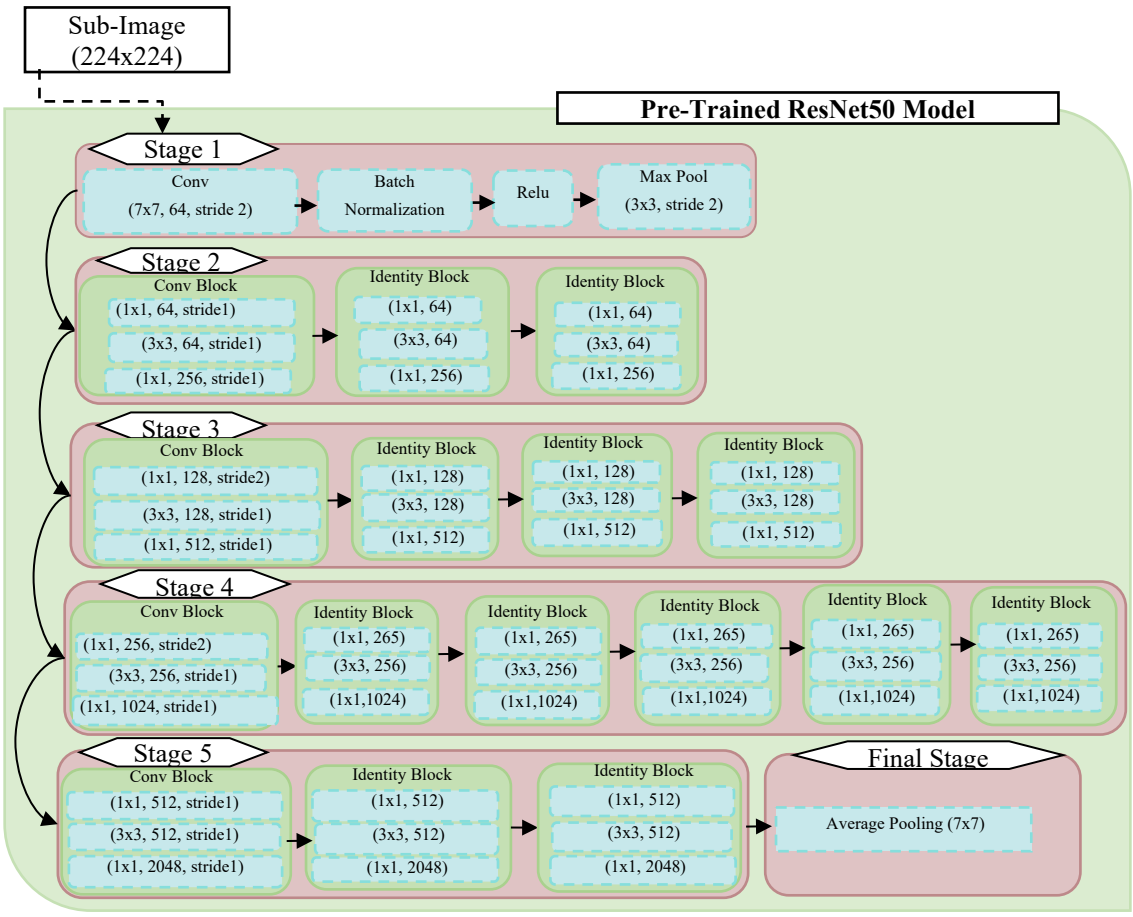


Fig. 6: The ResNet50 model

- InceptionV3 Model: It incorporates various types of kernels at the same level to "widen" the network and reduce heterogeneity in prominent feature location in the analyzed images. Multiple kernels working at the same level are made easier by Inception modules. On this foundation, InceptionV1 (Google Net) was proposed [43]. As introduced in [44], InceptionV2 and InceptionV3 designs augmented InceptionV1. Kernel factorization, batch normalization for auxiliary classifiers, and representational bottlenecks were added to enhance performance. The ILSVRC 2015 image classification competition placed InceptionV3 as the first runner-up. Refer to the pre-trained InceptionV3 Model depicted in Fig. 5.
- ResNet50 Model: It is a deep CNN architecture that uses the residual blocks (RBs) in Fig. 6 to mitigate the vanishing gradient issue. Each block incorporates skip connections that facilitate the effective propagation of gradients, hence enabling the training of exceedingly deep networks [45]. ResNet50 features a bottleneck architecture with layers structured into five stages, integrating convolutional operations and identity mappings, succeeded by global average pooling (GAP) and a fully connected layer (FC) for classification. As a result of its ability to produce excellent performance on image recognition tasks, this architecture has become an essential component of contemporary deep learning.
- DenseNet201 Model: One feature of the Dense Net architecture is its dense connections. linking each layer to all other layers enhances the ResNet architecture [46]. In these intricately linked structures, each layer acquires feature maps from all preceding layers and conveys its own feature map to the subsequent layer. Another significant advantage of this type of architecture is the ability to reuse features while utilizing the fewest possible parameters overall. One of the several widely utilized variants of the DenseNet architecture is the DenseNet201 architecture, which was employed in this study. Refer to the pre-trained DenseNet201 Model depicted in Fig. 7.

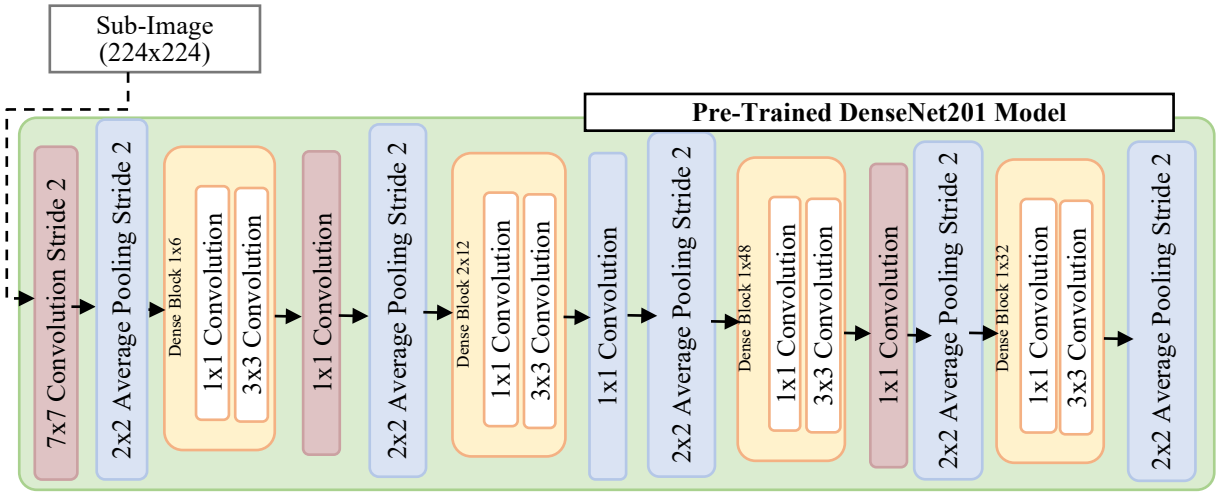


Fig. 7: The DenseNet201 model

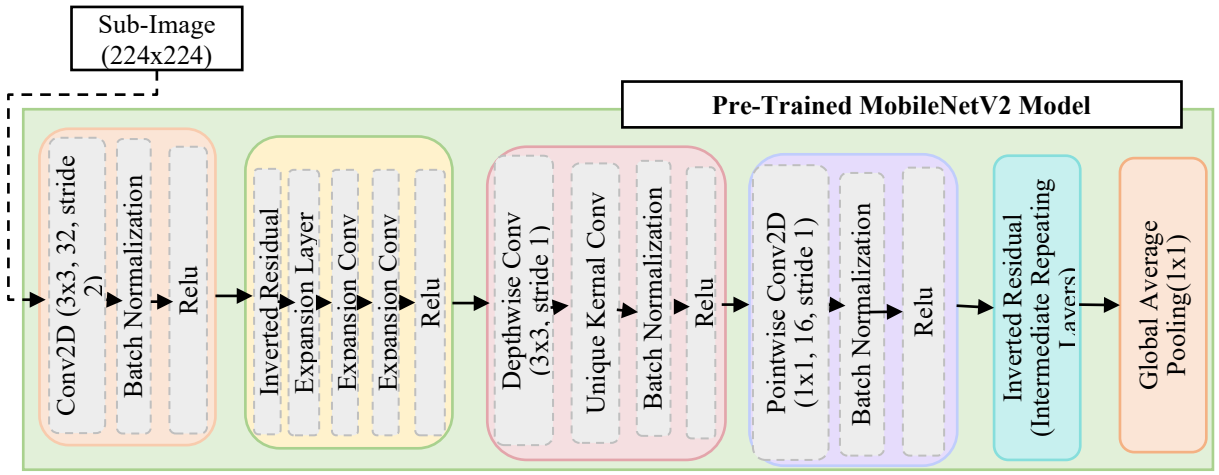


Fig. 8: The MobileNetV2 model

- MobileNetV2 Model: A lightweight and efficient CNN for mobile and embedded vision applications, as depicted in Fig.8. The model's input layer is first built for 224 × 224 × 3 images. After that, a 3x3 convolutional layer with 32 filters, a stride of 2, batch normalization (BN), and ReLU activation are utilized. MobileNetV2's core is linear bottlenecked inverted residual blocks. Additionally, each block includes batch normalization, ReLU activation, depthwise convolutions, expansion layers with 1×1 convolutions, and pointwise convolutions for channel size adjustment. Repeated blocks improve feature extraction. After several inverted residual blocks, the network combines spatial information with a global average pooling layer and a flattening layer. The last layers provide class probabilities with a fully connected layer (FC) and a softmax activation function. MobileNetV2 efficiently extracts features with fewer parameters, making it suitable for resource-constrained situations [47].

Completing extraction features from each model independently, the resultant feature vectors are concatenated into a cohesive representation, where the activation functions of each model (InceptionV3, ResNet50, Dense Net201, and MobileNetV3) are concatenated-1 that passed through convolution2D (Kernel (6x6x5440x8), bias (8)) that produces concatenated-2 with batch normalization (gamma(8), beta (8), moving_mean (8), and moving_variance (8)), scale (gamma (8), and beta (8)), activation functions, convolution2D (Kernel (1x1x8x384)), batch normalization (gamma(384), beta (384), moving_mean (384), and moving_variance (384)), scale (gamma (384), and beta (384)), activation functions, zeropadding2D, and convolution2D (Kernel (3x3x384x96). Next produces concatenated-3 with batch normalization (gamma (104), beta (104), moving_mean (104), and moving_variance (104)), scale (gamma (104), and beta (104)), activation functions, convolution2D (Kernel (1x1x104x384)), batch normalization (gamma (384), beta (384), moving_mean (384), and moving_variance (384)), scale (gamma (384), and beta (384)), activation functions, zeropadding2D, and convolution2D (Kernel (3x3x384x96). Then produces concatenated-4 with batch normalization (gamma (200), beta (200), moving_mean (200), and moving_variance (200)), scale (gamma (200), and beta (200)), activation functions, convolution2D (Kernel (1x1x200x384)), batch normalization (gamma (384), beta (384), moving_mean (384), and moving_variance (384)), scale (gamma (384), and beta (384)), activation functions, zeropadding2D, and convolution2D (Kernel (3x3x384x96). Followed by produces concatenated-5 with batch normalization (gamma (296), beta (296), moving_mean (296), and moving_variance (296)), scale (gamma (296), and beta (296)), activation functions, convolution2D (Kernel (1x1x296x384)), batch normalization (gamma (384), beta (384), moving_mean (384), and moving_variance (384)), scale (gamma (384), and beta (384)), activation functions, zeropadding2D, and convolution2D (Kernel (3x3x384x96). Then produces concatenated-6 with batch normalization (gamma (392), beta (392), moving_mean (392), and moving_variance (392)), scale (gamma (392), and beta (392)), activation functions, convolution2D (Kernel (1x1x392x384)), batch normalization (gamma (384), beta (384), moving_mean (384), and moving_variance (384)), scale (gamma (384), and beta (384)), activation functions, zeropadding2D, and convolution2D (Kernel (3x3x384x96). After then produces concatenated-7 with batch normalization (gamma (488), beta (488), moving_mean (488), and moving_variance (488)), scale (gamma (488), and beta (488)), activation functions, convolution2D (Kernel (1x1x488x384)), batch normalization (gamma (384), beta (384), moving_mean (384), and moving_variance (384)), scale (gamma (384), and beta (384)), activation functions, zeropadding2D, and convolution2D (Kernel (3x3x384x96). Followed by batch normalization (gamma (7), beta (7), moving_mean (7), and moving_variance (7)), scale (gamma (7), and beta (7)), activation functions, convolution2D (Kernel (1x1x584x67)), and global average pooling 2D. Finally, concatenated-8 and concatenated-9 are used to produce concatenated-10 with input into the softmax layer (Dense (Kernel (5515x256), and bias (256)), activation functions, (Dense (Kernel (256x1), and bias (256x1)), activation functions, and Dense to produce the probability of belonging to a class. By combining local, global, and lightweight features into a single, reliable, and effective model, this stage makes use of deep learning's strength. Table 1 displays the hyperparameter and parameter selections determined for the concatenated deep learning models.

Table 1: Hyperparameter values for concatenated DL models

Concatenated DL Models	
Hyper Parameters	**Values**
Input Size	224x224
Batch Size	32
Seed	42
Optimizer	SGD
Learning-rate	1e-3
Epochs	200
Loss Function	Binary-Cross Entropy
Dense	2048
Total parameters	52,123,327
Trainable Parameters	1,729,307
Non-Trainable Parameters	50,394,020

3.3.3 Feature-Level Fusion of DFKP and FN at the Concatenated Stage

A feature-level fusion process was conducted using a concatenation strategy after obtaining feature vectors from various deep models. The vectors extracted from each model (e.g., InceptionV3, ResNet 50, DenseNet201, MobileNetV2) have been combined into a single, comprehensive, high-dimensional vector, enabling the system to leverage the complementary representations provided by each model. Since each model extracts unique patterns of multi-level and multi-scale features from the input, this fusion enhances the representational power. It produces a richer and more robust vector for further classification or encryption. In addition to a concatenated fusion of DFKP and FN features for all fingers (little, ring, middle, index, and thumb) for both hands in the 11-k Hands and Poly UHD datasets, the concatenated stage produces three different feature-level fusions that are extracted independently for DFKP and FN-Basic, DFKP and FN-Major, and DFKP and FN-Minor, for each finger.

- Feature Vector Fusion for DFKP and FN-Right-Basic-Each model (InceptionV3, ResNet50, DenseNet201, MobileNetV2): (DFKP and FN-Right Little-Basic-Each model) + (DFKP and FN-Right Ring-Basic-Each model) + (DFKP and FN-Right Middle-Basic-Each model) + (DFKP and FN-Right Index-Basic-Each model) + (DFKP and FN-Right Thumb-Basic-Each model)
- Concatenated Fusion all-InceptionV3: (DFKP and FN-Right-Basic-InceptionV3) + (DFKP and FN-Right-Major-InceptionV3) + (DFKP and FN-Right-Minor-InceptionV3)
- Concatenated Fusion all-ResNet50: (DFKP and FN-Right-Basic-ResNet50) + (DFKP and FN-Right-Major-ResNet50) + (DFKP and FN-Right-Minor-ResNet50)
- Concatenated Fusion all-DenseNet201: (DFKP and FN-Right-Basic-DenseNet201) + (DFKP and FN-Right-Major-DenseNet201) + (DFKP and FN-Right-Minor-DenseNet201)
- Concatenated Fusion all-MobileNetV2: (DFKP and FN-Right-Basic-MobileNetV2) + (DFKP and FN-Right-Major-MobileNetV2) + (DFKP and FN-Right-Minor-MobileNetV2)
- Concatenated Fusion all-DL Model: Concatenated Fusion all-InceptionV3 + Concatenated Fusion all-ResNet50 + Concatenated Fusion all-DenseNet201+ Concatenated Fusion all-MobileNetV2
- Following that, the left hand is then applied to the same concatenated feature vector fusion of (Basic, Major, and Minor) and Fusion all-DL Model.

3.3.4 Rubik's Cube Technique for the Key Generation Stage

The Rubik's Cube-Based Cryptographic Technique is a contemporary cryptographic technique derived from the three-dimensional configuration of the Rubik's Cube. The six faces can be rotated around the three axes (X, Y, Z), resulting in an estimated $10^{98} \times 43$ potential configurations for the cube (3x3x3). This intricate, rotatable structure serves as a useful foundation for dynamic key generation and ensures confusion and diffusion features in encrypted data. It is perfect for implementing the ideas of permutation and substitution in a very complicated cryptographic environment [48], [49]. The transformation process, DFKP, and FN feature-to-key involve several sequential steps, exhibited in Fig. 9. Each step will include a detailed explanation in detail below.

In this work, the Rubik's Cube technique is adopted as the sole component to generate keys utilizing key components combining feature vectors (Basic, Major, Minor, Fingernails) of five fingers on both hands for each user. It is used to compute the secret key (SK), which is then used in a different encryption algorithm. The Rubik's Cube design is able to produce dynamic and unique keys. The concatenated feature vector (DFKP and FN) is normalized to a range of 0–255 (8-bit integer values) using min-max normalization:

$$\text{normalized} = \frac{\text{value} - \min(\text{data})}{\max(\text{data}) - \min(\text{data})} \times 255 \quad (5)$$

This normalization ensures that all feature values are compatible with the subsequent cube-filling step. The cube consists of six faces, each represented as a 16×16 matrix of integer values. The six faces are labeled as: front, back, top, bottom, left, and right. Each face initially contains zeros.

$$6 \text{ faces} \times 16 \times 16 = 1536 \text{ cells} \quad (6)$$

Thus, after performing the process Feature-level Concatenated Fusion in the all-DL model, with features of Basic-knuckle, Major-knuckle features, Minor-knuckle features, and Fingernail knuckle features, as a step in data preparation, normalization, and filling the cube.

To increase randomization, chaotic sequences (like logistic maps) dictate the fill pattern. The logistic map is defined as:

$$x_{n+1} = r \cdot x_n \cdot (1 - x_n) \quad (7)$$

where:

- x_0 is the initial seed (set to 0.1),
- r is the control parameter (set to 3.99, ensuring chaotic behavior),
- n is the number of iterations (set to 50 for scrambling and 20 for move selection).

The generated logistic sequence is used to determine the order and type of scrambling moves applied to the cube.

Afterwards, the resulting quantized binary sequence from the extracted feature vector is divided into equal parts, and these parts are then mapped to the six faces of the Rubik's Cube. Each part is then reshaped into a 16 × 16 two-dimensional matrix, and row and column permutations, where are applied the Logistic-Map used to generate a logistic sequence (0.1, 3.99, number moves), along with various rotations (Cube Rotations: rotating all cube faces to the right, rotating all cube faces to the left, rotating the front face clockwise, and rotating the front face anticlockwise) derived from sub-biometric features, are applied to enhance randomness and diffusion before the final secret key is generated, as a step in the logistic map for scrambling, applying scrambling moves, and permutation of face elements.

$$R_k = reshape(B_k, 16{,}16), \qquad k = 1{,}2, \dots{,}6 \tag{8}$$

$$R_k^{(1)} = P_r R_k P_c \tag{9}$$

$$R_k^{(2)} = Rotate\left(R_k^{(1)}, M\right) \tag{10}$$

Where:
B_k represents the binary segment assigned to face k of the Rubik's Cube.
R_k represents the corresponding 16 × 16 face matrix.
P_r, P_c represent the row and column permute matrices.
M represents the rotation sequence (right, and left, clockwise, and counterclockwise).

The final extract key is hashed using SHA-512, and the resulting bytes are then converted to uppercase letters A–Z for the first 16 bytes of the hash, as a step in extracting the symmetric key cryptography.

$$K = SHA - 512(B_8) \tag{11}$$

$$\text{character} = \text{chr}(65 + (\text{byte} \bmod 26)) \tag{12}$$

Where B_8 is the resulting binary string after all shuffling and rotation, and it is the final secret key.
The resultant transformation constitutes the secret key (SK). In this ECS-DFKPFNRS, we adopted a default key length of 128 bits (SK) to ensure conformance with the specifications of the intended lightweight or standard encryption algorithms, while allowing for adjustments in length based on the requirements of the algorithm that will use the key. The role of the Rubik's Cube in this design is limited to assisting in data encryption.
To analyze the strength and robustness of the random system and the security of the generated keys, a variety of common security metrics have been adopted, including Shannon Entropy (SE), Key Space (KS), Attack Resistance (AR), and Key Derivation Function (KDF). As shown below, these measures are utilized to estimate randomness, the range of important probabilities, and the resistance of the system against statistical and brute-force attacks, as explained below [50],[51],[52],[53], as a step in security analysis step:
Shannon Entropy (SE): It measures the extent of randomness or uncertainty in the distribution of keys. The system is closer to a perfect distribution when $H(s)$ is higher, which makes key prediction more challenging.

$$\log_2\left(p(s_i)\right).p(s_i)\sum_{i=1}^{n} -= H(s) \tag{13}$$

Where $p(s_i)$ signifies the probability of a symbol, s_i, and n is the total number of potential symbols, and $H(s)$ is the entropy value. The optimal entropy value for an 8-bit image is 8, which signifies maximum randomness.

- Key Space (KS): This space characterizes an overall number of possible keys. The bigger key space increases the resistance of the system to the brute force attacks.

$$k_2 = Key\ Space \tag{14}$$

Where K is the bit-length of the key.

- Attack Resistance (AR): It quantifies and describes the dependency between the time required to break the system T, resistance rate R, and key strength k. A higher value of resistance R will be associated with a longer period of time required to launch the attack, which will indicate the strength of the system against statistical and analytical attacks.

$$\frac{k^2}{R} = T \tag{15}$$

with R being the attack rate (operations per second), K being the key length and T being the time to run through all the potential keys.

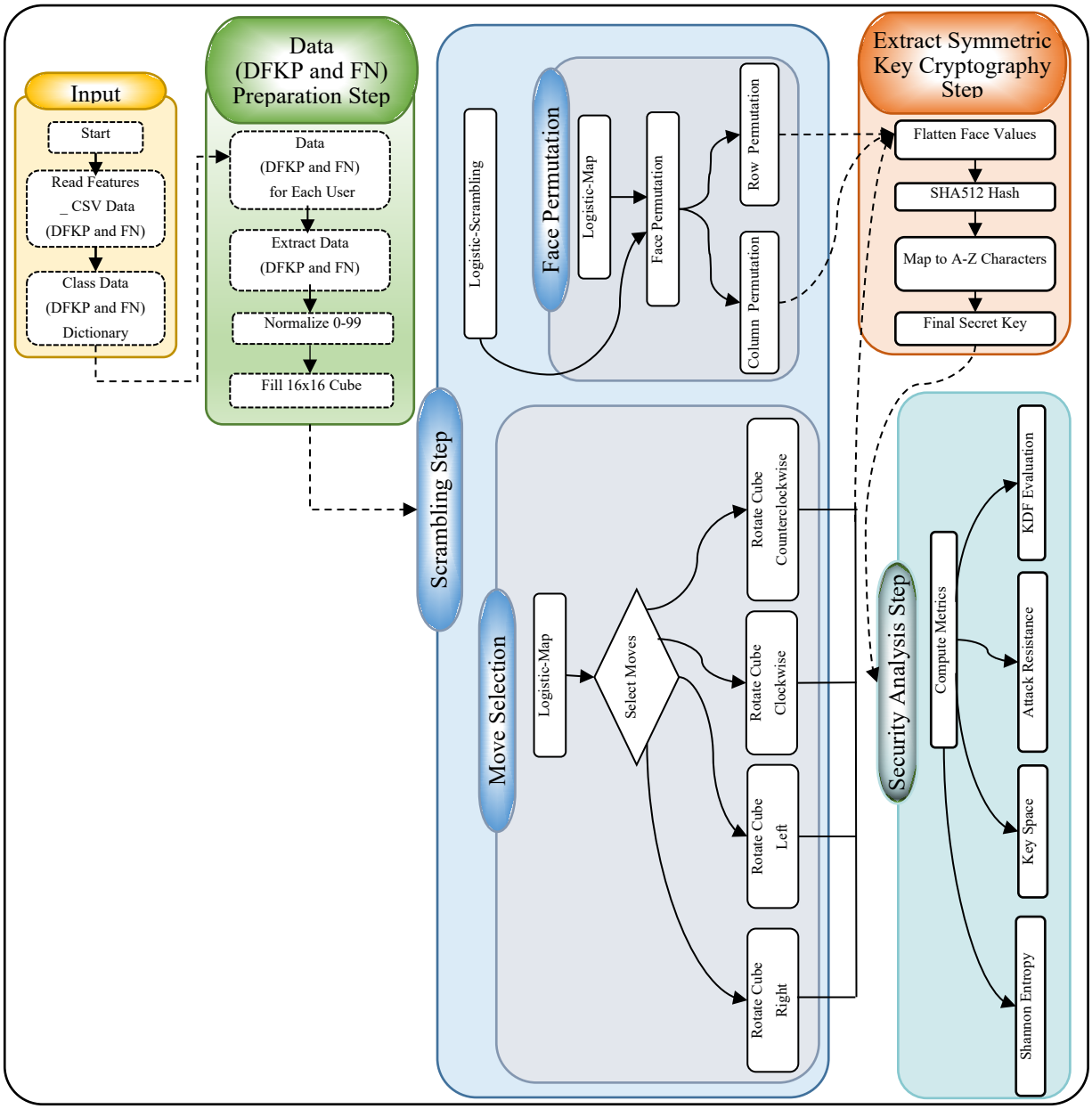


Fig. 9: Step-by-step transformation Process for fused DFKP and FN features into the Key using the Rubik's cube technique

- Key Derivation Function (KDF): It is a multi-layer encryption scheme which creates secure and strong key based on a fundamental secret value P, which may be a password or main key with a salt factor of 5 and a number of repetitions. This purpose increases resistance to statistical and guessing attacks by increasing the complexity of reverse key derivation. It ensures that it produces unique and highly sensitive keys that are difficult to retrieve or predict.

$$KDF(P, S, c, dkLen) = K \tag{16}$$

Where $dkLen$ is the desired key length, c is the number of iterations, S is a random salt, and p is the input password.

3.3.5 Grain-128a Encryption Algorithm Stage

The suggested ECS-DFKPFNRS uses Grain-128a encryption algorithm, which is a lightweight encryption algorithm, which uses linear and nonlinear shift registers (LFSR and NLFSR) to derive a 128-bit keystream to encrypt data effectively. In the proposed system, extracted DFKP and FN features are used to encrypt data. Keys are generated from biometric features and derived via KDF to produce high-entropy random keys. Key Space (KS) and Attack Resistance (AR) are systematic measures that express the size of the possible key space and the system's ability to resist attacks. This integration contributes to enhancing cybersecurity by increasing randomness, expanding the key space, and increasing the system's resistance to statistical and brute-force attacks, while maintaining execution speed and performance in resource-constrained systems. The proposed ECS-DFKPFNRS has been explained in Algorithm 1.

Algorithm for Dorsal FKP & FN: Proposed ECS-DFKPFNRS Framework
Input: Dataset 11k Hands, PolyUHD
Output: User Recognition

BEGIN
Step1: **Read the original hand image from the dataset (11k Hands and PolyUHD)**
1. FOR each row (I) from 1 to 224 pixels (image height)
2. FOR each column (J) from 1 to 224 pixels (image width)

Step2: **Pre-Processing Stage**
3. While the hand image (I) is available, do
4. Segmenting the DFKP and FN
5. Utilize the blurring for the hand image
6. Convert to HSV color space
7. Utilize two of the morphological operations for dilation and erosion
8. Utilize median filtering (MF)
9. Apply the Mediapipe Module
10. **END FOR (J)**
11. **END FOR (I)**
12. Return Final Result
13. Create a database with sub-images of each component.

Step3: **Feature Extraction (FE) Stage**
14. for $p \leftarrow 1$ to 19 do
15. Separate the database into three distinct sets: testing, validation, and training classes for model evaluation and training.
16. Set the network configuration (**Concatenated DL Model for InceptionV3, ResNet50, DenseNet201, MobileNetV2**)
17. Create augmentation images on the validation and training sets
18. Utilizing specific epochs to train.
19. Determine the network weight (W) that achieves the highest validation accuracy and F1_ Score
20. Utilizing the weight (W), extract the features from the pairings (a) and (b) for further analysis.

Step4: **Feature DFKP and FN- to- Key GenerationTransformation Stage**
Input→ Feature Vector (FV)
Output→ Generated Secret Key (K)

Step4.1: **Read and Process CSV Data**
21. FUNCTION Read_CSV(file_path):
22. Open CSV file
23. Initialize class_data as an empty dictionary
24. FOR each row in CSV:
25. Extract class_label (last column)
26. Extract values (all but last column) and convert to numbers
27. IF class_label is NOT in class_data:
28. Initialize class_data[class_label] as an empty list
29. Append values to class_data[class_label]
30. RETURN class_data

Step4.2: **Normalize Data**
31. FUNCTION Normalize_Data(data):
32. Compute min_value and max_value of data
33. FOR each value in data:

34. Normalize using the formula:
normalized_value = (value - min_value) / (max_value - min_value)
35. Convert to an integer in range [0, 255]
36. RETURN normalized_data

Step4.3: **Initialize Cube Structure**

37. FUNCTION Initialize_Cube(size=16):
38. Create a dictionary "cube_faces" with 6 keys: 'front', 'back', 'top', 'bottom', 'left', 'right'
39. Each key contains a 16×16 matrix initialized with zeros
40. RETURN cube_faces

Step4.4: **Fill Cube with Data**

41. FUNCTION Fill_Cube(cube_faces, data):
42. index ← 0
43. FOR each face in cube_faces:
44. FOR i = 0 to 15:
45. FOR j = 0 to 15:
46. IF index < length(data):
47. cube_faces[face][i][j] ← data[index]
48. index ← index + 1
49. ELSE:
50. cube_faces[face][i][j] ← 0 # Fill remaining cells with zero
51. RETURN cube_faces

Step4.5: **Generate Logistic Map Sequence for Scrambling**

52. FUNCTION Logistic_Map (x0, r, n):
53. Initialize sequence[n] with zeros
54. x ← x0
55. FOR i = 0 to n-1:
56. x ← r * x * (1 - x) # Logistic Map Formula
57. sequence[i] ← x
58. RETURN sequence

Step4.6: **Apply Scrambling Moves**

59. FUNCTION Apply_Move(cube_faces, move):
60. SWITCH (move):
61. CASE "rotate_cube_right": Rotate all cube faces right
62. CASE "rotate_cube_left": Rotate all cube faces left
63. CASE "rotate_face_clockwise_front": Rotate front face clockwise
64. CASE "rotate_face_counterclockwise_front": Rotate front face counterclockwise
65. RETURN cube_faces

Step4.7: **Scramble the Cube Using the Logistic Map**

66. FUNCTION Scramble_Cube(cube_faces, num_moves=20):
67. Define possible_moves = ["rotate_cube_right", "rotate_cube_left", "rotate_face_clockwise_front", "rotate_face_counterclockwise_front"]
68. Generate logistic_sequence using Logistic_Map(0.1, 3.99, num_moves)
69. Convert logistic_sequence to integer move indices
70. moves_performed ← empty list
71. FOR each move_index in move_indices:
72. move ← possible_moves[move_index % length(possible_moves)]
73. cube_faces ← Apply_Move(cube_faces, move)
74. Append move to moves_performed
75. RETURN cube _ faces, moves _ performed

Step4.8: **Permute Cube Faces Using Logistic Map**

76. FUNCTION Permute_Face(face):
77. Generate logistic row_sequence and column_sequence
78. Compute row_perm and col_perm by sorting logistic sequences
79. Apply row_perm and col_perm to shuffle face elements
80. RETURN permuted_face

Step4.9: **Extract Key Using SHA-512 Hashing**

81. FUNCTION Extract_Key_From_Cube(cube_faces):
82. key ← ""
83. FOR each face in cube_faces:
84. Flatten face into a 1D list
85. Convert list into bytes
86. Compute the SHA-512 hash

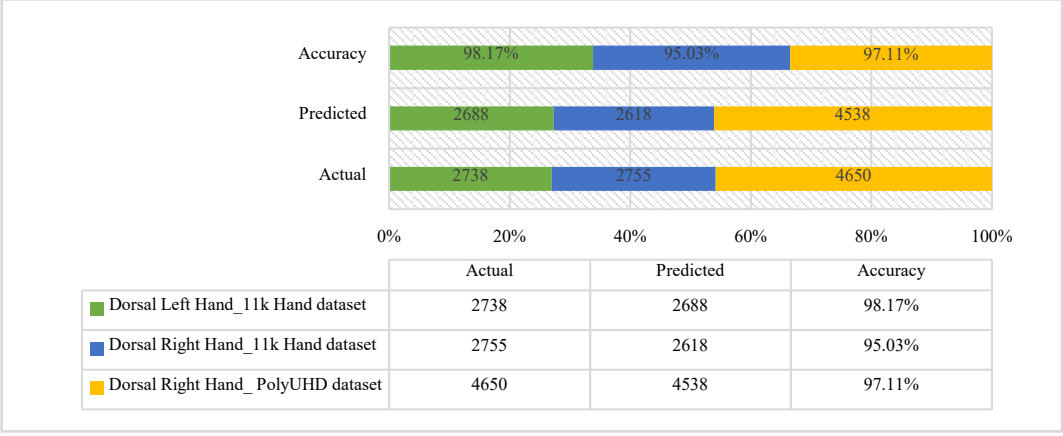
```
        Take the first 16 bytes of the hash
        Convert bytes to uppercase characters A-Z using:
        character = chr(65 + (byte % 26))
        Append characters to key
    RETURN key
Step4.10: Main Execution
    FUNCTION Main ():
    csv_file_path ← "Major_L_Dorsal_M.csv"
    class_data ← Read_CSV(csv_file_path)
    class_keys ← empty dictionary
    FOR each class_label, data in class_data:
        normalized_data ← Normalize_Data(data)
        cube_faces ← Initialize_Cube()
        cube_faces ← Fill_Cube(cube_faces, normalized_data)
        cube_faces, moves ← Scramble_Cube(cube_faces, 20)
        key ← Extract_Key_From_Cube(cube_faces)
          class_keys[class_label] ← key
    RETURN class_keys
Step5: Encryption Grain-128a algorithm stage
    Inputs →128-bit key and 96-bit IV (IV[0] selects authentication)
    Initialize state → Load key into NESR, IV into LFSR, rest bits set (31 ones + 1 zero)
    Warm-Up → Run 256 rounds, feedback output into LFSR/NFSR, no keystream output
    MAC Setup (optional) → Use the first 2w bits to load the accumulator (A) and shift register
    Encryption→ generates keystream bits and XOR with plaintext
    Output→ Ciphertext only, or ciphertext + authentication tag
END
```

4.0 Experimental Analysis and Performance Evaluation

The section covers the analysis and evaluation of the DFKP and FN segmentation method for both hands as well as the feature-level fusion evaluations based on concatenation for four DL models for DFKP and FN, the analysis and evaluation of strong feature key generation that depends Rubik's Cube technique, the analysis and evaluation of a lightweight encryption algorithm that depends Grain-128a algorithm as well as the analysis attack using statistical attack, and the overall evaluation of the final performance of ECS-DFKPFNRS. The experiment results were executed on a computer equipped with an 11th Gen Intel Core i7-1362OH processor, 16 GB of RAM, a 512 GB SSD NVME, and Windows 11, which is adequate for running Python 3.9 and managing data-intensive activities. PyTorch and TensorFlow are two examples of Python libraries that can leverage the benefits of the performance of Nvidia GPUs' in the fields of deep learning and machine learning.

4.1 Analysis and Evaluation of the DFKP and FN Segmentation Method

The subsection focuses on the analysis and evaluation of the preprocessing stage, which employed the segmentation methods to obtain DFKP and FN for both hands. The accuracy of all DFKP and FN samples utilizing the Hands landmark model on the 11k Hands and Poly UHD datasets is demonstrated in Fig. 10.



	Actual	Predicted	Accuracy
Dorsal Left Hand_11k Hand dataset	2738	2688	98.17%
Dorsal Right Hand_11k Hand dataset	2755	2618	95.03%
Dorsal Right Hand_ PolyUHD dataset	4650	4538	97.11%

Fig. 10: The accuracy for all samples, DFKP and FN (Left & Right) for both 11K Hands and PolyUHD datasets

According to our results, both the left-DFKP and left-FN in the 11k Hands dataset produced superior and more discriminating results than those produced by the right-DFKP and right-FN in the 11k Hands and PolyUHD datasets. The highest level of accuracy was illustrated by the actual and predicted values for all samples in the 11K Hands and PolyUHD datasets, as shown in Fig. 10.

4.2 Analysis and Evaluation of the Concatenated Deep Learning Models

This subsection presents the analysis and evaluation of the concatenated DL models in our experiment, where DFKP and FN features were extracted using four pre-trained DL models (InceptionV3, ResNet50, DenseNet201, MobileNetV2). By applying seven concatenated with the set of layers for each architecture model. The DFKP and FN performance was evaluated using many metrics, including F1_score, accuracy (ACC), precision (PPV), precision (PPV), recall (sensitivity), and specificity (SPC). These metrics were obtained using the equations (17-20) [54], [55], [56], [57].

$$Accuracy(ACC) = \frac{T_p+T_n}{T_p+T_n+F_p+F_n} \quad (17)$$

$$Precision(PPV) = \frac{T_p}{T_p+F_p} \quad (18)$$

$$Recall(Sensitivity) = \frac{T_p}{T_p+F_n} \quad (19)$$

$$F1\_Score = \frac{2*T_p}{2*T_p+F_p+F_n} \quad (20)$$

The terms TP, TN, FP, and FN refer to True Positive, True Negative, False Positive, and False Negative, respectively.

The initial dataset can be divided into three distinct classes: training, validation, and testing. At the end of each training cycle, the suggested concatenated DL models are evaluated against the validation set after being trained with training data during the duration of the training cycle. Following that, the DL models are evaluated by making use of the testing dataset, and evaluation metrics are utilized in order to measure the performance of the concatenated DL models. Tables 2 and 3 show the results of classification metrics for multi-classification of the concatenated DL models for both DFKP and FN (right and left). As well as Fig. 11 shows the F1-score and accuracy for each DFKP and FN (right and left) utilizing the concatenated DL models.

While Table 4 shows the summarized classification results of the concatenated fusion all-DL Model for every key component DFKP and FN (right and left), and Fig. 12 shows the results of Precision, Recall, F1-Score, and accuracy for every key component DFKP and FN (right and left) utilizing the concatenated DL Model.

Table 2: Multi-classification to evaluate the concatenated DL model for 11k Hands & Poly UHD datasets (right dorsal)

	Concatenated DL model			
	11k Hands (Right Dorsal)		**Poly UHD (Right Dorsal)**	
Sub-Images	**F1 Score**	**Accuracy**	**F1 Score**	**Accuracy**
BRDI	99.07%	99.08%	98.99%	98.99%
BRDL	99.07%	99.08%	98.99%	98.99%
BRDM	99.04%	99.04%	99.23%	99.26%
BRDR	99.11%	99.11%	98.99%	99.00%
BRDTH	99.21%	99.26%	99.06%	99.09%
FNRDI	99.18%	99.21%	99.12%	99.14%
FNRDL	99.20%	99.24%	99.09%	99.14%
FNRDM	99.20%	99.26%	99.04%	99.07%
FNRDR	99.34%	99.41%	99.17%	99.23%
FNRDTH	99.11%	99.13%	99.25%	99.35%
MRDI	99.26%	99.29%	99.24%	99.27%
MRDL	99.28%	99.33%	99.30%	99.36%
MRDM	99.39%	99.49%	99.42%	99.51%
MRDR	99.50%	99.55%	99.30%	99.39%
MRDTH	99.31%	99.43%	99.23%	99.29%
MiRDI	99.22%	99.27%	99.15%	99.20%
MiRDL	99.22%	99.29%	99.33%	99.42%
MiRDR	99.28%	99.33%	99.26%	99.29%
MiRDM	99.18%	99.24%	99.35%	99.46%

Table 3: Multi-classification to evaluate the Concatenated DL model for the 11k Hands dataset (left dorsal)

	Concatenated DL Model (Left Dorsal)	
	11k Hands	
Sub-Images	**F1 Score**	**Accuracy**
BLDI	99.08%	99.08%
BLDL	99.19%	99.28%
BLDM	99.04%	99.07%
BLDR	99.20%	99.25%
BLDTH	99.29%	99.40%
FNLDI	99.25%	99.30%
FNLDL	99.28%	99.34%
FNLDM	99.23%	99.26%
FNLDR	99.39%	99.48%
FNLDTH	99.25%	99.33%
MLDI	99.39%	99.47%
MLDL	99.29%	99.40%
MLDM	99.59%	99.68%
MLDR	99.52%	99.62%
MLDTH	99.34%	99.45%
MiLDI	99.39%	99.44%
MiLDL	99.36%	99.54%
MiLDR	99.45%	99.62%
MiLDM	99.27%	99.37%

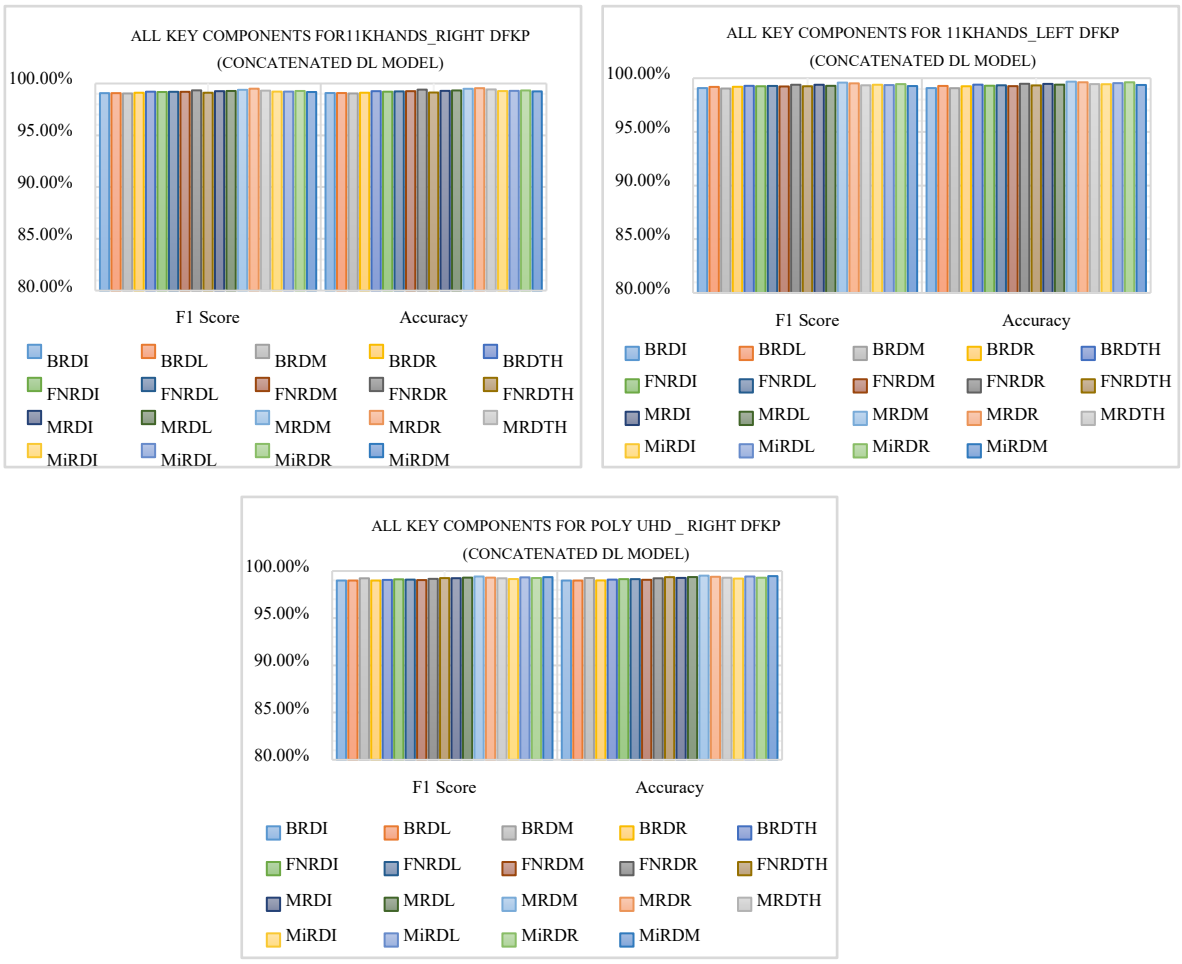


Fig.11: The F1-score and accuracy for each Left DFKP and Right DFKP based on the Concatenated DL Model for the 11K Hands and PolyUHD datasets

Table 4: The accuracy and F1_Score (summarized _ classification_ testing) for Concatenated DL model (11k Hands (left dorsal &right dorsal) and PolyUHD (right dorsal))

Summarized _ Classification for Concatenated DL model				
Testing _ 11k Hands (Right Dorsal)				
	Precision	Recall	F1_Score	Accuracy
Micro	99.10%	99.10%	99.10%	99.10%
Macro	99.16%	99.16%	99.46%	99.16%
Weighted	99.1%	99.16%	99.46%	99.16%
Testing _ 11k Hands (Left Dorsal)				
	Precision	Recall	F1_Score	Accuracy
Micro	99.27%	99.27%	99.27%	99.27%
Macro	99.57%	99.27%	99.78%	99.27%
Weighted	99.57%	99.27%	99.78%	99.27%
Testing _PolyUHD (Right Dorsal)				
	Precision	Recall	F1_Score	Accuracy
Micro	100%	99.54%	99.54%	99.54%
Macro	99.54%	99.54%	99.74%	99.54%
Weighted	99.54%	99.54%	99.64%	99.54%

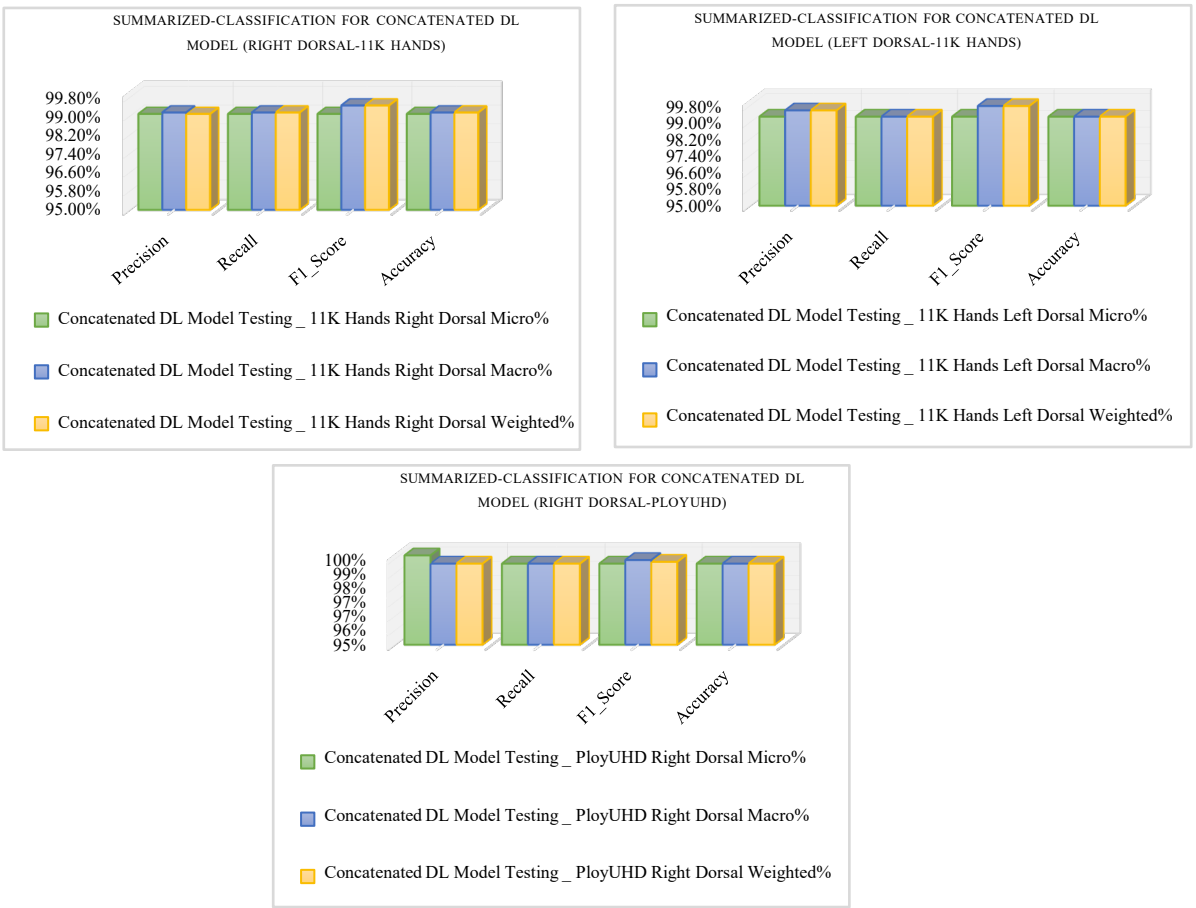
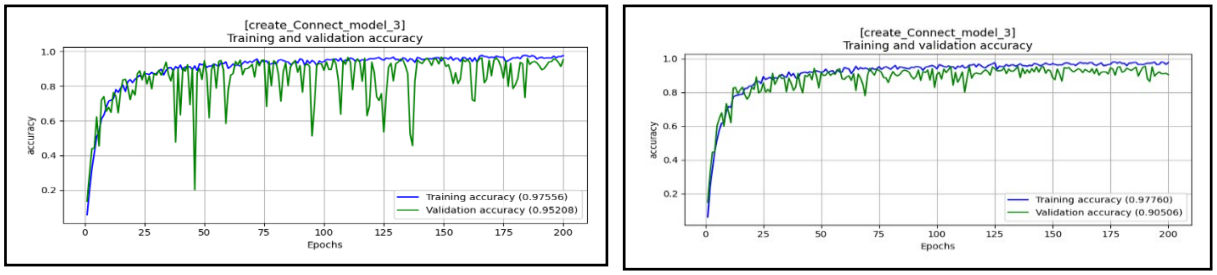
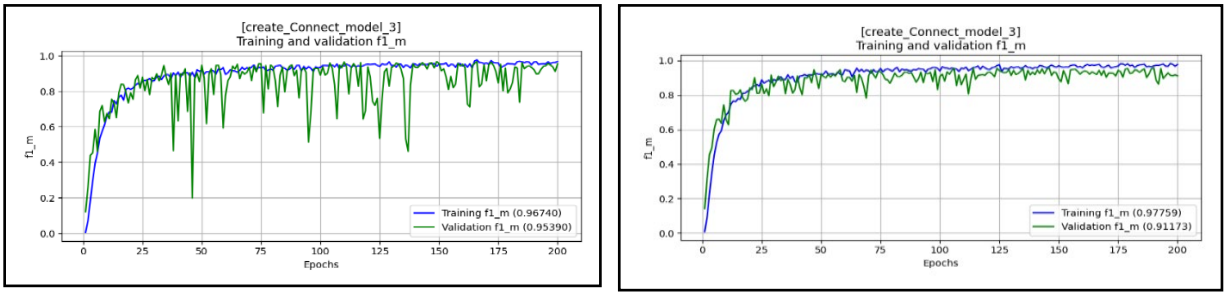


Fig.12: The summarized-classification for the right (DFKP and FN), and left (DFKP and FN) based on the Concatenated DL Model for the 11K Hands and PolyUHD datasets

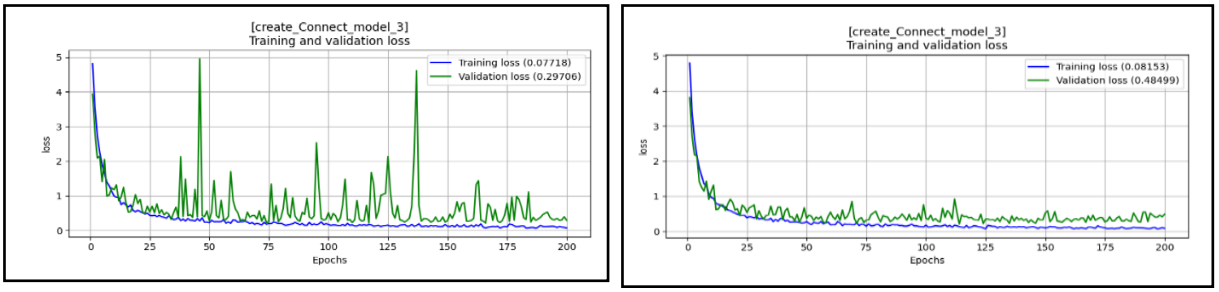
Utilizing the 11k hands dataset, the Concatenated DL Model was trained to classify right DFKP and FN, as well as left DFKP and FN, using 2755 and 2738 images, respectively. Additionally, the PolyU HD dataset was used to employ 4650 images for the right DFKP and FN. These models were first trained over 200 epochs, utilizing images that were 224 pixels wide and 224 pixels high as their input. Fig.13 provide the results of the training and validation sets, which include the accuracies, F1_score, loss, recall, and precision that were achieved. Fig. 14 shows the effectiveness of the evaluation of the concatenated DL model achieved through the utilization of the confusion matrix (CM).



(a)



(b)



(c)

Fig.13: The samples results for concatenated DL model (right (DFKP and FN), and left (DFKP and FN)) for the 11K Hands and PolyUHD datasets (a) accuracy for training & validation, (b) F1-Score for training & validation, and (c) Loss for training & validation

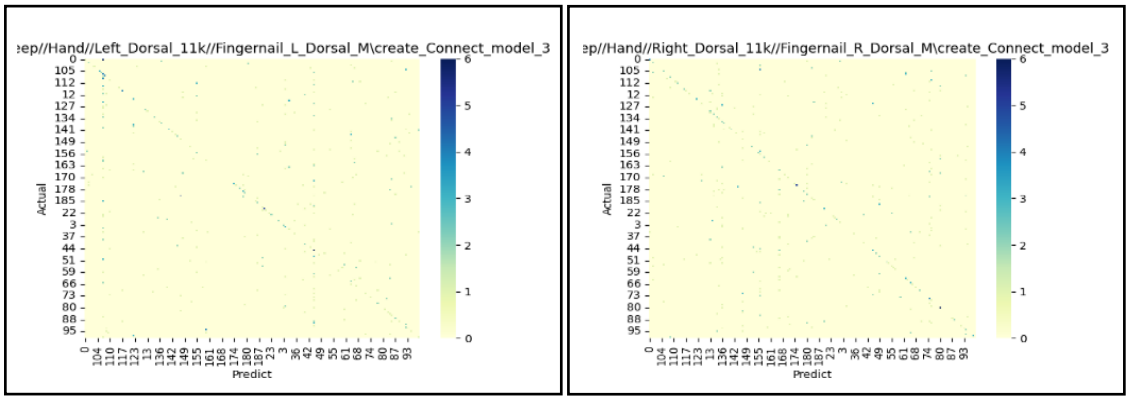


Fig.14: Confusion matrix for some samples, left (DFKP and FN-11K Hand dataset), and right (DFKP and FN-PloyUHD dataset) using concatenated DL model

4.3 Analysis and Evaluation of the Computational Complexity of the Individual and Proposed Concatenated Deep Learning Models

This subsection will analyze and evaluate the computational complexity of the models utilized in the proposed system. The analysis and comparison between the individual and the concatenated DL models were carried out depending on the following three parameters: Total number of parameters, Trainable parameters, and Non-trainable parameters. These parameters are significant in determining the size, computational load, and learning ability of the model. This analysis also contributes to understanding the relationship between model complexity and recognition performance, and therefore to the choice of a feature integration process for the proposed system, which involves adding features extracted from different DL models. Table 5 details the three metrics used for the individual model and the proposed concatenated DL model.

Table 5: Computational Complexity Metrics of the Individual and Proposed Concatenated DL Models

Model	Total parameters	Trainable Parameters	Non-Trainable Parameters
InceptionV3	22,948,829	13,958,717	8,990,112
ResNet50	63,728,912	63,675,792	53,120
DenseNet201	26,988,797	8,654,781	18,334,016
MobileNetV2	3,119,056	2,604,240	514,816
InceptionV3 + ResNet50 + DenseNet201	51,221,995	6,315,831	44,906,164
InceptionV3 + ResNet50 + DenseNet201 + MobileNetV2 (Concatenated DL)	52,123,327	1,729,307	50,394,020

4.4 Analysis and Evaluation of the Ablation Study of the Individual and Proposed Concatenated Deep Learning Models

This subsection will present an analysis and evaluation of the ablation study of individual and concatenated DL models performed in the experiments. Table 6 shows the results of the ablation study conducted on the individual DL models and the suggested concatenated model. All four proposed DL models were evaluated for their recognition performance, and the concatenation of DL models performed the best.

Table 6: Results of the Ablation Study for four DL models individually and concatenated

	11k Hands (Right Dorsal)			
Model	**Precision**	**Recall**	**F1_Score**	**Accuracy**
InceptionV3	93.51%	91.67%	91.84%	91.67%
ResNet50	96.23%	96.13%	96.53%	96.13%
DenseNet201	94.42%	92.97%	92.94%	92.97%
MobileNetV2	95.74%	95.74%	95.74%	95.74%
InceptionV3 + ResNet50 + DenseNet201	86.60%	83.12%	84.77%	83.97%
InceptionV3 + ResNet50 + DenseNet201 + MobileNetV2 (Concatenated DL)	99.1%	99.16%	99.46%	99.16%
	11k Hands (Left Dorsal)			
	Precision	**Recall**	**F1_Score**	**Accuracy**
InceptionV3	95.81%	95.44%	95.39%	95.44%
ResNet50	100%	99.99%	99.98%	99.99%
DenseNet201	90.11%	87.85%	87.83%	87.85%
MobileNetV2	99.03%	98.97%	98.97%	98.97%
InceptionV3 + ResNet50 + DenseNet201	90.04%	86.52%	88.18%	87.75%
InceptionV3 + ResNet50 + DenseNet201 + MobileNetV2 (Concatenated DL)	99.57%	99.27%	99.78%	99.27%
	PolyUHD (Right Dorsal)			
	Precision	**Recall**	**F1_Score**	**Accuracy**
InceptionV3	96.64%	95.44%	95.59%	95.44%
ResNet50	95.64%	95.54%	95.59%	95.54%
DenseNet201	94.25%	93.48%	93.51%	93.48%
MobileNetV2	93.99%	93.99%	93.99%	93.99%
DenseNet201 + InceptionV3 + ResNet50	88.60%	85.74%	87.1%	87.1%
DenseNet201 + InceptionV3 + ResNet50 + MobileNetV2 (Concatenated DL)	99.54%	99.54%	99.64%	99.54%

4.5 Analysis and Evaluation of the Rubik's Cube Technique for the Key Generation

This subsection of the analysis and evaluation process focuses on the key generation technique utilizing the 3D Rubik's Cube technique, a crucial stage in the proposed system aimed at attaining a high level of randomness and mathematical complexity in the generation of cryptographic keys. The concept involves depicting the extracted biometric features, including DFKP and FN, within a virtual six-faced Rubik's Cube structure. The data has been divided into a three-dimensional matrix with dimensions of 16×16×6. The cube is subsequently rotated, scrambled, and handled by row and column permutations based on sequences given by the Logistic Maps. These processes generate a nonlinear, complicated state which is used in building a unique, random, high-entropy key. Fig. 15 shows a 3D view of the state of the cube prior to the scrambling process and after the scrambling process.

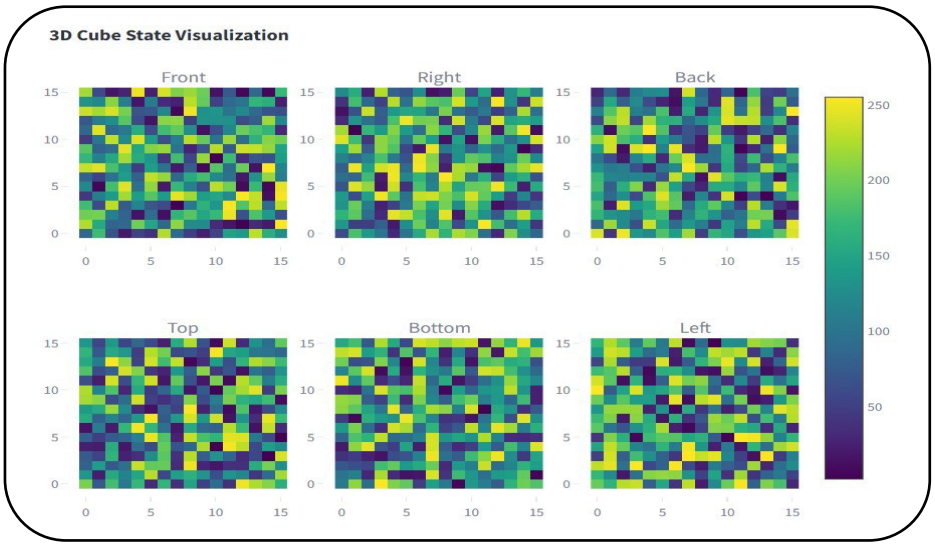


Fig.15: The state visualization of the 3D Rubik's Cube technique

These are represented as two-dimensional arrays (16x16) of numerical values based on biometric data, initially, the six sides of the cube (front, right, back, top, bottom, and left). In order to improve the performance of the cube, it is recommended to use SHA-512, 20 shuffling moves, random seed, and 50 iterations. The values of the faces redistributing due to the mathematically controlled random rotation is significantly different. The statistical outcomes revealed a total of 1536 elements and 255 distinct values, a mean value of 127.34 with a standard deviation of 73.05, indicating uniform as well as highly random distribution over all facets. This indicates that the Rubiks Cube algorithm provides a large key space and a large entropy value, which makes the generation process more resistant to statistical and brute force attacks and the execution time is low enough to handle applications requiring real time in biometrics. Table 7 shows the quantitative analysis of the performance evaluation of the key generation using Rubik cube when applied to the 11K Hands and PolyU HD datasets. The results show that average processing time was 0.0096 seconds using 11K Hands dataset and 0.1556 seconds using PolyU HD dataset which is very efficient in creating speed although the number of classes increased to more than 500 classes. The close values of entropy (3.913-3.914) depict that the generated keys are highly randomized with a high degree of consistency in generating non repeating values. The key space number (10^{38}) and brute-force resistance (512 bits) imply that the system is highly resilient to prediction and mathematical attacks. A KDF score of 32 signifies that the system produces sub keys with high effectiveness and stable derivation. Collision resistance (1) and side-channel attack resistance (0.0039) indicate that key generation provides high security with minimal information loss during execution. The system, with a fixed key length of 128 bits, is compatible with lightweight encryption algorithms like Grain-128a.

The performance analysis and evaluation results of the key generation process utilizing Rubik's Cube technology are shown in Fig. 16 for some metrics (Entropy Distribution, Key Length, Brute Force Resistance, and Processing Time) utilizing 11k Hands datasets. Fig. 17 depicts the statistical analysis of the relationships among the performance indicators for the Rubik's Cube-based key generation stage. The initial image illustrates the correlation matrix between processing time and entropy, which gives a value of 0.0114 that indicates a negligible negative correlation, implying that an increase in execution time has no substantial impact on randomness levels. This indicates that there is a constant high entropy (3.9) of the system irrespective of the processing time, therefore, validating the stable behaviour of the underlying computational architecture in generating random keys with no time delay.

Table 7: Results of the analysis and performance evaluation for the key generation process by employing the Rubik's Cube technique

	11 K Hands Dataset	PolyU HD Dataset
Total Classes	189	501
Avg. Processing Time	0.0096s	0.1556s
Avg. Entropy	3.914	3.913
Avg. Key Space	10^{38}x 3.4	10^{38}x 1.7946
Avg. Brute Force-Bits	512	512
Avg. KDF Evaluation	32	32
Avg. Collision Resistance	1	1
Avg. Channel Attack Resistance	0.0039269	0.0039266
Avg. Key Length	128	128

The second image depicts a 3D performance space which combines processing time, entropy and key length, and a color gradient that shows resistance to brute-force attacks (brute-force bits). The data sets are concentrated on the fixed block of key 128 bits and high entropy values with a high level of security, which means that the system is in the ideal range that balances the processing performance and security level. While the third and fourth images present an analysis of the entropy distribution and the processing time distribution. The entropy plot indicates that the majority of samples have values within the range of 3.9–3.95, with a few outliers, which indicates homogenous randomness and minimal variation between keys. The distribution of processing times indicates that the majority of values are under 0.02 seconds, with a few outliers, demonstrating the stable computational efficiency and high execution speed of the generation technique.

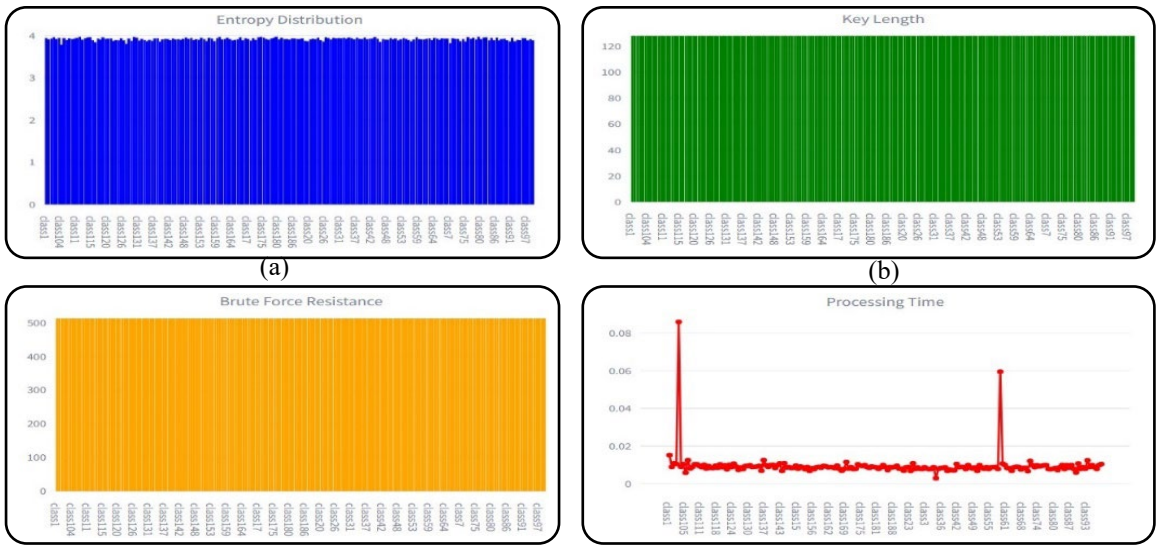
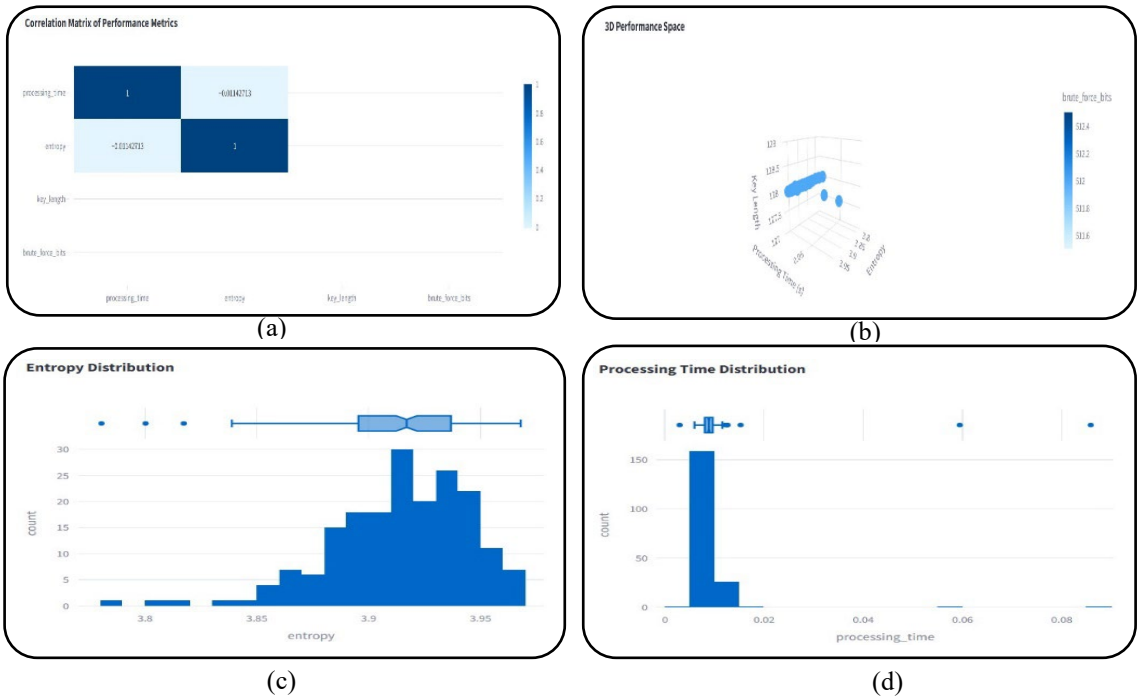


Fig.16: The results from the performance analysis for the 11k Hands dataset (a) Entropy Distribution, (b) Key Length, (c) Brute Force Resistance, and (d) Processing Time

The final security rating of the keys that were developed utilizing a 3D Rubik's Cube technique during the key generation stage is displayed in Table 8. These keys were generated by concatenating fusion features of the DFKP and FN. All generated keys from the Hands 11 and PolyU HD datasets were analyzed for their overall security level and the number of keys classified as High Security Keys or Low Security Keys. The results indicate that the average security score was 99.3 out of 100 in both datasets, indicating that the generated keys exhibit a very high level of resistance to statistical and brute force attacks. Additionally, the values demonstrate that all of the keys (189/189 in Hands 11 and 501/501 in PolyU HD) have been classified as High Security Keys, and that there were no low security keys (0/189 and 0/501) were found.



(a) (b) (c) (d)

Fig.17: Advanced visualizations for the 11K Hands dataset (a) Correlation Metrix of performance metric, (b) 3D Performance Space, (c) Entropy Distribution, and (d) Processing Time Distribution

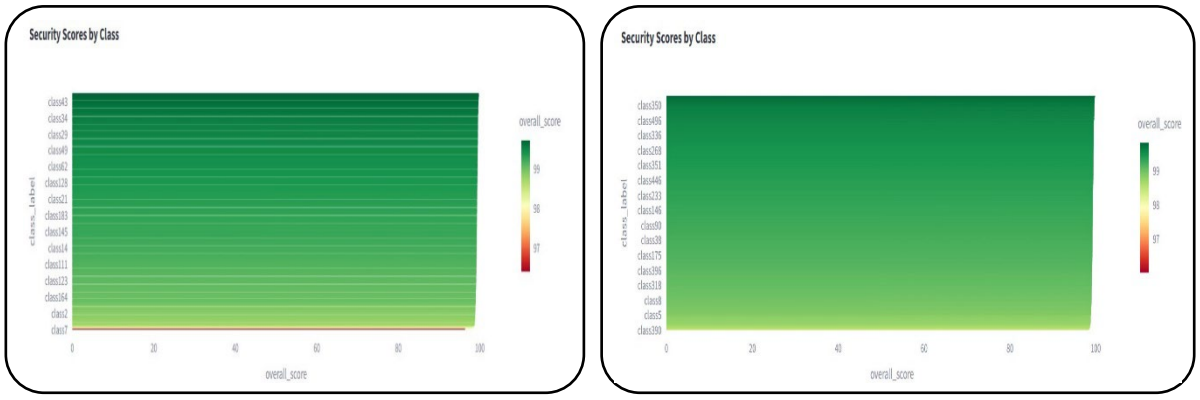
Table 8: Security analysis using the Rubik's Cube technique for all of the key concatenated fusion features of DFKP

	Average Security Score	High Security Keys	Low Security Keys
11 K Hands	99.3/100	189/189	0/189
PolyU HD	99.3/100	501/501	0/501

After concatenating fusion features of the DFKP and FN within the Rubik's Cube-based key generation technique, the two images that are displayed in Fig.18 illustrate the distribution of security scores (security scores) across all classes that were derived from the 11K Hands and PolyU HD datasets. For the 11K Hands dataset, the initial image indicates that all classes attained high security ratings between 97 and 99+, exhibiting near-perfect stability in security performance and the absence of any low-security class. In a similar manner, the second image, which belongs to the PolyU HD dataset, exhibits a uniform distribution of security scores across all 501 classes, indicating the reliability and stability of the generating technique across various samples.

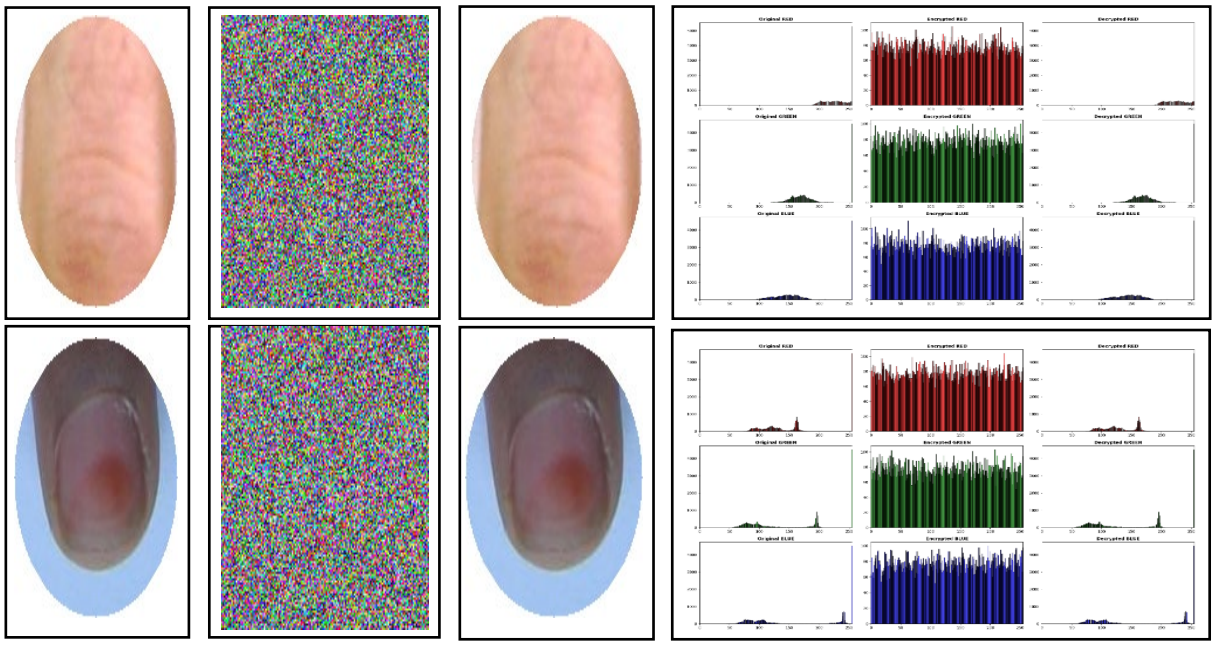
4.6 Analysis and Evaluation of the Grain-128a Encryption Algorithm

This subsection presents the performance analysis and evaluation results of the lightweight Grain-128a encryption algorithm applied to DFKP and FN images of both hands, after key generation using the Rubik's Cube technique to enhance security. The algorithm employs a stream cipher that utilizes a 128-bit key and a 96-bit initialization vector (IV) to generate a pseudorandom bit stream (keystream) utilized in the XOR operation with the image data. The findings revealed that the encrypted images had totally lost their original visual structure, but the decrypted images had perfectly restored their actual forms owing to inversion property of XOR. The experiments were on the original, encrypted and decrypted images and their corresponding histograms. The encrypted image showed uniform distribution in the values (0-255) indicating a high level of randomness and dispersion, but the histograms of the decrypted images matched with the original images, which indicated that Grain-128a algorithm was efficient and accurate in the encryption and decryption process, as shown in Fig. 19.



(a) (b)

Fig.18: The security score analysis of some sample classes for two datasets (a) 11 K hands, and (b) PolyU HD



(a) (b) (c) (d)

Fig. 19: The results of some samples, DFNP for both 11 K hands and PolyU HD datasets (a) original image, (b) encrypted image, (c) decrypted image, and (d) histograms for original, encrypted, and decrypted images

A variety of analytical tests were employed to evaluate the Grain-128a algorithm's performance in the cybersecurity field. The algorithm's sensitivity to slight input changes and resistance to differential attacks were evaluated using differential attack analysis (NPCR and UACI). Using equations (21 to 22), NPCR and UACI are computed [58], [59] ,[60].

$$\mathrm{UACI} = \frac{1}{\mathrm{M}\times\mathrm{N}} \sum_{\mathrm{x}=1}^{\mathrm{M}} \sum_{\mathrm{y}=1}^{\mathrm{N}} \frac{\mathrm{CI}_1(\mathrm{x},\mathrm{y}) - \mathrm{CI}_2(\mathrm{x},\mathrm{y})}{255} \times 100\% \tag{21}$$

$$NPCR = \frac{1}{M\times N} \sum_{x=1}^{M} \sum_{y=1}^{N} D(x,y) \times 100\% \tag{22}$$

The ciphered image of the plain image is denoted by $CI_1(x, y)$, while the ciphered image that matches the plain image after a slight modification is denoted by $CI_2(x, y)$. The value of $D(x, y)$ is equal to zero. If $CI_1(x, y)$ is equal to $CI_2(x, y)$, then the condition follows. In that case, $D(x, y)$equals 1.

Immediately after, the level of randomness present in the ciphertexts was evaluated by randomness analysis employing entropy selection, as shown in equation (23) [60], [61].

$$H(m) = -\sum_{i=0}^{2^N-1} p(m_i) \log_2 p(m_i) \quad (23)$$

In this context, “m” denotes the information source, "N" denotes the total number of bits that represent the symbol m_i, $p(m_i)$ denotes the probability that the symbol m_i is present, and the best value of information entropy is close to 8.
Next, the quality of the encrypted images was evaluated, and the differences from the original image were measured using the Structural Similarity Index (SSIM). equation (24) is utilized to compute it [62], [63], [64].

$$SSIM = \frac{(2\mu_O\mu_D+F_1)(2\sigma_{OD}+F_2)}{(\mu_{O^2}+\mu_{D^2})(\sigma_{O^2}+\sigma_{D^2}+F_2)} \quad (24)$$

With $F_1 = (0.01 \times 255)^2$, $F_2 = (0.01 \times 255)^2$, the variances, means, and covariance of the original and deciphered images are represented by σ_O, σ_D, μ_O, μ_D, $\sigma_{OD,}$ respectively.
The NPCR, UACI, Entropy, SSIM, and performance (Time) results for selected samples of concatenated fusion features from DFKP and FN utilizing the concatenated DL model, along with the averages for all these measures, are shown for all DFKP and FN feature fusion samples in the two datasets in Tables 9 and 10.

Table 9: Results of the set of selected feature concatenated fusion DFKP and FN samples in terms of NPCR, UACI, Entropy, SSIM, and performance (time)

		Keys Concatenated Fusion (11K Hands Dataset)		**Keys Concatenated Fusion (PolyUHD Dataset)**
		Fusion-Right Dorsal	Fusion-Left Dorsal	Fusion-Right Dorsal
Security Metrics	**NPCR**	99.64%	99.64%	99.62%
	UACI	30.32%	30.36%	30.32%
	Entropy	799.72%	799.74%	799.69%
	SSIM	0.0210	0.0216	0.0192
Perfor-mance Results	**Encryption Time (ms)**	23076.04ms	139450.15ms	28545.32ms
	Decryption Time (ms)	75387.22ms	140357.60ms	84410.57ms

Table 10: Results of security metrics and performance (Time) for each sample 11K Hands and PolyUHD datasets

Datasets	**Avg. NPCR**	**Avg. UACI**	**Avg. Entropy**	**Avg. SSIM**	**Avg. Time**
11K Hands	99.95%	30.23%	799.70%	0.0209	148.02s
PolyU HD	99.63%	30.21%	799.69%	0.0215	188.09s

As depicted in Fig. 20, the final comprehensive analysis for NPCR, UACI, entropy, and processing time was conducted utilizing the 11k Hands and PloyU HD datasets.

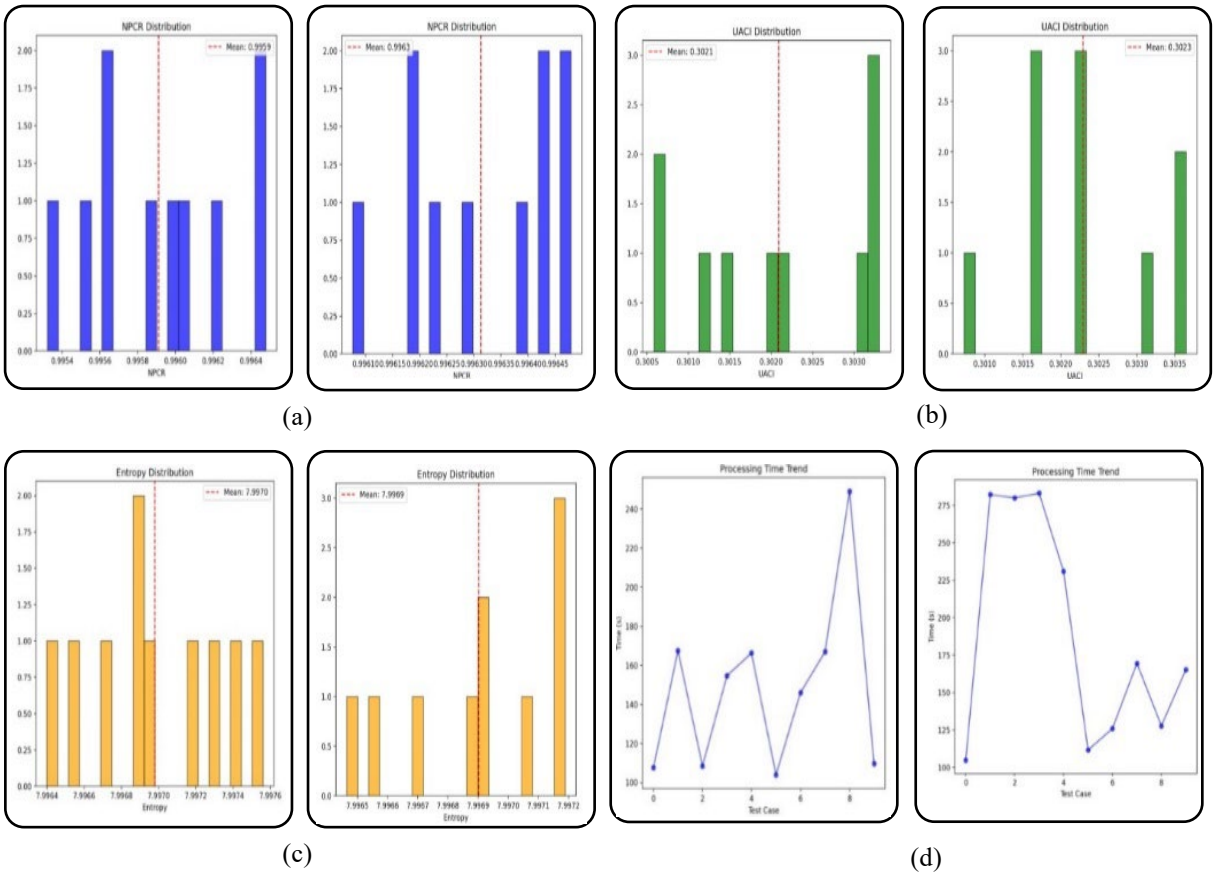


Fig. 20: The final comprehensive analysis for both the 11k Hands and PloyUHD datasets (a) NPCR, (b) UACI, (c) entropy, and (d) processing time

In the final analysis, the NIST Randomness Tests package was utilized in order to provide a comprehensive and internationally recognized evaluation of the quality of the encrypted keys and data [65], [66]. This evaluation is presented in Table 11.

4.7 Analysis and Evaluation of the Statistical Attack

Correlation analysis (CA) is a fundamental statistical test employed for evaluating the robustness of cryptographic systems against statistical attacks. It measures the degree of diagonal, vertical, and horizontal connection between adjacent pixels. Equation (25) utilize covariance and variance to derive the correlation coefficient (CC) [67], [68].

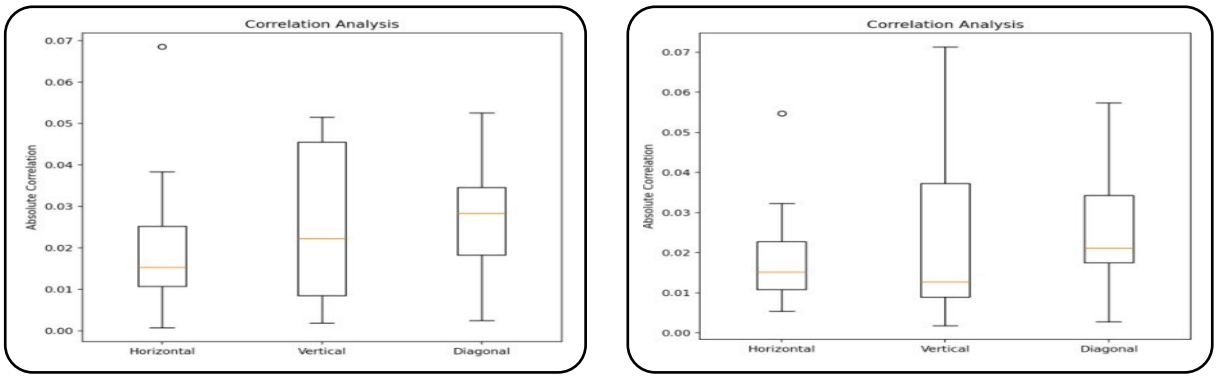
$$\begin{aligned} \mathrm{Corr} &= \frac{\mathrm{Con}(I_1, I_2)}{\sqrt{F(I_1)F(I_2)}} \\ \mathrm{Cov}(I_1, I_2) &= E([I_1 - E(I_1)][I_2 - E(I_2)]) \\ E(I) &= \frac{1}{M}\sum_{x=1}^{M} I_x \\ F(I) &= \frac{1}{M}\sum_{x=1}^{M} [I_x - E(I)]^2 \end{aligned} \tag{25}$$

Correlation analysis and entropy distribution analysis are two of the most important statistical tools used to evaluate the efficiency of image encryption algorithms. Low pixel correlation values after encryption mean that the statistical relationship between adjacent pixels has been completely broken, preventing any visual information from being inferred from the encrypted image. High entropy values, close to the ideal value (8.0 for 8-bit grayscale images), are a strong indicator of statistical randomness of the pixel distribution—that is, the resulting image

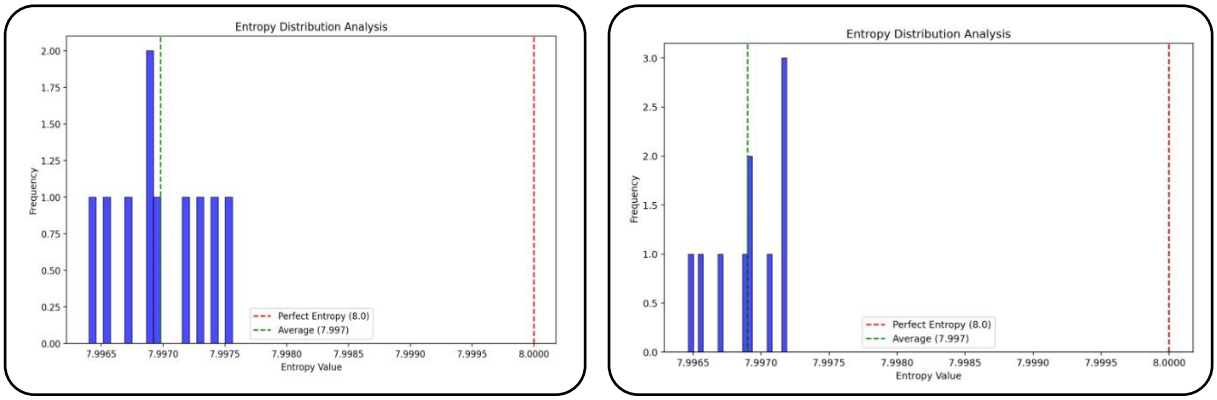
contains no predictable patterns. Accordingly, these two analyses are used to confirm the success of the Grain-128a encryption algorithm in achieving high randomness and resistance to statistical attacks. Fig. 21 shows the results of these two analyses on the 11K Hands and PolyU HD datasets to evaluate the performance of the proposed system. In the correlation analysis, the results show that the absolute correlation coefficients in the horizontal, vertical, and diagonal directions are very low, with medians ranging only between 0.015 and 0.025. This drastic reduction affirms the capability of Grain-128a to destroy the statistical associations between neighboring pixels to create encrypted images with the pattern-less, strong random distribution. It is observed that there is a little bit more dispersion in the vertical direction, which demonstrates a wider range of values having no impact on the overall correlation. These findings continued to hold this trend in both algorithms, and showed the consistency in the performance of the algorithm when tested on various datasets. Analysis of the entropy distribution, which was previously noted in this paper as being based on the Shannon Entropy Equation, showed that the average entropy for both algorithms was approximately 7.997, a value very close to the ideal value of 8.0, within a narrow range (7.9965–7.9975). These results confirm that the Grain-128a algorithm achieves a near-perfect pixel distribution, reflecting very high randomness and stable performance even with high-resolution images (such as PolyU HD).

Table 11: NIST statistical test for the encryption DFKP

Test Item	P-Value	Succeeded
Block Frequency Test	0.174529	TRUE
Run Test	0.011081	TRUE
Non-overlapping Template Matching Test	0.098237	TRUE
Universal Statistical Test	0.123446	TRUE
Random Excursion Test	3.769888	TRUE
Random Excursion Test	2.67635	TRUE
Random Excursion Test	4.01466	TRUE
Random Excursion Test	1.1875	TRUE
Random Excursion Test	3.75	TRUE
Random Excursion Test	4.972222	TRUE
Random Excursion Test	4.75535	TRUE
Random Excursion Test	9.674302	TRUE
Random Excursion Variant Test	80	TRUE
Random Excursion Variant Test	93	TRUE
Random Excursion Variant Test	103	TRUE
Random Excursion Variant Test	90	TRUE
Random Excursion Variant Test	72	TRUE
Random Excursion Variant Test	64	TRUE
Random Excursion Variant Test	65	TRUE
Random Excursion Variant Test	59	TRUE
Random Excursion Variant Test	54	TRUE
Random Excursion Variant Test	58	TRUE
Random Excursion Variant Test	46	TRUE
Random Excursion Variant Test	48	TRUE
Random Excursion Variant Test	47	TRUE
Random Excursion Variant Test	37	TRUE
Random Excursion Variant Test	33	TRUE
Random Excursion Variant Test	43	TRUE
Random Excursion Variant Test	47	TRUE
Random Excursion Variant Test	60	TRUE



(a)



(b)

Fig. 21: The results of the statistical attack for both 11K Hands and PolyU HD Hands datasets (a) correlation analysis, and (b)Entropy distribution analysis

5.0 DISCUSSION

For analysis and evaluation, ECS-DFKPFNRS performance on both 11,076K Hands and the 'Poly U HD' datasets, the proposed system achieves high performance for both user recognition and obtains high security by leveraging optimal deep features to generate strong keys relying on DFKP and FN for both hands. This system integration between biometric data and an encryption algorithm through a deep concatenated fusion of features, in addition to analyzing the system against statistical attacks, to embedded into an extensive cybersecurity system. The DFKP and FN segmentation for right and left hands on the actual and predicted images were compared, as demonstrated in Figure 9, to evaluate the accuracy of DFKP and FN detection on the 11,076K Hands and the 'Poly U HD' datasets. Right and left DFKP and FN detection for both hands showed excellent accuracy at the highest levels across both datasets. The recognition accuracy in the DFKP and FN detection method will be evaluated utilizing a number of steps that impact the precision of feature extraction and classification. As demonstrated in our study [27], this segmentation method was utilised and suggested to be applied to the left and right IKP on 11K's hands. According to our experimental analysis and performance evaluation, concatenated DL models are the most effective for recognizing users, independent of fine-tuning models.

The concatenated DL models with left (DFKP and FN) were found to outperform the right (DFKP and FN) in extracting abstract and high-level features on both 11K Hands and PloyU HD datasets in Tables (2, and 3) in two sections (4.2). We noticed that the best accuracy was achieved by the left DFKP and FN on the 11K Hands dataset, followed right DFKP and FN for the PloyU HD datasets, which ranked second, while the right DFKP and FN for the 11K Hands dataset achieved ranked third, in terms of extracting the key components of users individually and through fusion. The F1_score for the concatenated DL models on the BRDTH, FNRDR, MRDR, MIRDL, and MIRDL is 99.21%, 99.34%, 99.50%, 99.22%, and 99.22% for the 11K Hands dataset, while the F1_score for the concatenated DL models on the BRDM, FNRDTH, MRDM, AND MIRDM is 99.23%, 99.25%, 99.42%, and 99.35% for the PloyU HD dataset. F1_score for the concatenated DL models on the BLDTH, FNLDR, MLDR, and MILDR is 99.29%, 99.39%, 99.52%, and 99.27% for the 11K Hands dataset; thus, we noticed that concatenated fusion of both DFKP and FN (left and right) achieved an F1_score of 99.78%, and 99.46% for the

11K Hands dataset, while the F1_score for concatenated fusion of the right DFKP and FN achieved an F1_score of 99.64% for the PloyU HD dataset.

During analysis and evaluation, the computational complexity exhibited that the proposed concatenated DL models have a larger number of parameters as compared to the individual models, and thus, the computational requirement of the proposed system is increased. DenseNet201, however, has better feature reusability and better enhancement of information flow; InceptionV3 has better feature extraction with multi-scale; ResNet50 has better deep feature learning with residual connection; MobileNetV2 has better feature representation with lower computation. Combining these complementary features results in a robust and comprehensive representation of the biometric data features, which justifies the increased computational complexity in exchange for the substantial enhancement in recognition performance, as shown in Table 5 in section (4.3).

To evaluate the contribution of the proposed concatenated DL Models strategy, an ablation study was conducted by comparing the performance of individual DL models with the proposed concatenated DL models utilizing four pre-trained models. The results exhibited that all individual models performed well in the recognition task, but the combination of four models (DenseNet201, InceptionV3, ResNet50, and MobileNetV2) achieved the highest accuracy compared to the individual models, as well as compared to the combination of three models (DenseNet201, InceptionV3, and ResNet50), where F1_score for individual InceptionV3, ResNet50, DenseNet201, and MobileNetV2 models are 91.84%, 96.53%, 92.94%, and 95.74%, while the F1_score for the concatenation of three DL models is 84.77% for 11k Hands dataset (Right Dorsal). The F1_score for individual InceptionV3, ResNet50, DenseNet201, and MobileNetV2 models are 95.59%, 95.59%, 93.51%, and 93.99%, while the F1_score for the concatenation of three DL models is 87.1% for the PolyU HD dataset (Right Dorsal). We noticed F1_score for individual InceptionV3, ResNet50, DenseNet201, and MobileNetV2 models are 95.39%, 99.98%, 87.83%, and 98.97%, while the F1_score for the concatenation of three DL models is 88.18% for the 11k Hands dataset (Left Dorsal), as shown in Table 6 in section (4.4). This improvement is attributed to the complementary nature of the extracted features, as each model is able to capture various patterns and characteristics of biometric features, providing a more comprehensive and distinctive representation compared to utilizing any single model.

During the key generation process, the proposed ECS-FKPRS utilized the Rubik's Cube technique. We observed that the 11k Hands and PloyU HD datasets achieved excellent results in generating various keys by utilizing concatenated feature fusion, left DFKP and FN, and right DFKP and FN, where, in terms of average security score, it is 99.3%, these results confirm that combining the DFKP and FN biometric features within the Rubik's Cube architecture generates keys with strong random stability. High numerical homogeneity and excellent attack resistance clearly demonstrate the efficiency of the proposed generation algorithm in producing secure and balanced keys for all biometric samples. Overall, these results confirm that the Rubik's Cube-based approach offers an ideal balance between execution speed, high randomness, and robust security, making it an effective and practical choice for key generation in real-time biometric encryption systems. The stability of the biometric key is an important factor in biometric encryption systems, meaning that the features extracted from biometric data may contain some noise and/or slight variations in the way the biometric data was captured. The proposed system mitigates the effect of such variations in the preprocessing phases by utilizing the hand module (MediaPipe Module) for DFKP and FN Segmentation and utilizing DL models that are capable of learning more stable and distinct features. The current system does not explicitly use mechanisms like fuzzy extractors or error correction codes, but their inclusion would add significantly to the stability of the biometric key and its resistance to variations resulting from noise or different imaging conditions, and this is a promising direction for future research.

Thus, in the encryption phases, we observed that the system achieved high security based on the fusion of excellent features obtained from the concatenated DL models applied to both DFKP and FN (left and right), where the concatenated feature fusion keys in the lightweight Grain-128a algorithm were utlized for encryption, the first-ranked left DFKP and FN with NPCR, UACI, Entropy, and SSIM (99.64%, 30.36%, 799.74%, and 0.0216), followed by right DFKP and FN with the same security metrics (99.64%, 30.32%, 799.72%, and 0.0210) which is second-ranked. These results for the 11k hands dataset utilizing the concatenated DL models, while third-ranked security metrics are 99.62%, 30.32%, 799.69%, and 0.0192 for PloyU HD datasets utilizing the concatenated DL models. We observed that the time taken for encryption and decryption was less, which indicates that the algorithm utilized for encryption with keys generated through the Rubik's Cube concept achieved excellent results. Thus, average for security metrics for the 11k Hands dataset was (99.95%, 30.23%, 799.70%, and 0.0209), while the average for security metrics for the PloyU HD dataset was (99.63%, 30.21%, 799.69%, and 0.0215).

The suggested encryption algorithm demonstrated robust resistance to statistical attacks, with high entropy values close to the ideal value (8.0 for 8-bit grayscale images) and correlation coefficients between adjacent pixels approaching zero, indicative of a high level of randomness and security. Overall, the correlation and entropy results confirm that the proposed Grain-128a-based system produces cryptographic images with strong randomness, near-perfect pixel independence, and high resistance to statistical attacks, making it suitable for advanced biometric security applications.

The pioneering system of the proposed ECS-DFKPFNRS is highlighted in Tables 12 and 13 through comparison with several state-of-the-art for cryptographic key generation techniques. Our system shows a higher recognition accuracy and enhanced cybersecurity compared to other systems that utilize DL models for

classification and cryptographic key generation techniques, as evidenced by related reviews. This system distinctly outperforms existing methods when analysis and evaluated against key parameters, consisting of dataset description, classifiers, feature extraction (FE) for concatenated DL models by four pre-trained models, preprocessing, key generation, encryption algorithms, and resistance to statistical attacks.

Table 12: Comparison of the suggested ECS-DFKPFNRS with some state-of-the-art for DL models

Ref.	Dataset	Preprocessing Methods	FE Models	Classifier Methods	Results
[13]	PolyU FKP, and the PolyU Contactless FKI	Region of Interest (ROI) segmentation	CNN		EER= 0.928 %, and CRR=97.01%
[15]	Delhi Finger Knuckle	-	VGG-19 model for FC (F6 and F7), Use PCA for reduction dimensionality	ANN, KNN, NB, and RF	Accuracy=95%
[16]	Kaggle[27], and Roboflow[29]	Resized, Normalized, RGB conversion	VGG-16 model	RF	Accuracy=97.02%, and F1 score = 97.01%
[17]	11,076K Hand	edge-preserving smoothing for every image (Bilateral Filter (BF))	CNN	SVM	Accuracy=0.966%
[18]	Hong Kong Polytechnic University" (PolyU)	-	AlexNet, DenseNet, EfficientNet, GoogleNet, SCNNs, ResNet50, and “VisionTransformer” (VT) models		Accuracy= 98.176% for EfficientNet, 96.224% for AlexNet, 95.601% for GoogleNet, 92.598% for ResNet50, 81.224% for DenseNet, and 79.513% for VT
Our Proposed ECS-DFKPFNRS	11,076K Hands and the ’Poly U HD’	Blurring Hand Images, Convert to HSV Color Space, Morphological Operations (Dilation and Erosion), Median Filtering, and MediaPipe Module	a Concatenated DL model by utilizing pre-trained models, e.g., InceptionV3, ResNet 50, DenseNet201, and MobileNetV2)		F1-Score=99.78%, and 99.46% for 11,076K Hands, and F1-Score=99.64% for Poly U HD

Table 13: Comparison of the suggested ECS-DFKPFNRS with some state-of-the-art for cryptographic key generation techniques

Ref.	Preprocessing Methods	FE Methods	Key Generation	Cryptographic Algorithm	Results
[19]	Gray-Scale, threshold, binarized from higher to lower intensity levels, and centroid	a Tree-based algorithm	a key based on nail images named "NailKey"	-	FAR = 0.001 %, and FRR = < 1 %
[20]	Region of Interest (ROI) Extraction	"Finger Dorsal Feature Extraction Network" (FDFNet)	Bio-Hashing technique		EER = 0.0008%, and CRR= 100%
[21]	Key component Extraction utilizing SIFT method	k-means algorithm	Hash Algorithm (SHA) function	AES algorithm	GAR=99.4%, and FRR=0.6
Our Proposed ECS-DFKPFNRS	Blurring Hand Images, Convert to HSV Color Space, Morphological Operations (Dilation and Erosion), Median Filtering, and MediaPipe Module	a Concatenated DL model by utilizing pre-trained models, e.g., InceptionV3, ResNet 50, DenseNet201, and MobileNetV2)	Enhanced Rubik's Cube technique by adding combined logistic scrambling, permutations, and SHA-512 hashing	Grain-128a lightweight algorithm and utilizes attack resistance (such as Correlation analysis and entropy distribution analysis)	NPCR (99.95% and 99.63%), and UACI (30.23% and 30.21%)

6.0 CONCLUSION AND FUTURE WORK

In the presented research, an enhanced cybersecurity for the Dorsal-FKP and FN recognition system, named ECS-DFKPFNRS, is proposed for analysis and performance evaluation on the two datasets: 11K Hands and PloyU HD. The proposed system utilizes a novel segmentation method that depends on the advanced Mediapipe Module after applying a set of processes on the hands, such as median filters, blurring, HSV color space conversion, and morphological operations (Dilation and Erosion) to obtain the dorsal surface of the key components namely the base knuckle, the major knuckle, the minor knuckle, and the fingernails for all fingers (little, ring, middle, index, and thumb) for both hands. These keys are utilized for extracting optimal feature vectors for user recognition by adopting four pre-trained models to implement a concatenated DL model. The proposed ECS-DFKPFNRS was analyzed and performance evaluated, where images were captured in closed, partially closed, and fully open situations. The proposed system continuously improves the recognition accuracy, as the biometric procedure is based on concatenated fusion of all DFKP and FN or DFKP and FN separately for all fingers on both hands. The ECS-DFKPFNRS suffers limitations in some of the key knuckle hand components (basic, Major, and Minor) of the right fingers (Index, Little, and Thumb) when using the concatenated DL model for concatenated fusion features vector recognition accuracy. Due to the use of parameter values that are inappropriate for the concatenated DL model in two datasets, the results, while satisfactory, are insufficient when compared to other fingers. Furthermore, it generates keys in Rubik's Cube techniques, which have attained high security employing keys in the lightweight Grain-128a algorithm by utilizing these fusion features, left DFKP and FN and right DFKP and FN. The constraints arising from the four pre-trained deep learning models impact key generation and, consequently, the encryption process. Therefore, future research on this subject should examine more intricate feature extraction models and select optimal parameter values for concatenated models. Enhancing the system's cybersecurity and resilience to statistical attacks is the main goal of these procedures.

Authors' Declaration

- Conflicts of Interest: None.
- We hereby confirm that all the Figures and Tables in the manuscript are ours. Furthermore, any Figures and images, that are not ours, have been included with the necessary permission for re-publication, which is attached to the manuscript.

Authors' Contribution Statement

H.S.C., and T.A. participated in proposing the research idea, the significant roles that each researcher played can be summed up as follows: The author H.S.C. find relevant sources, collected the dataset and configured the final folders of each category, and designed the multimodal, including the architecture of the deep learning for each models, make descriptive tables, and analyze the results and tables. On the other hand, researcher T.A. played a major role in drawing the figures, writing algorithms, the grammatical aspect, the linguistic aspect, minimizing plagiarism and quoting, determining the scientific methodology of the research, determining the initial research directions, and directly supervise the intellectual and scientific material of the article.

ACKNOWLEDGEMENT

The author(s) would like to thank Mustansiriyah University (www.uomustansiriyah.edu.iq), Baghdad, Iraq, for its support in the present work, and also the Sfax University, National School of Electronics and Telecommunications of Sfax (ENET'COM), Sfax, Tunisia.

REFERENCES

[1] D. C. Joseph, "The Role of Biometrics in Enhancing Cybersecurity," *International Journal of Research Publication and Reviews,* vol. 5, no. 3, pp. 5953-5958, 2024.

[2] S. Lee, S.-W. Jang, D. Kim, H. Hahn, and G.-Y. Kim, "A novel fingerprint recovery scheme using deep neural network-based learning," *Multimedia Tools and Applications,* vol. 80, no. 26, pp. 34121-34135, 2021. https://doi.org/10.1007/s11042-020-09157-1.

[3] K. Příhodová and M. Hub, "Hand-Based Biometric Recognition Technique-Survey," 2020. https://dx.doi.org/10.25046/aj050683.

[4] A. Aftab, F. A. Khan, M. K. Khan, H. Abbas, W. Iqbal, and F. Riaz, "Hand-based multibiometric systems: state-of-the-art and future challenges," PeerJ Computer Science, vol. 7, p. e707, 2021. https://doi.org/10.7717/peerj-cs.707.

[5] N. E. Chalabi, A. Attia, and A. Bouziane, "Multimodal finger dorsal knuckle major and minor print recognition system based on PCANET deep learning," ICTACT J Image Video Process, vol. 10, no. 3, pp. 2153-2158, 2020. https://doi.org/10.21917/ijivp.2020.0308.

[6] D. L. Woodard and P. J. Flynn, "Finger surface as a biometric identifier," *Computer vision and image understanding,* vol. 100, no. 3, pp. 357-384, 2005. https://doi.org/10.1016/j.cviu.2005.06.003.

[7] N. Duta, "A survey of biometric technology based on hand shape," *Pattern recognition,* vol. 42, no. 11, pp. 2797-2806, 2009. https://doi.org/10.1016/j.patcog.2009.02.007

[8] M. S. Obaidat, S. P. Rana, T. Maitra, D. Giri, and S. Dutta, "Biometric security and internet of things (IoT)," in *Biometric-based physical and cybersecurity systems*: Springer, 2018, pp. 477-509.

[9] G. Meena and S. Choudhary, "Biometric authentication in internet of things: A conceptual view," *Journal of Statistics and Management Systems,* vol. 22, no. 4, pp. 643-652, 2019. https://doi.org/10.1080/09720510.2019.1609722.

[10] A. Nigam and P. Gupta, "Finger knuckleprint based recognition system using feature tracking," in *Chinese Conference on Biometric Recognition*, 2011: Springer, pp. 125-132. https://doi.org/10.1007/978-3-642-25449-9_16.

[11] A. Kuznetsov, D. Zakharov, E. Frontoni, L. Romeo, and R. Rosati, "Deep learning based fuzzy extractor for generating strong keys from biometric face images," in *2022 IEEE 9th International Conference on*

Problems of Infocommunications, Science and Technology (PIC S&T), 2022: IEEE, pp. 421-426. https://doi.org/10.1109/PICST57299.2022.10238643.

[12] Q. Yao, D. Song, X. Xu, and K. Zou, "Visual feature-guided diamond convolutional network for finger vein recognition," *Sensors (Basel, Switzerland),* vol. 24, no. 18, p. 6097, 2024. https://doi.org/10.3390/s24186097.

[13] D. Thapar, G. Jaswal, and A. Nigam, "Fkimnet: a finger dorsal image matching network comparing component (major, minor and nail) matching with holistic (finger dorsal) matching," in *2019 international joint conference on neural networks (IJCNN)*, 2019: IEEE, pp. 1-8. https://doi.org/10.1109/ijcnn.2019.8852390.

[14] Y. Fan, M. You, J. Ge, G. Zhai, and S. Wang, "Segmentation and Feature Extraction of Fingernail Plate and Lunula Based on Deep Learning," *bioRxiv,* p. 2024.07. 26.605289, 2024. https://doi.org/10.1101/2024.07.26.605289.

[15] A. S. Tarawneh *et al.*, "DeepKnuckle: deep learning for finger knuckle print recognition," *Electronics,* vol. 11, no. 4, p. 513, 2022. https://doi.org/10.3390/electronics11040513.

[16] A. Sharma, H. Phulwani, A. Gupta, and A. P. Singh, "Fingernail Image-based Health Assessment Using a Hybrid VGG16 and Random Forest Model," 2024. https://doi.org/10.5109/7326956.

[17] M. Afifi, "11K Hands: Gender recognition and biometric identification using a large dataset of hand images," *Multimedia Tools and Applications,* vol. 78, no. 15, pp. 20835-20854, 2019. https://doi.org/10.1007/s11042-019-7424-8.

[18] A. Altaher *et al.*, "Finger knuckle print classification using pretrained vision models," in *2023 IEEE 20th International Conference on Smart Communities: Improving Quality of Life using AI, Robotics and IoT (HONET)*, 2023: IEEE, pp. 62-67. https://doi.org/10.1109/HONET59747.2023.10374940.

[19] Y. Hang and Z. Yang, "NailKey: Mutable Biometric Using Fingernails," in *Proceedings of the 20th ACM Asia Conference on Computer and Communications Security*, 2025, pp. 1506-1519. https://doi.org/10.1145/3708821.3736196.

[20] A. Singh, A. Arora, S. H. Patel, G. Jaswal, and A. Nigam, "FDFNet: A secure cancelable deep finger dorsal template generation network secured via. bio-hashing," in *2019 IEEE 5th International Conference on Identity, Security, and Behavior Analysis (ISBA)*, 2019: IEEE, pp. 1-9. https://doi.org/10.1109/isba.2019.8778520.

[21] M. Arunachalam and K. Subramanian, "Finger Knuckle Print Authentication Using AES and K-Means Algorithm," *International Arab Journal of Information Technology (IAJIT),* vol. 12, 2015.

[22] A. Kumar and Z. Xu, "Personal identification using minor knuckle patterns from palm dorsal surface," *IEEE Transactions on Information Forensics and Security,* vol. 11, no. 10, pp. 2338-2348, 2016. https://doi.org/10.1109/tifs.2016.2574309.

[23] M. Hell, T. Johansson, and W. Meier, "Grain: a stream cipher for constrained environments," *International journal of wireless and mobile computing,* vol. 2, no. 1, pp. 86-93, 2007. https://doi.org/10.1504/ijwmc.2007.013798.

[24] M. Hell, T. Johansson, A. Maximov, and W. Meier, "A stream cipher proposal: Grain-128," in *2006 IEEE International Symposium on Information Theory*, 2006: IEEE, pp. 1614-1618. https://doi.org/10.1109/isit.2006.261549.

[25] M. Agren, M. Hell, T. Johansson, and W. Meier, "Grain-128a: a new version of Grain-128 with optional authentication," *International Journal of Wireless and Mobile Computing,* vol. 5, no. 1, pp. 48-59, 2011. https://doi.org/10.1504/IJWMC.2011.044106.

[26] N. M. Naser and J. R. Naif, "A systematic review of ultra-lightweight encryption algorithms," *International Journal of Nonlinear Analysis and Applications,* vol. 13, no. 1, pp. 3825-3851, 2022. http://dx.doi.org/10.22075/ijnaa.2022.6167.

[27] H. S. Chyad and T. Abbes, "Finger Knuckles Patterns and Fingernails Recognition For Personal Identification Based On Multi-Model Deep Learning Features," *Malaysian Journal of Computer Science,* vol. 38, no. 1, pp. 1-28, 2025.

[28] R. Yan and L. Shao, "Blind image blur estimation via deep learning," *IEEE Transactions on Image Processing,* vol. 25, no. 4, pp. 1910-1921, 2016.

[29] H. Yamaura, M. Tamura, and S. Nakamura, "Image blurring method for enhancing digital content viewing experience," in *International Conference on Human-Computer Interaction*, 2018: Springer, pp. 355-370. https://doi.org/10.1007/978-3-319-91238-7_29.

[30] V. Chernov, J. Alander, and V. Bochko, "Integer-based accurate conversion between RGB and HSV color spaces," *Computers & Electrical Engineering,* vol. 46, pp. 328-337, 2015. http://dx.doi.org/10.1016/j.compeleceng.2015.08.005.

[31] D. Giuliani, "Metaheuristic algorithms applied to color image segmentation on HSV space," *Journal of Imaging,* vol. 8, no. 1, p. 6, 2022. https://doi.org/10.3390/jimaging8010006.

[32] F. S. Hassan and A. Gutub, "Improving data hiding within colour images using hue component of HSV colour space," *CAAI Transactions on Intelligence Technology,* vol. 7, no. 1, pp. 56-68, 2022. https://doi.org/10.1049/cit2.12053.

[33] K. A. M. Said and A. B. Jambek, "Analysis of image processing using morphological erosion and dilation," in *Journal of Physics: Conference Series*, 2021, vol. 2071, no. 1: IOP Publishing, p. 012033. https://doi.org/10.1088/1742-6596/2071/1/012033.

[34] S. Raju and E. Rajan, "Skin texture analysis using morphological dilation and erosion," *Int. J. Pure Appl. Math,* vol. 118, pp. 205-223, 2018.

[35] J. E. Rodríguez and D. Ayala, "Erosion and Dilation on 2-D and 3-D Digital Images: A New Size-Independent Approach," in *VMV*, 2001.

[36] B. Justusson, "Median filtering: Statistical properties," in *Two-dimensional digital signal processing II: transforms and median filters*: Springer, 2006, pp. 161-196. https://doi.org/10.1007/bfb0057597.

[37] H. Soni and D. Sankhe, "Image restoration using adaptive median filtering," *Image,* vol. 6, no. 10, 2019.

[38] K. O. Boateng, B. W. Asubam, and D. S. Laar, "Improving the effectiveness of the median filter," 2012.

[39] R. Kumar, A. Bajpai, and A. Sinha, "Mediapipe and CNNs for real-time asl gesture recognition," *arXiv preprint arXiv:2305.05296,* 2023.

[40] G. Amprimo, G. Masi, G. Pettiti, G. Olmo, L. Priano, and C. Ferraris, "Hand tracking for clinical applications: Validation of the Google MediaPipe Hand (GMH) and the depth-enhanced GMH-D frameworks," *Biomedical Signal Processing and Control,* vol. 96, p. 106508, 2024. https://doi.org/10.1016/j.bspc.2024.106508.

[41] L. Nanni, S. Ghidoni, and S. Brahnam, "Handcrafted vs. non-handcrafted features for computer vision classification," *Pattern Recognition,* vol. 71, pp. 158-172, 2017. https://doi.org/10.1016/j.patcog.2017.05.025.

[42] V. A. Bharadi, "Texture Feature Extraction For Biometric Authentication using Partitioned Complex Planes in Transform Domain," in *IJACSA Special Issue on Selected Papers from International Conference Workshop On Emerging Trends In Technology*, 2012, pp. 39-46. https://doi.org/10.14569/specialissue.2012.020105.

[43] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1-9. https://doi.org/10.1109/cvpr.2015.7298594.

[44] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818-2826. https://doi.org/10.1109/cvpr.2016.308.

[45] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770-778. https://doi.org/10.1109/cvpr.2016.90.

[46] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700-4708.

[47] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510-4520. https://doi.org/10.1109/cvpr.2018.00474.

[48] K. Loukhaoukha, J.-Y. Chouinard, and A. Berdai, "A secure image encryption algorithm based on Rubik′ s cube principle," *Journal of Electrical and Computer Engineering,* vol. 2012, no. 1, p. 173931, 2012. https://doi.org/10.1155/2012/173931.

[49] M. Helmy, E.-S. M. El-Rabaie, I. M. Eldokany, and F. E. A. El-Samie, "3-D image encryption based on Rubik's cube and RC6 algorithm," *3D Research,* vol. 8, no. 4, p. 38, 2017. https://doi.org/10.1007/s13319-017-0145-8.

[50] S. D. Khudhur and H. A. Jeiad, "A content-based file identification dataset: Collection, construction, and evaluation," *Karbala International Journal of Modern Science,* vol. 8, no. 2, pp. 63-70, 2022. https://doi.org/10.33640/2405-609X.3222.

[51] H.-J. Jo and J. W. Yoon, "A new countermeasure against brute-force attacks that use high performance computers for big data analysis," *International journal of distributed sensor networks,* vol. 11, no. 6, p. 406915, 2015. http://dx.doi.org/10.1155/2015/406915.

[52] K. Mali, S. Chakraborty, and M. Roy, "A study on statistical analysis and security evaluation parameters in image encryption," *entropy,* vol. 34, p. 36, 2015.

[53] Y. Shen, L. Wang, and D. Gu, "Security Analysis of NIST Key Derivation Using Pseudorandom Functions," *Cryptology ePrint Archive,* 2025.

[54] S. M. Hameed and Z. H. Ali, "SMS Spam Detection Based on Fuzzy Rules and Binary Particle Swarm Optimization," *International Journal of Intelligent Engineering & Systems,* vol. 14, no. 2, 2021. https://doi.org/10.22266/ijies2021.0430.28.

[55] B. Gülmez, "A novel deep learning model with the Grey Wolf Optimization algorithm for cotton disease detection," *Journal of Universal Computer Science,* vol. 29, no. 6, p. 595, 2023. https://doi.org/10.3897/jucs.94183.

[56] L. K. Suong and K. Jangwoo, "Detection of potholes using a deep convolutional neural network," *Journal of universal computer science,* vol. 24, no. 9, pp. 1244-1257, 2018.

[57] M. Solar and P. Aguirre, "Deep learning techniques to process 3D chest CT," *Journal of Universal Computer Science,* vol. 30, no. 6, p. 758, 2024. https://doi.org/10.3897/jucs.112977.

[58] R. S. Mohammed, K. K. Jabbar, and H. A. Hilal, "Image encryption under spatial domain based on modify 2D LSCM chaotic map via dynamic substitution-permutation network," *International Journal of Electrical and Computer Engineering,* vol. 11, no. 4, p. 3070, 2021. https://doi.org/10.11591/ijece.v11i4.pp3070-3083.

[59] Y. Wu, J. P. Noonan, and S. Agaian, "NPCR and UACI randomness tests for image encryption," *Cyber journals: multidisciplinary journals in science and technology, Journal of Selected Areas in Telecommunications (JSAT),* vol. 1, no. 2, pp. 31-38, 2011.

[60] S. A. Mehdi, K. K. Jabbar, and F. H. Abbood, "Image encryption based on the novel 5D hyper-chaotic system via improved AES algorithm," *International Journal of Civil Engineering and Technology,* vol. 9, no. 10, pp. 1841-1855, 2018.

[61] C. C. Gan and G. Learmonth, "Comparing entropy with tests for randomness as a measure of complexity in time series," *arXiv preprint arXiv:1512.00725,* 2015.

[62] M. B. Tuieb, N. S. Abdul-Razaq, and M. Z. Abdullah, "Novel Video Information Hiding Approach via Random Path and Frame Selection," 2020.

[63] C. A. Mello, M. M. Saraiva, D. P. Menor, and R. Nishihara, "A Comparative Study of Objective Video Quality Assessment Metrics," *J. Univers. Comput. Sci.,* vol. 23, no. 5, pp. 505-527, 2017.

[64] N. Buckley, A. K. Nagar, and S. Arumugam, "On real-valued visual cryptographic basis matrices," *Journal of Universal Computer Science,* vol. 21, pp. 1536-1562, 2016.

[65] L. Bassham *et al.*, "A statistical test suite for random and pseudorandom number generators for cryptographic applications," National Institute of Standards and Technology, 2008. https://doi.org/10.6028/nist.sp.800-22.

[66] A. Sabah, S. Hameed, and K. Maisa'a-Abid-Ali, "Key Generation based on Henon map and Lorenz system," *Al-Mustansiriyah J. Sci.,* vol. 31, no. 1, pp. 41-46, 2020. https://doi.org/10.23851/mjs.v31i1.734.

[67] Y. Wu, L. Zhang, S. Berretti, and S. Wan, "Medical image encryption by content-aware DNA computing for secure healthcare," *IEEE Transactions on Industrial Informatics,* vol. 19, no. 2, pp. 2089-2098, 2022. https://doi.org/10.1109/tii.2022.3194590.

[68] S. A. Mehdi, A. A. H. Alta'ai, and S. A. ABBAS, "A novel chaotic System For Color Image Encryption," *Journal of College of Education,* vol. 1, no. 1, 2017.