International Journal of Computer Networks and Applications (IJCNA)
DOI: 10.22247/ijcna/2024/48 Volume 11, Issue 6, November – December (2024)

RESEARCH ARTICLE

A Novel LPRPO-LSTDCNN Based Side Channel Attack Detection and Secure Data Transmission Framework Using DH-ATM-RFICC

Prasath Vijayan
Department of Computer Science and Engineering, Perunthalaivar Kamarajar Institute of Engineering and Technology, Karaikal, India.
✉ prasathvijayan@outlook.com

Sudalaimuthu T
School of Engineering and Technology, Hindustan Institute of Technology and Science (Deemed to be University), Chennai, India.
tsmuthu@hindustanuniv.ac.in

Received: 04 August 2024 / Revised: 28 October 2024 / Accepted: 08 November 2024 / Published: 30 December 2024

Abstract – Side Channel Attack (SCA) is the exploitation of data security due to information leakage from a device. However, the existing studies didn't focus on the detection of SCA based on its types and the prevailing works had security leakage and time complexities. Therefore, the paper presents the SCA detection with multi-class classification for secured data transmission. First, the device is installed and the public and private keys are generated. Next, Universally Unique Identifier (UUID) is generated and based on the public key and UUID, a hash code is generated using Interpolation Harmonic Entropy based SWIFFT(IHE-SWIFFT) hashing algorithm. At the same time, a secret key is generated using a public key and UUID. Then, with the help of a secret key and public key, data is encrypted using Diffie-Hellman Asymmetric Tent map-based Robust Frobenius Isogenies Curve Cryptography (DH-ATM-RFICC). For the purpose of user authorization, hash code matching is carried out. If the hash code is not matched, then the transaction will be declined to prevent from unauthorized transactions. If the hash code at the time of transaction initialization is matched with the hash code generated during the transaction, then the data is gathered by extracting the features using Gini Point Bi-serial Correlation-based Empirical Wavelet Transform (GPBC-EWT).The extracted features are then reduced using Bag of Deep Features (BoDF) and through hybrid classifier to detect the SCAs using Lagrange Polynomial Red Panda Optimization with Logistic Softmax Tree Hierarchical Deep Convolutional Neural Network (LPRPO-LSTDCNN). Finally, the data under no attack are decrypted in the server for successful transaction. Hence, the proposed model detected the attack with 98.87% Accuracy, 98.86% Precision, 98.88% Recall, and decrypted the data securely in 978ms, thus showing better performance than existing models.

Index Terms – Side Channel Attack, Deep Learning, Cryptography, Deep Convolutional Neural Network, Red Panda Optimization, Hashing algorithm.

1. INTRODUCTION

In recent years, the technologies under automation with top-notch Artificial Intelligence (AI) and sensor models have been increasing [1]. These AI applications are designed to provide the personal information of the users to make successful transactions [2]. For instance, private data, such as Personal Identification Number (PIN), passwords, card details, email data, etc., of the users are entered through the device [3] [4]. This private information is secured using cryptographic applications to prevent the leakage of personal details; but, the attack can occur on a physical platform through side channels [5]. Thus, exposing this private information in any form might be vulnerable to SCA, which was done by hackers [6]. The leakage of users' private data and the loss of users' critical information by the attacker is known as SCA [7]. Normally, SCAs are the significant components of present cybersecurity systems, especially for devices and networks.

The SCA is classified into Timing, Electromagnetic (EM), Simple Power Analysis (SPA), Differential Power Analysis (DPA), and Template Attacks [8]. In these types of attacks, the loss of information is in the form of time, electromagnetic radiation, power consumption, temperature, sound, and operational time [9]. Different types of SCAs are done by revealing the secret key of the user's device, and the hacker uses the secret key to collect private data [10]. Also, the Public Key is made available to every user, which can cause information leakage [11]. Therefore, it is important to detect the SCAs for successful transactions. SCA detection and secure data transmission protect sensitive information, especially as systems heavily rely on connected devices like the Internet of Things (IoT), embedded systems, and mobile

RESEARCH ARTICLE

applications. In most of the existing works, advanced encryption protocols, cryptographic techniques, and key management strategies were used; they provided data confidentiality, security, integrity, and availability across different devices and networks [12].

Also, SCA in the target device was detected using various Machine Learning (ML) and Deep Learning (DL) techniques [13]. Likewise, some existing research works employed ML techniques, such as Random Forest Algorithm (RFA) and Support Vector Machine (SVM) for SCA detection. Also, certain prevailing works used DL techniques like Convolutional Neural Network (CNN) for SCA detection [14]. However, all the types of attacks related to SCA were not detected in the existing deep learning networks. Also, not all the works secured the public and private keys of the target device, which led to unauthorized access and signal trace attacks [15]. Hence, to address these limitations and to improve attack detection, a novel framework for SCA detection using DF-ATM-RFICC and LPRPO-LSTDCNN is proposed in this paper.

1.1. Problem Statement

The drawbacks of the existing works are given below,

- The detection of types of SCA, such as Timing, EM, SPA, DPA, and Template attacks was not concentrated on the prevailing works.
- [16] analyzed the micro-architectural pattern through HPC features, which reduced the SCA detection performance.
- In [17], the training time was high due to the consideration of all the extracted features from the traces.
- [18] revealed the details of the public key to everyone in the network, which led the hacker to attack the data easily.
- The Design Space Exploration (DSE) used Electronic System-level Synthesis (ESLS) for leakage detection. However, ESLS was time-consuming; hence, the detection became slower.

The objectives of the proposed work are enlisted as follows,

- In the proposed work, different types of Side-Channel Attacks are detected by using LPRPO-LSTDCNN classifier.
- The features are extracted using the GPBC-EWT method to improve attack detection.
- To decrease the time required for training the LPRPO-LSTDCNN classifier, the inaccurate features are eliminated by using BoDF technique.
- To prevent the data from being attacked due to the exposure of the public key, the data is secured using the private key and secret key generated based on the public key and UUID.
- To reduce the attack detection time and add more security, hashcode matching, feature reduction, and weight optimization using LPRPO are carried out.

The rest of this paper is organized as follows: Section 2 explains the related work, Section 3 describes the details of SCA detection methods, Section 4 presents the experiment results and analysis, and Section 5 conveys the discussion about the proposed model. Finally, Section 6 concludes the paper with future scope.

2. LITERATURE SURVEY

[16] established a Side Channel Attack detection model in cryptographic application devices. The features of the data were extracted using the Neural Network Support Vector Machine (NEROSVM). Then, the grid search technique was used for the selection of the parameters needed for classification. At last, the selected features were classified for SCA detection using the Restricted Boltzmann Machine (RBM) method. Thus, the model detected the attack accurately and retrieved the Secret Key for decryption of the data. However, the pre-processing of the data was not done, which affected the classification performance.

[17] introduced a deep learning-based SCA detection based on the process of cryptographic algorithm. The signal traces were collected from the cryptographic devices. Then, the important features were extracted and classified using the CNN classifier. The classification was based on profiled SCA and non-profiled SCA. Depending on the leakage detection model, the secret key was generated in the server to decrypt the data. Hence, a successful data transaction was achieved by the introduced research. Yet, only the presence of an attack was detected. But, this model did not detect the type of attack.

[18] presented Leakage Resilient Certificate Based Authenticated Key Exchange (LR-CB-AKE) protocol for leakage detection during data transfer. Here, the data was secured using the entropy technique. The user certification was done by creating a session key. The LR-CB-AKE method generated a secret key, and this was used for decrypting the data. Hence, the secured data transfer was done by the LR-CB-AKE model. But the private keys generated could not be hidden from adversaries, which led to SCA.

[19] developed a deep learning-based side-channel preprocessing with autoencoders for cryptanalysis. The power consumption in the target device was first calculated. Then, a guessing key related to the power hypothesis was generated, and the correlation between the key and the power consumption was made. The correlated data was then classified using Differential Deep Learning Analysis (DDLA). Thus, the developed model accurately identified the

RESEARCH ARTICLE

Simple Power Analysis (SPA) attack and provided improved performance. On the contrary, other types of SCA were not detected by this framework.

[20] accomplished side-channel Gray-box attack for Deep Neural Network (DNN). The Gray-box attack was the SCA that occurred in the DNN, and this leaked the structural information of the DNN. The data acquisition from an Artificial Intelligence (AI) device was done. Then, the power features were extracted, and the Fast Gradient Sign Method (FGSM) was used for detecting the SCA. The adversarial attacks in DNN were accurately detected by the research. However, no preprocessing techniques were used in this model, leading to the misclassification of the data.

[21] demonstrated a distribution algorithm for side-channel analysis in IoT devices. The Points of Interest (POI) selection of the data was done using the Estimation of Distribution Algorithm (EDA). The Template Attacks were identified using the optimization technique. Thus, the research excellently identified the leakage of information in IoT devices. Yet, the EDA detected the Template Attack in the important features, which led to delayed attack detection.

[22] implemented an efficient countermeasure technique for SCA detection in IoT devices. First, based on the defense approach criteria, the Timing Side-Channel Attack Countermeasure Technique (TSCA-CT) was used for decision-making. Then, a decision matrix was built. Then, by using the Fermatean Fuzzy Decision Opinion Score Method (F-FDOSM), the SCA present in the input data was accurately detected. But the criterion that followed the electromagnetic functional attacks could not be classified in this model.

[23] evaluated an encryption security model against the SCA using a lightweight approach. Here, a Hash-based Authenticated Nonce-Misuse Resistant Encryption (HANMRE) was used for encrypting the data. This HANMRE method was an Authenticated Encryption with Associated Data (AEAD) scheme, which was further used for identifying lightweight leakage. Hence, the SPA was precisely identified using the HANMRE technique. Yet, the secret keys were not generated, which resulted in difficulty during data decryption.

[24] integrated Autoencoder-based side-channel analysis in IoT networks. First, from the target device, an identical clone device was made. Then, the side-channel information of the cloned device was categorized in the profiling phase. Using Autoencoder, the features were extracted. Finally, for the extracted feature, the SCA classification was performed using the CNN technique. Thus, the presence of SCA was classified effectively by CNN; also, the model provided high-level security. However, the selection of optimal features was not done, which led to improper SCA detection.

[25] explored integrated side-channel processing in DNN. The input data was mapped and then down-sampled regarding the time domain. Meanwhile, spectral transformation for the mapped data was done regarding the frequency domain. Finally, the transformed data and the down-sampled data were given to the Multi-scale Convolutional Neural Networks (MCNN). Thus, the MCNN classified the SCA attack present in the input data and was proved as a promising model for SCA detection. Yet, the capturing of temporal data during down-sampling was difficult, which delayed the classification of SCA.

Thus, the existing methodologies obtained poor performance in SCA detection owing to the improper pre-processing of data, private key leakage, difficulties in data decryption, improper selection of features, and so on. Also, the prevailing studies didn't identify the types of SCA. Likewise, the existing works provided poor and limited performance for a large number of data sizes or transactions. To conquer these shortcomings, an effective LPRPO-LSTDCNN and DH-ATM-RFICC-based SCA attack detection and secure data transmission framework is proposed in this article.

3. PROPOSED SIDE CHANNEL ATTACK DETECTION MODEL

In the proposed model, the SCA is detected by using LPRPO-LSTDCNN, and the data is secured by utilizing DH-ATM-RFICC methods. This work involves major processes, such as hashcode generation, data encryption, pre-processing, feature extraction, feature reduction, and SCA classification. The framework of the proposed model is depicted in Figure 1.

3.1. Device Setup and Key Generation

First, the devices, such as the Oscilloscope and Processing Unit, which generates signal traces, are setup. These signal traces (T) are the input data and are represented in the equation (1),

$$T = [T_1, T_2, T_3, .., T_{e-1}, T_e] \quad (1)$$

Where, (e) is the number of signal traces. Next, the keys, such as Public Key (β) and Private Key (γ) are generated from the key generation center using DH-ATM-RFICC, which is mentioned in section 3.5. Next, UUID is generated as explained below,

3.2. UUID Generation

To authorize the user during the transaction, a timestamp-based UUID (f) is generated, which is represented in equation (2). This UUID is a 128-bit identifier, and the time stamp ensures the unique generation of (f).

RESEARCH ARTICLE

$$f = \{f^1, f^2, f^3, ..., f^a\} \quad (2)$$

Where, (a) is the number of UUID generated. Next, by using (f), the hashcode is generated as shown in section 3.3.

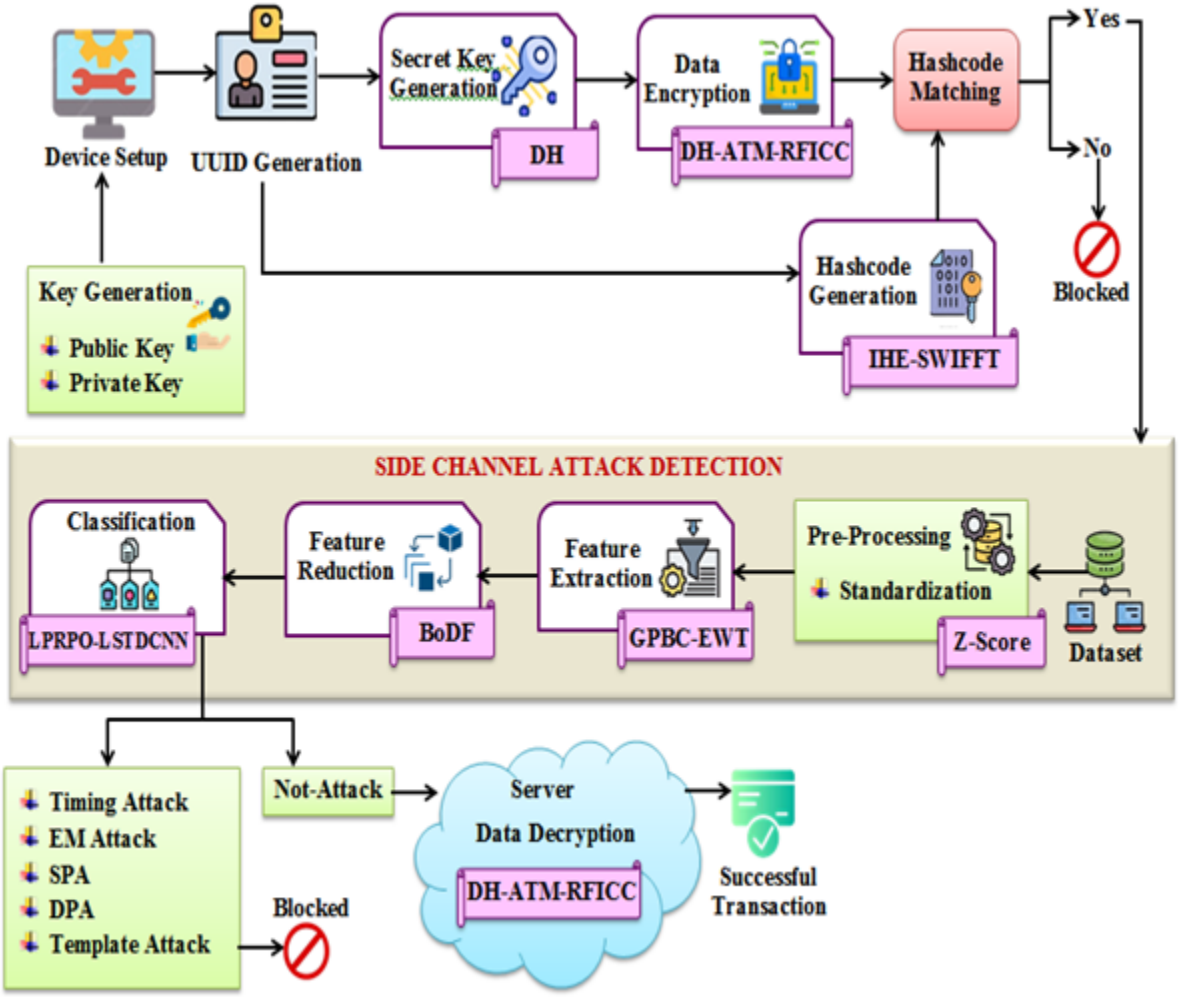

Figure 1 Architecture of Proposed Work

3.3. Hashcode Generation

Here, by using the Public Key (β) and UUID (f), a hashcode is generated using the IHE-SWIFFT algorithm. The SWIFFT hashing algorithm generates a hashcode regarding Fourier coefficients. Due to the generation of hashcode, SWIFFT enhances mathematical security. However, the SWIFFT hashing algorithm is not collision-resistant i.e., it provides the same output for two different inputs. To avoid this issue, Interpolation Harmonic Entropy (IHE), which differs from the hashing output to form a unique hashcode, is used along with SWIFFT. IHE offers collision resistance in SWIFFT, which can include superior distribution of entropy, enhanced non-linearity, greater resistance to attacks, and increased complexity in both the input-output relationship. This makes the IHE harder for adversaries to create collisions, thus improving the overall security of the hash function.The IHE-SWIFFT method is described as: First, the Public Key (β) and UUID (f) are combined and represented by (M), which is equated in the below equation (3),

$$M = \beta \oplus f \quad (3)$$

The combined input (M) is then converted into a matrix $[M]_{c\times d}$ with dimension $(c \times d)$ and is represented in equation (4),

RESEARCH ARTICLE

$$M \rightarrow [M]_{c\times d} \quad (4)$$

To avoid the collision of output, IHE (g) given in equation (5) summarizes the data regarding $[M]_{c\times d}$, is calculated as,

$$g = \sum \frac{1}{2}[M] + \left\langle \{[M]_c - [M]_d\} \times \frac{\{[M]_c - [M]_d\}^2}{\{[M]_d - [M]_c\}^2} \right\rangle \quad (5)$$

Finally, the hashcode is generated in equations (6) and (7) based on $[M]_{c\times d}$, the Fourier coefficient (χ), and IHE (g) and is given as,

$$\delta = c * \log(\chi)\left\{\sum_c [M] \times g\right\} \quad (6)$$

$$\chi = \frac{1}{c}\int \cos(4\pi[M]) \times \exp^{-\left(\frac{2\pi cM}{0.5}\right)} \partial M \quad (7)$$

Where, (δ) is the generated hashcode and $(\pi = 3.14)$. Now, the secret key is generated for data security, which is mentioned in section 3.4.

3.4. Secret Key Generation

The Diffie-Hellman (DH) method that securely exchanges the keys over a public channel is used for generating a Secret Key (SK). The SK is generated to securely encrypt and decrypt the transaction details and prevent them from attacks. The DH algorithm is explained below,

A prime number (i) and its primitive root (i'') are assumed such that $(i'' < i)$. The SK (ε) is generated using Public Key (β) and UUID (f) as given in equation (8),

$$\varepsilon = (\beta)^f * \text{mod}(i) \quad (8)$$

Where, (mod) is a modulo operator. The SK (ε) is then used for encrypting the data, which is discussed in section 3.5.

3.5. Data Encryption

After the generation of SK (ε), the data (T) is encrypted to secure the transaction. Here, Elliptic Curve Cryptography (ECC), which encrypts and decrypts the data faster, is used for encryption. But, in ECC, the Private Key is generated randomly to encrypt the data into variable length, which might lead to a side channel attack.

Hence, to mitigate this issue, Robust Frobenius Isogenies Curve Cryptography (RFICC) that fixes each element into the curve is used instead of ECC. Also, an Asymmetric Tent Map (ATM) that generates the key uniquely is used for generating the private key. Actually, ATM provides improved non-linearity for unpredictability; also, ATM produces high-entropy outputs owing to its chaotic dynamics, which prevents key leakage by diminishing the possibility of duplicate keys and strengthening the resistance against attacks.

The process of DH-ATM-RFICC is detailed below,

- Curve Formation

The Robust Frobenius Isogenies (F) that forms an algebraic closure pattern and fixes the data (T) into the curve is expressed in the below equation (9),

$$F = \left(u^{\kappa^b}, v^{\kappa^b}\right) \quad (9)$$

Here, the curve has parameters (u, v), characteristics (κ), and integers (b). The elliptic curve with (F) and constant (r) is expressed as,

$$\left[v^{\kappa^b}\right]^2 = \left[u^{\kappa^b}\right]^3 + r\left(u^{\kappa^b}\right) + r \quad (10)$$

The aforementioned equation (10) specifies the elliptical curve.

- Key Generation

The curve shown in equation (9) and the Public Key (β) are used for Private Key (γ) generation. In equation (11), the Public Key (β) is generated using the point (o) and constants (ω_1, ω_2) of the curve and is given as,

$$\beta = o \times (\omega_1 + \omega_2) \quad (11)$$

The ATM is a piecewise linear equation, and this automatically generates the (γ) as equated in the below equation (12),

$$\gamma = \beta \times \min(h, h-1) \quad (12)$$

Where, (h) is the point from the curve.

- Encryption

The data (T) is finally encrypted using the Secret Key (ε) and the Private Key (γ) as,

$$T^* = (T + \varepsilon) + (h * \gamma) \quad (13)$$

In equation (13), (T^*) represents the encrypted data. The pseudocode for DH-ATM-RFICC is given in algorithm 1.

RESEARCH ARTICLE

Input: Signal Traces (T)

Output: Encrypted Data (T^*)

Begin

Initialize curve parameters (u, v)

Generated Secret Key

$$\varepsilon = (\beta)^f * \mathrm{mod}(i)$$

Calculate Robust Frobenius Isogenies $F = \left(u^{\kappa^b}, v^{\kappa^b}\right)$

Form the curve

$$\left[v^{\kappa^b}\right]^2 = \left[u^{\kappa^b}\right]^3 + r\left(u^{\kappa^b}\right) + r$$

While $\left(h \in v^{\kappa^b}\right)$

Evaluate Private Key $\gamma = \beta \times \min(h, h-1)$

For (ε)

Encrypt data

$$T^* = (T + \varepsilon) + (h * \gamma)$$

End for

End while

Obtain Encrypted data (T^*)

End

Algorithm 1 DH-ATM-RFICC

Next, the hashcode is matched to transmit the encrypted data (T^*), which is explained in section 3.6.

3.6. Hash Code Matching

To avoid unauthorized users, the hashcode (δ) generated from the device during transaction initialization is matched with the hashcode generated during the transaction by the server. The condition to match hashcodes (H) is represented in the equation (14) as,

$$H = \begin{cases} \Lambda & \forall\left(\delta = \hat{\delta}\right) \\ \Diamond & \forall\left(\delta \neq \hat{\delta}\right) \end{cases} \quad (14)$$

Where, (Λ) is the hashcode matched condition, and $(\Diamond)$ is the hashcode mismatched condition. When the hashcode is not matched $(\Diamond)$, the encrypted data (T^*) is blocked for further processing, and when the hashcode gets matched (Λ), (T^*) is given for SCA detection.

3.7. SCA Detection

For detecting SCA in real time, first, the SCA detection model using the LPRPO-LSTD CNN-based deep learning classifier is trained with respect to the publically available SCA dataset. The training of the SCA detection model is detailed as follows,

3.7.1. Data Collection

First, the signal traces $(\dddot{T})$ are collected from the AES_HD dataset and are represented in the equation (15),

$$\dddot{T} = \left\{\dddot{T}_1, \dddot{T}_2, \dddot{T}_3 ..., \dddot{T}_w\right\} \quad (15)$$

Where, (w) is the number of data used for training. Now, $(\dddot{T})$ is preprocessed as detailed below,

3.7.2. Pre-Processing

Here, the data $(\dddot{T})$ is preprocessed using the Z-Score standardization technique, which scales the input accurately into a standard range of $(0,1)$. The preprocessing, which is represented in equation (16), is done because it helps in extracting relevant features from the data with less time. The Z-Score equation is described as,

$$T'' = \left(\dddot{T} - \eta\right) \Big/ \sigma^{T^*} \quad (16)$$

$$\eta = \frac{\sum_{e} \dddot{T}}{w} \quad (17)$$

Where, (T'') is the preprocessed data, (w) is the number of $(\dddot{T})$, and (η) signifies the mean of $(\dddot{T})$, which is given in equation (17), and $\left(\sigma^{T^*}\right)$ is the standard deviation of $(\dddot{T})$. Then, the features required for the classification of different types of SCAs are extracted from (T'') as described in section 3.8.

3.7.3. Feature Extraction

In this phase, for effective SCA detection, the features are extracted from (T''). To extract the features, Empirical Wavelet Transform (EWT), which extracts the data by creating a wavelet filter bank, is used. Also, EWT excellently captures the important components of the signal by adapting

RESEARCH ARTICLE

to the specific frequency bands of the signal, thus leading to better feature extraction for signals. But, the Short-Time Fourier Transform (STFT) and Continuous Wavelet Transform (CWT) employ fixed basis functions; thus, they are not optimal for all types of signals. However, shift invariance occurs in EWT i.e., the input and output relationship changes with time and affects the feature extraction. Hence, Gini Point Biserial Correlation (GPBC), is used in EWT, which makes comparisons over the continuous variables of the input data. Also, when dealing with non-stationary and complex signals, GPBC-EWT effectively performs feature extraction compared to techniques, such as STFT and CWT. The GPBC-EWT technique is explained below,

Step 1: Local Maxima and Minima

First, let the number of decompositions be represented by (D) for the input (T'') with the (w) number of data. Now, the boundary limits (l^e) are set as given in below equation (18),

$$l^e = \begin{cases} l_0^e = 0 \\ l_D^e = \pi \end{cases} \tag{18}$$

Where, (l_0^e) is the local minimum with zero value and (l_D^e) is the local maxima with $(\pi = 22/7)$.

Step 2: Correlation Factor

The correlation factor (λ) is calculated using GPBC, which uses the maxima and minima limits of the data to extract the feature correctly. The (λ) is equated in the upcoming equation (19),

$$\lambda = \frac{2\sum_e T''}{D \times \sigma^{T''}} * \sqrt{\frac{\left(l_D^e - l_0^e\right)}{l^e}} \tag{19}$$

Where, $(\sigma^{T''})$ is the standard deviation of (T'').

Step 3: Scaling function and Empirical Wavelet

In equations (20) and (21), the scaling function (B) and the Empirical Wavelet (V) of (T'') are calculated using (λ).

$$B(T'') = \begin{cases} 1 & \forall\left[l_D^e \le (1-\lambda) l_0^e\right] \\ \cos\left(\frac{\pi \times \lambda}{2}\right) & \forall\left[(1-\lambda) l_0^e \le l_D^e \le (1+\lambda) l_0^e\right] \\ 0 & otherwise \end{cases} \tag{20}$$

$$V(T'') = \begin{cases} 1 & \forall\left[(2+\lambda) l_0^e \le l_D^e \le (2-\lambda) l_0^e\right] \\ \cos\left(\frac{\pi \times (\lambda+1)}{2}\right) & \forall\left[(2-\lambda) l_0^e \le l_D^e \le (2+\lambda) l_0^e\right] \\ \sin\left(\frac{\pi \times (\lambda+1)}{2}\right) & \forall\left[(1-\lambda) l_0^e \le l_D^e \le (1+\lambda) l_0^e\right] \\ 0 & otherwise \end{cases} \tag{21}$$

As given in equation (22), the values (B) and (V) are used for decomposing the data.

Step 4: Transforms

Finally, the features (S), such as normalized signal, decomposition number, wavelet decomposition, Fast Fourier Transform (FFT), Fourier decomposition analysis, apply filters, enforce sample flow type, convert integer float, check mono, and calculated wavelet FFT are extracted as,

$$S = B(T'') \times V(T'') \tag{22}$$

$$S \rightarrow \{S_1, S_2, S_3, ..., S_m\} \tag{23}$$

Where, S_m is the number of extracted features, which is given in equation (23). Next, the features are reduced as given in the following section,

3.7.4. Feature Reduction

Here, from (S), important features are selected by the process of Bag of Deep Features (BoDF). BoDF superiorly captures complex, nonlinear patterns in the data; also, BoDF preserves more important semantic features than other techniques. Likewise, BoDF proficiently learns significant patterns, leading to more reliable feature reduction, even in noisy environments. Owing to the aforementioned advantages, the BoDF is chosen for feature reduction. This method clusters the features (S); then, the features are reduced based on histogram formation, which is explained further,

- K-Means Clustering

First, the clusters (k) are chosen from (S) to group the features. Then, the centroid to cluster the features is selected randomly and is represented by (C). The distance between the centroid (C) and the features (S) is calculated using Euclidean distance (ρ) as given in the below equation (24),

$$\rho = \sqrt{\sum_m (C - S)^2} \tag{24}$$

RESEARCH ARTICLE

Where, (m) is the number of features. In equations (25) and (26), the features with the shortest distance are clustered as,

$$k \to (\rho > S) \quad (25)$$

$$k(\rho) = \{k_1, k_2, k_3, ..., k_j\} \quad (26)$$

Where, (j) is the number of clusters formed after calculating (ρ). Now, the histogram of each feature in (j) is calculated to reduce the features from the grouped features.

- Histogram of Features

Here, the unique features from the grouped data $[k(\rho)]$ are selected using the Histogram equation as represented in equations (27) and (28),

$$N(v) = \frac{(1-m^*)}{m} k(\rho) \quad (27)$$

$$N = [N_1, N_2, N_3, ..., N_s] \quad (28)$$

Where, (v) is the histogram of $[k(\rho)]$ with points $(v = 1,2,...,1-m^*)$, (m^*) is the maximum histogram point, (N) is the reduced feature, and (s) is the number of reduced features. Next, the different types of SCA are classified from the reduced features (N), which are further described in 3.7.5.

3.7.5. Classification

Here, the Tree Hierarchical Deep Convolutional Neural Network (THDCNN) that learns the features automatically and combines the features hierarchically to detect the attack is used for classification. But this classifier has the problem of a slow convergence rate and detects the attack with high latency.

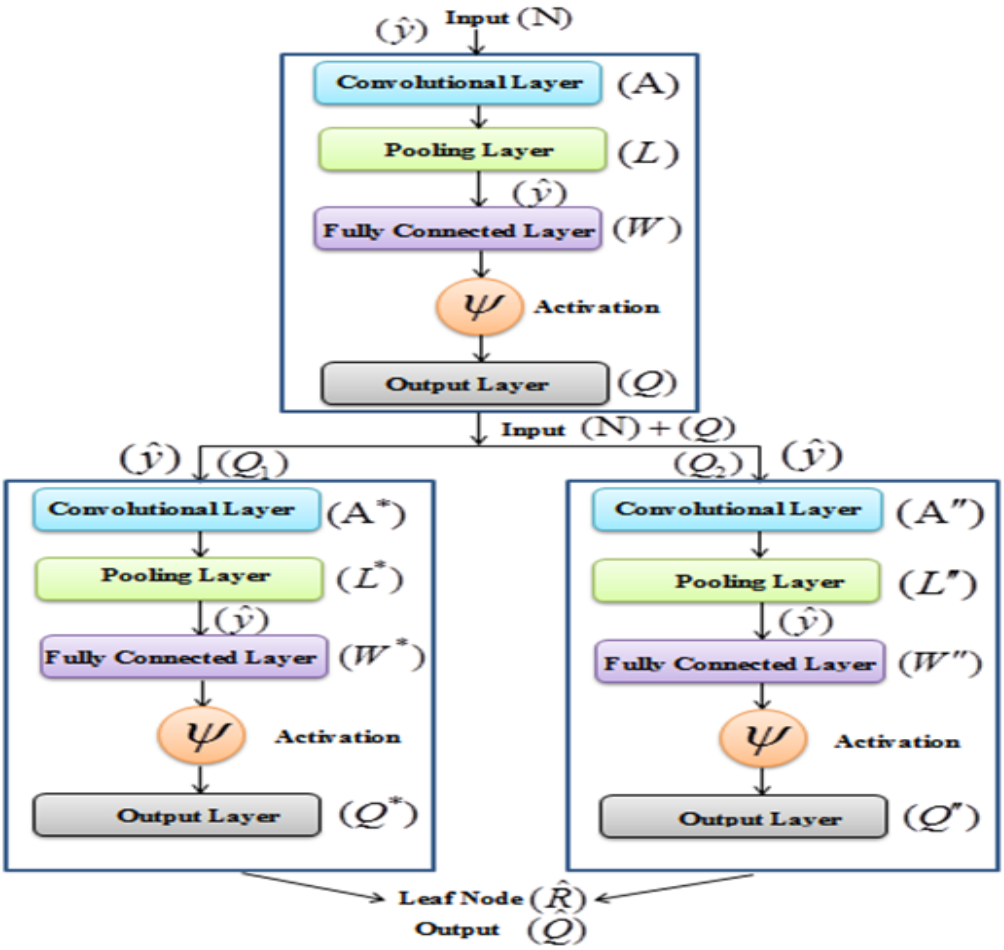

Figure 2 Framework of LPRPO-LSTDCNN Classifier

RESEARCH ARTICLE

Therefore, the Logistic Softmax (LS) activation function that provides probability values and makes the process faster, especially for complex attack types in THDCNN. Also, the LS activation function gives a clear indication of the likelihood of each class than other activation functions. Likewise, by emphasizing the most applicable class-specific patterns, LS enables the THDCNN to learn more discriminative features from the side-channel data.

Furthermore, the LS activation function improves the convergence speed and accuracy and reduces the time complexities; so, it is chosen over other activation functions. Also, to enhance the SCA detection, the weight is initialized by the Red Panda Optimization (RPO) algorithm, which selects the weights according to the behavior of the Red Panda. RPO is chosen due to its ability to find global optima in complex and high-dimensional search spaces.

However, RPO has premature convergence issues due to random selection in the climbing process, thus leading to suboptimal solutions. To avoid this problem, the Lagrange Polynomial function is included in Red Panda Optimization (LPRPO), which effectively solves the premature convergence problem and enhances the global search capabilities.

The architecture of the Lagrange Polynomial-based Red Panda Optimization-based Logistic Softmax-Tree Hierarchical Deep Convolutional Neural Network (LPRPO-LSTDCNN) is depicted in Figure 2.

First, for the input (N), the weight value of the LSTDCNN classifier is initialized as follows,

- Weight Initialization Using LPRPO

To improve the performance of the classifier in detecting SCA, the weight values are initialized using the LPRPO optimization algorithm. The Red Panda Optimizer (RPO) is capable of selecting the required weights using the foraging strategy and tree-climbing behavior.

However, the random selection of the tree is made during the climbing phase, which affects the position updation of red pandas, thus leading to premature convergence and affecting the optimal weight values for classification.

Thus, to mitigate this issue, the Lagrange Polynomial (LP) technique that interpolates the Red Panda (RP) position is used instead of the random position. The LPRPO algorithm is explained in detail as follows,

- Population Initialization

The population of RP, which is the weight value of LSTDCNN, is randomly initialized as,

$$Y = \begin{bmatrix} Y_1 \\ \vdots \\ Y_q \\ \vdots \\ Y_J \end{bmatrix} = \begin{bmatrix} y_{1,1} & \cdots & y_{1,p} & \cdots & y_{1,G} \\ \vdots & \ddots & \vdots & \iddots & \vdots \\ y_{q,1} & \cdots & y_{q,p} & \cdots & y_{q,G} \\ \vdots & \iddots & \vdots & \ddots & \vdots \\ y_{J,1} & \cdots & y_{J,p} & \cdots & y_{J,G} \end{bmatrix}_{J \times G} \quad (29)$$

Where, (Y) is the population matrix of RP (weight values), which is specified in equation (29), (Y_q) is the (q^{th}) RP, $(y_{q,p})$ is the position of RP with (p^{th}) dimension, (J) is the number of Red Panda, and (G) is the number of problem variables. Now, the initial position of RP is initialized in equation (30),

$$y_{q,p} = lb_p + n\left(ub_p - lb_p\right) \quad (30)$$

Where, (lb_p, ub_p) are the lower bound and upper bound values of the (p^{th}) dimension of the Red Panda, and (n) is the random variable. The current position $(y_{q,p})$ gets updated regarding the fitness function.

- Fitness

Regarding maximum classification accuracy (K), the fitness value (ς), which is used to update the position of the RP, is calculated as given in equation (31),

$$\varsigma = \max(K) \quad (31)$$

Thus, by using (ς), the weight values are selected from the population matrix (Y).

- Position Update

In search of food, the Red Panda uses two strategies for position updation. The two strategies are foraging behavior and climbing behavior. The position update is as follows:

- Foraging

The RP has the ability to search the food by the use of hearing, smell, and vision. Thus, for the selected food source (I) of the RP and constant value (z), the position of RP gets updated as,

$$y^*_{q,p} = y_{q,p} + \left[I\left(y_{q,p}\right) - z\right] \quad (32)$$

Where, $(y^*_{q,p})$ is the new position of the RP, which is indicated in equation (32).

RESEARCH ARTICLE

- Climbing

After obtaining the food, the RP climbs the tree and rests on it. At this stage, the RP chooses the tree randomly, which affects the position update. Hence, LP, which compares and interpolates the tree value and avoids random selection, is used. The LP (τ) is calculated in the below equation (33),

$$\tau = \prod_{J} \frac{O - \overline{O}}{\overline{O} - O''} \quad (33)$$

Where, $(O, \overline{O}, O'')$ are the tree values that are near to the RP. The position of RP by the climbing behavior is updated as,

$$y''_{q,p} = y^{*}_{q,p} + \left[\frac{lb_p + \tau(ub_p - lb_p)}{x} \right] \quad (34)$$

Where, $(y''_{q,p})$ is the newly updated position of RP after climbing, which is represented in the equation (34), and (x) is the iteration value with maximum iteration (X).

Finally, by updating the positions of RP, the weight value $(\hat{y})$ is selected. The pseudocode for the LPRPO algorithm is given in algorithm 2.

Input: Random weights (y)

Output: Selected weight value $(\hat{y})$

Begin

Initialize population, iteration (x, X)

Evaluate random position of Red Panda

$y_{q,p} = lb_p + n(ub_p - lb_p)$

Calculate fitness $\varsigma = \max(K)$

While $(x \le X)$

//Foraging updation

For $(y^{*}_{q,p})$

Search food and move $y^{*}_{q,p} = y_{q,p} + [I(y_{q,p}) - z]$

If $\langle \varsigma(y^{*}_{q,p}) < \varsigma(y_{q,p}) \rangle$

Update new position $(\hat{y})$

Else

Original position $(y_{q,p})$

End if

End for

// Climbing behavior

For $(y''_{q,p})$

Calculate $\tau = \sum_{J} \frac{O - \overline{O}}{\overline{O} - O''}$

Climb and rest $y''_{q,p} = y^{*}_{q,p} + \left[\frac{lb_p + \tau(ub_p - lb_p)}{x} \right]$

If $\langle \varsigma(y''_{q,p}) < \varsigma(y_{q,p}) \rangle$

Best solution $(\hat{y})$

Else

Same position $(y_{q,p})$

End if

End for

End while

Return Selected weight value $(\hat{y})$

End

Algorithm 2 LPRPO Algorithm

These optimal weight values $(\hat{y})$ are further used in the classification for SCA detection. Further, along with the optimal weight values $(\hat{y})$, the reduced features (N) are given as input to the classification model for types of SCA. The classifier has multiple nodes connected in a tree shape. The nodes are Root Node (R), Branch Nodes (R''), and Leaf Nodes $(\hat{R})$. The SCA classification is described as follows,

- Root Node

The input (N) is passed into the classifier for super-class classification, and this phase is known as Root Node (R). The layers present in (R) are given below,

- Convolutional Layer

The Convolutional Layer (A) initializes the weights $(\hat{y})$ obtained from the LPRPO algorithm and multiplies them with

RESEARCH ARTICLE

the input data. The value of (N) is convolved and activated using the Rectified Linear Unit (ReLU) activation function. The output of (A) is expressed in the equation (35).

$$A = \left[(N * \hat{y}) + t\right] \times \xi \tag{35}$$

$$\xi = \max(0, N) \tag{36}$$

Where, (t) is the bias value of the convolutional layer and (ξ) defines the ReLU activation function, which is given in equation (36). Now, (A) is passed to the pooling layer.

- Pooling Layer

The Pooling Layer (L) reduces the feature count and helps in increasing the training speed. The important features needed for SCA detection are chosen by this layer (L) and are equated in the equation (37),

$$L'' = \max(A) \tag{37}$$

The pooled output (L'') is given to the fully connected layer.

- Fully-Connected Layer

In the Fully-Connected Layer (W), each neuron present in (L'') is connected to form a single neuron. This layer helps in detecting certain features and is used for predicting the respective class from the input feature. In equation (38), the output of a fully connected layer $(\dddot{W})$ is derived as,

$$\dddot{W} = \left[\sum (L'' * \hat{y})\right] + t \tag{38}$$

Where, $(\hat{y})$ is the optimized weight value of the fully-connected layer. Finally, (W) is activated using the LS activation function (ψ).

- Activation

Here, the LS activation function (ψ) that separates the data linearly to provide accurate classification is used. The (ψ), regarding the largest element $(\wp)$ in (W), is calculated in the below equation (39),

$$\psi = \frac{\wp \times \exp^{W}}{1 + \sum \exp^{W}} \tag{39}$$

In equations (40) and (41), the output (Q) of the Root Node (R) is obtained as given below,

$$Q = W * \psi \tag{40}$$

$$Q \rightarrow (Q_1, Q_2) \tag{41}$$

Where, (Q_1) and (Q_2) are the superclass-classified outputs, and these outputs along with the input (N) are given for further classification into the Branch Node (R'').

- Branch Node

Here, in the Branch Node (R''), the deep convolution takes place as seen in the Root Node. The input data (Q_1) is sent to the Convolutional Layer (A^*), Pooling Layer (L^*), and Fully-Connected Layer (W^*) and are finally activated to give the output (Q^*).

Similarly, for input (Q_2), the output (Q'') is obtained by passing it into the Convolutional Layer (A''), Pooling Layer (L''), and Fully-Connected Layer (W'') and is finally activated by Activation (ψ).

- Leaf Node

The Leaf Node $(\hat{R})$ has the final classified output $(\hat{Q})$, which is obtained from the outputs (Q^*) and (Q''). The final classified output $(\hat{Q})$ is represented in the equation (42),

$$\hat{Q} = \left\lfloor \hat{Q}_1, \hat{Q}_2, \hat{Q}_3, \hat{Q}_4, \hat{Q}_5, \hat{Q}_6 \right\rfloor \tag{42}$$

Where, $(\hat{Q}_1)$ is the data without SCA, and $(\hat{Q}_2)$, $(\hat{Q}_3)$, $(\hat{Q}_4)$, $(\hat{Q}_5)$, $(\hat{Q}_6)$ are the data with Timing, EM, SPA, DPA, and Template, respectively. In real-time, the authorized (hashcode-matched) user's encrypted data (T^*) is given to the SCA detection model.

The data is classified as per the classes, such as under attack and no-attack. The decryption of the data is explained in section 3.8.

3.8. Data Decryption

Regarding the Secret Key (ε) and Private Key (γ), the data (T^*) is decrypted using DH-ATM-RFICC method, which is explained in section 3.5. The decrypted data is calculated as,

$$\lambda = (T^* + \varepsilon) + \gamma \tag{43}$$

RESEARCH ARTICLE

Where, $(\hat{\lambda})$ is the decrypted data, which is represented in the equation (43). The decryption of the data takes place in the server, leading to a successful transaction. Thus, the proposed model effectively detected the types of SCA. The performance assessment of the proposed model is explained below in Section 4.

4. RESULTS

In this section, the performance of the proposed SCA detection is compared with the existing models. All experiments are done in the working platform of PYTHON with the AES_HD Dataset.

4.1. Dataset Description

The signal trace data is collected from the AES_HD Dataset, and the performance of the proposed method is compared with the existing models to show the effectiveness of the proposed work. The AES_HD dataset is a publicly available one that is used for SCA detection. From the dataset, for each class 15000 traces is taken such as Timing Attack, EM Attack, SPA, DPA and Template Attack. Among that, 80% of the data is used for training and 20% of the data is used for testing the SCA detection model.

4.2. Performance Analysis

Here, the performance of the proposed techniques, such as LPRPO-LSTDCNN, DH-ATM-RFICC, and IHE-SWIFFT are compared with the existing methods to prove the effectiveness of the proposed model in SCA detection.

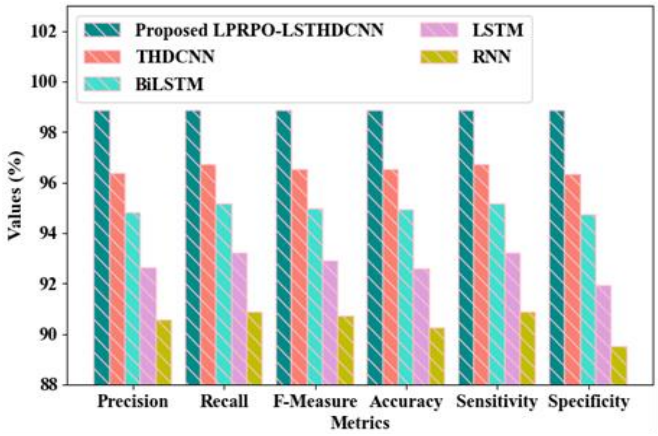

Figure 3 SCA Detection Results

The comparative analysis of the proposed LPRPO-LSTDCNN classifier and the existing THDCNN, Bidirectional Long Short-Term Memory (BiLSTM), Long Short-Term Memory (LSTM), and Recurrent Neural Network (RNN) in SCA detection is depicted in Figure 3. The performance metrics, such as Precision, Recall, F-measure, Accuracy, Sensitivity, and Specificity are used for comparison. The proposed classifier used the LS activation function to avoid slow convergence during SCA detection. Thus, the LPRPO-LSTDCNN method detected the SCA with 98.8659% Precision, 98.8873% Recall, 98.8766% F-Measure, 98.8748% Accuracy, 98.8873% Sensitivity, and 98.8625% Specificity. But the existing models obtained lower metrics value with an average of 93.5922% Precision, 93.9971% Recall, 93.7941% F-Measure, 93.5782% Accuracy, 93.9961% Sensitivity, and 93.1279% Specificity. Thus, the proposed model performed better than the existing classifiers in SCA detection.

Table 1 Comparative Analysis of LPRPO-LSTDCNN

Techniques	Training Time (ms)	TPR (%)	TNR (%)	PPV (%)	NPV (%)
Proposed LPRPO-LSTDCNN	37485	98.8873	98.8605	98.8666	98.8835
THDCNN	42698	96.7142	96.3274	96.3699	96.6863
BiLSTM	47125	95.1565	94.7192	94.8061	95.0608
LSTM	53624	93.2228	91.9161	92.6163	92.5726
RNN	58476	90.8997	89.5197	90.5723	89.8615

RESEARCH ARTICLE

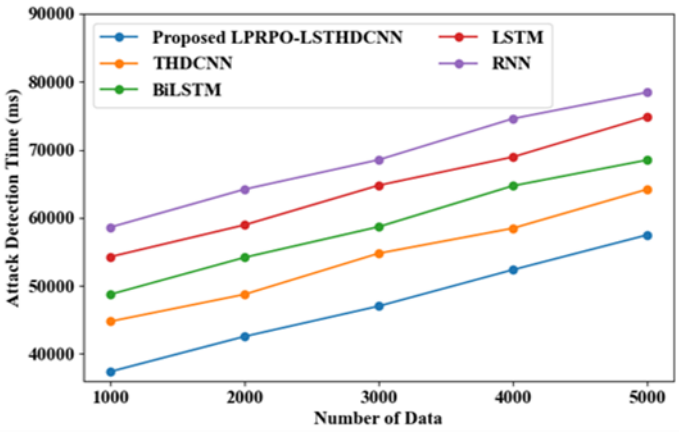

Figure 4 Comparative Assessment Regarding Attack Detection Time for a Number of Data

Figure 4 displays the comparative assessment of the proposed LPRPO-LSTDCNN and prevailing techniques in terms of attack detection time for a number of data sizes. Here, the proposed model utilizes LPRPO and LS activation functions for effective SCA detection. The proposed LPRPO-LSTDCNN took a low attack detection time of 37405ms for 1000 numbers of data and 57481ms for 5000 numbers of data. But, the prevailing techniques like THDCNN, BiLSTM, LSTM, and RNN obtained a high average attack detection time of 51615.5ms for 1000 numbers of data and 71498.75ms for 5000 numbers of data. Thus, the proposed LPRPO-LSTHDCNN provided high scalability for the increased number of transactions or data sizes. This made the proposed model suitable for large-scale deployments, which showed excellent real-world applicability. Thus, the reliability and scalability of the proposed model were proven.

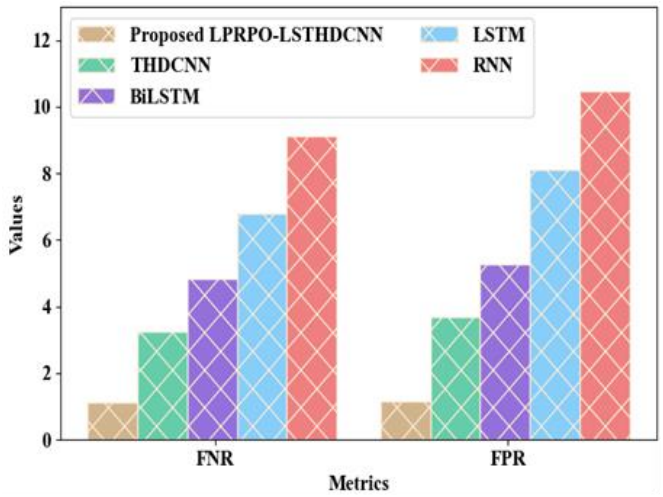

Figure 5 Performance Evaluation in Terms of FNR and FPR

Table 1 and Figure 5 show the comparative analysis of the proposed and existing THDCNN, BiLSTM, LSTM, and RNN classifiers regarding metrics, such as Training Time, True Positive Rate (TPR), True Negative Rate (TNR), Positive Predictive Value (PPV), Negative Predictive Value (NPV), False Negative Rate (FNR), and False Positive Rate (FPR). The proposed method classified the reduced feature in 37485ms with 98.8873% TPR, 98.8605% TNR, 98.8666% PPV, 98.8835% NPV, 1.1126% FNR, and 1.1372% FPR. The proposed model detected the SCA with lower Training Time, FNR, and FPR and with higher TPR, TNR, PPV, and NPV than the existing classifiers. This is because of the usage of the LS activation function and Optimized weight initialization in the proposed classifier. Hence, the proposed model outperformed the existing techniques in identifying the SCA.

Table 2 Performance Validation Across Different SCAs

Different SCAs	Performance metrics	Proposed LPRPO-LSTDCNN	THDCNN	BiLSTM	LSTM	RNN
Timing	Accuracy (%)	98.65	96.63	94.85	92.65	89.65
	Precision (%)	98.47	96.85	93.69	91.23	88.45
	Recall (%)	98.23	95.63	93.84	90.56	87.54
	F-Measure (%)	98.63	96.84	94.58	91.65	89.65
EM	Accuracy (%)	97.89	96.88	94.25	92.32	90.78
	Precision (%)	98.65	95.36	93.62	91.54	89.65
	Recall (%)	98.12	96.12	93.25	90.65	88.45
	F-Measure (%)	97.25	96.85	93.26	91.56	89.25
SPA	Accuracy (%)	98.63	96.75	94.15	92.84	89.84

RESEARCH ARTICLE

	Precision (%)	97.84	96.23	94.25	91.85	88.47
	Recall (%)	98.56	95.12	93.62	91.23	89.62
	F-Measure (%)	97.96	96.89	93.45	92.21	88.95
DPA	Accuracy (%)	98.74	96.99	94.85	92.32	90.65
	Precision (%)	97.98	96.88	93.65	91.56	90.24
	Recall (%)	98.56	96.84	93.25	92.26	89.62
	F-Measure (%)	98.47	95.65	93.32	91.87	88.48
Template	Accuracy (%)	98.89	96.32	94.84	92.48	89.78
	Precision (%)	98.47	95.84	93.22	92.32	89.12
	Recall (%)	97.58	96.20	94.20	91.21	87.21
	F-Measure (%)	98.58	95.65	93.05	91.89	88.95

Table 2 shows the performance validation of the proposed model and conventional techniques across different SCAs. Here, the proposed model provided accurate and effective outcomes for SCA detection. The proposed LPRPO-LSTDCNN achieved a high accuracy of 98.65%, 97.89%, 98.63%, 98.74%, and 98.89% for timing attacks, EM attacks, SPA, DPA, and template attacks, respectively. Also, the proposed LPRPO-LSTDCNN obtained high precision, recall, and F-measure across different SCAs. But the existing LSTM attained a precision of 91.23%, 91.54%, 91.85%, 91.56%, and 92.32% for timing attacks, EM attacks, SPA, DPA, and template attacks, which were lesser than the proposed technique.

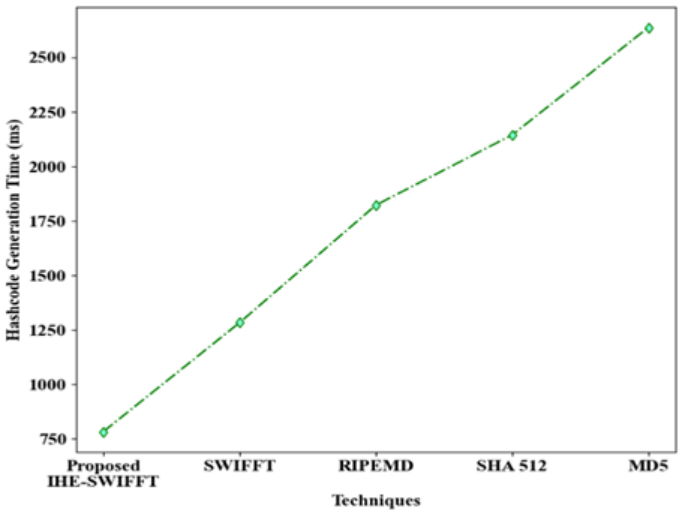

Figure 6 Comparative Analysis of IHE-SWIFFT

Likewise, the prevailing RNN attained a low accuracy, precision, recall, and F-measure of 90.65%, 90.24%, 89.62%, and 88.48% for DPA attacks, respectively. Similarly, all the existing techniques like THDCNN and BiLSTM obtained poor performance metrics for different SCAs. Here, the LPRPO and LS activation functions were modified with THDCNN for improving the SCA detection accuracy. Thus, the results proved the trustworthiness of the proposed technique.

The performance of the proposed IHE-SWIFFT hashing algorithm is compared with the existing SWIFFT, RACE Integrity Primitives Evaluation Message Digest (RIPEMD), Secure Hash Algorithm 512 (SHA512), and Message Digest 5 (MD5) hashing methods. As described in Figure 6, the hashcode to verify the authenticated user was generated by the proposed IHE-SWIFFT model in 638ms, whereas the existing SWIFFT, RIPEMD, SHA512, and MD5 generated hashcode in 1254ms, 1687ms, 1968ms, and 2358ms, which are higher than the proposed model. In the proposed IHE-SWIFFT model, the hashcode was generated using the IHE collision-resistance technique. Thus, the proposed model generated the hashcode more quickly than the existing methods and showed better performance in hashcode generation.

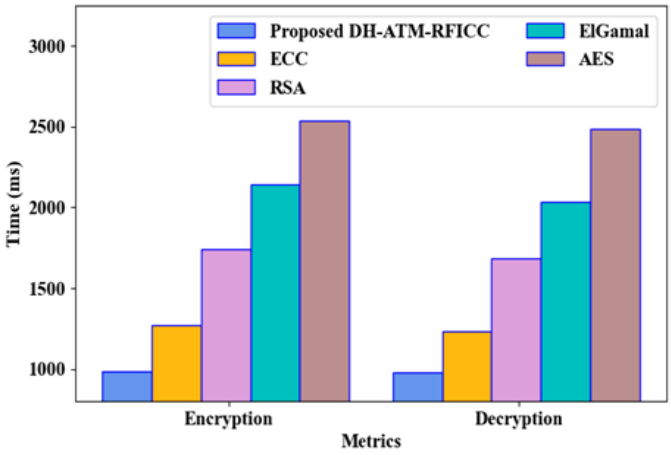

Figure 7 Comparison of Encryption Time and Decryption Time

RESEARCH ARTICLE

Metrics, such as Encryption Time and Decryption Time for the proposed DH-ATM-RFICC method and the existing ECC, Rivest Shamir Adleman (RSA), ElGamal, and Advanced Encryption Standard (AES) are compared as depicted in Figure 7. To secure the data, the Secret Key generated using the DH method and the Private Key generated using the ATM techniques were used in the proposed model. Hence, the proposed DH-ATM-RFICC technique encrypted and decrypted the data in 987ms and 978ms, respectively. However, the ECC, RSA, ElGamal, and AES models encrypted the data in 1268ms, 1742ms, 2145ms, and 2536ms, respectively, and decrypted the data on the receiver side in 1235ms, 1684ms, 2034ms, and 2487ms, respectively, which are higher than the proposed Encryption and Decryption Time. Thus, the proposed technique secured the data more effectively than the existing methods.

Table 3 Comparative Analysis of DH-ATM-RFICC

Methods	Memory Usage on Encryption (kb)	Memory Usage on Decryption (kb)
Proposed DH-ATM-RFICC	1287465324	2546105062
ECC	2365725478	3126503254
RSA	3216463254	4128921473
ElGamal	4781220412	5872505208
AES	5421602503	6489208401

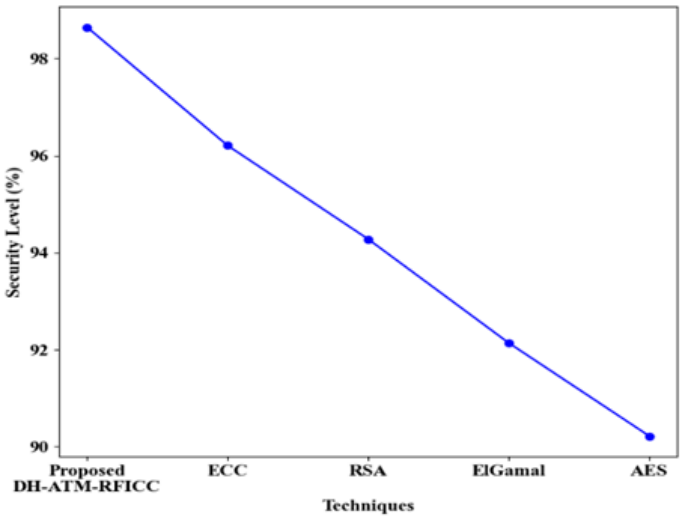

Figure 8 Graphical Representation Regarding Security Level

In the proposed model, the Public Key and Private Key along with the Secret Key were used for the encryption and decryption of signal traces. This was done for the purpose of data security. Hence, the proposed model achieved a Security Level of 98.65% and used memory of 1287465324kb on encryption and 2546105062kb on decryption. The proposed DH-ATM-RFICC was then compared with the existing ECC, RSA, ElGamal, and AES models as shown in Table 3 and Figure 8. These existing ECC, RSA, ElGamal, and AES models obtained a Security Level of 96.21%, 94.28%, 92.14%, and 90.22%, respectively, and higher memory usage during data encryption and decryption. Hence, it is proved that the proposed model outperformed the existing models in data security.

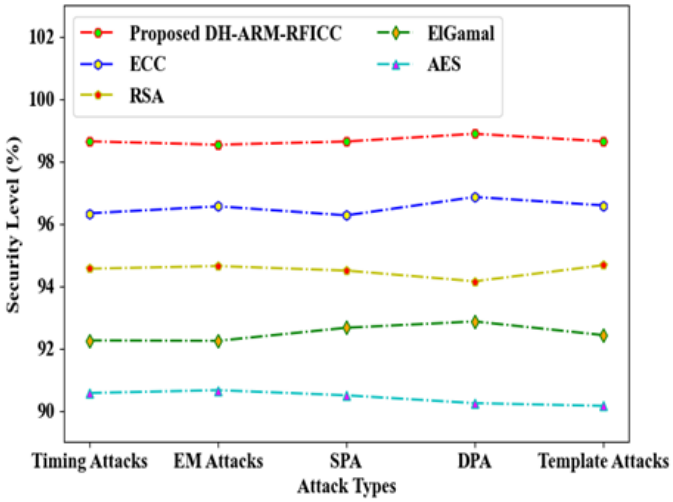

Figure 9 Security Level Analysis Across Different SCAs

Security level analysis across different SCAs of the proposed DH-ARM-RFICC and prevailing techniques is shown in Figure 9. Here, the proposed DH-ARM-RFICC achieved high-security levels of 98.63%, 98.52%, 98.62%, 98.87%, and 98.63% for timing attacks, EM attacks, SPA, DPS, and template attacks, respectively. The Asymmetric Tent Map and Robust Frobenius Isogenies-based curves are employed in DH-ARM-RFICC for providing enhanced security level. The existing ECC, RSA, ElGamal, and AES attained low-security levels of 96.57%, 94.65%, 92.41%, and 90.14%, respectively for template attacks. Likewise, the conventional techniques obtained poor security levels for different types of SCAs. Thus, the results proved the robustness of the proposed DH-ARM-RFICC in preventing key leakage.

Graphical representation of the proposed GPBC-EWT and conventional techniques like EWT, STFT, CWT, and Discrete Wavelet Transform (DWT) are displayed in Figure 10. Here, the proposed GPBC-EWT achieved a low Mean Squared Error (MSE) of 0.0289 and a high Signal to Noise Ratio (SNR) of 54.3694. Likewise, the conventional EWT, STFT, CWT, and DWT obtained a high average MSE of 6.8354 and a low average SNR of 40.2415. Thus, the results proved that the proposed GPBC-EWT provided better performance for feature extraction than other existing techniques with the help

RESEARCH ARTICLE

of Gini Point Biserial Correlation, thereby demonstrating the reliability of the proposed model.

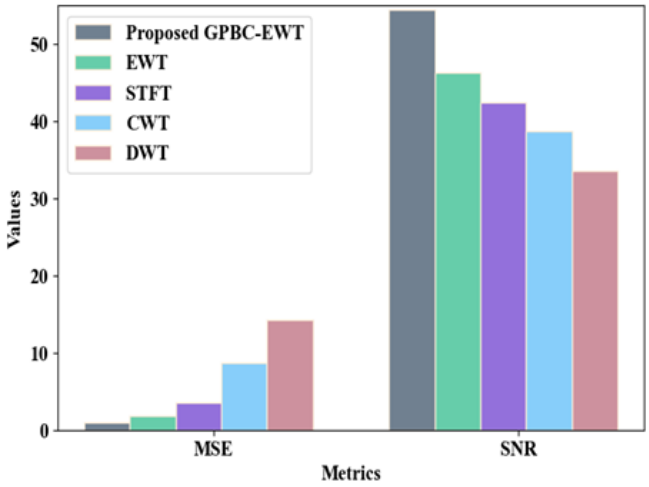

Figure 10 Graphical Representation Regarding MSE and SNR

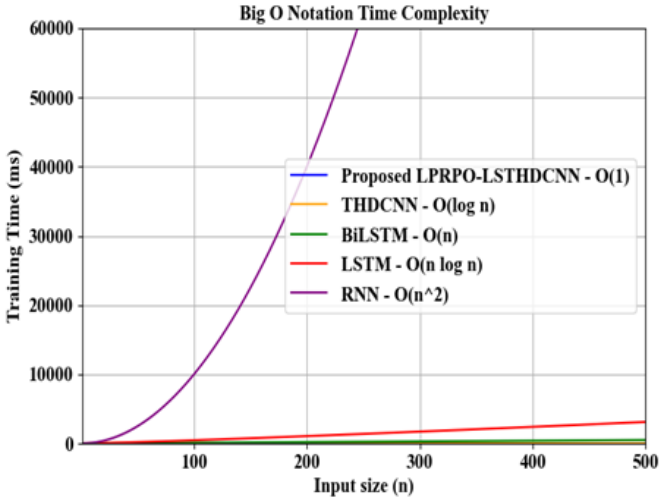

Figure 11 Complexity Analysis of the Training Time

Figure 11 displays the training time complexity of the proposed LPRPO-LSTDCNN and prevailing techniques using Big O notation. Generally, Big O notation is employed to define the upper bound on the running time of an algorithm that gives a theoretical estimate of the algorithm's performance with input size. Here, the proposed LPRPO-LSTDCNN had O(1) training time complexity, where the running time remains constant due to the inclusion of LPROP-based weight initialization and LS activation function. But, the existing LSTM had a training time complexity of O(n log n), which means that the LSTM's running time increased linearly with the input size. Also, the prevailing RNN had a training time complexity of O(n^2), which means that the RNN's running time increased quadratically with the input size. Likewise, conventional techniques like THDCNN and BiLSTM had high time complexities. Thus, the proposed LPRPO-LSTDCNN had low time complexity, and it effectively detected the SCAs than the prevailing techniques.

Table 4 Comparative Study Regarding Related Works

Study	Method	Accuracy (%)	Training Time (ms)	FPR (%)
Proposed Work	LPRPO-LSTDCNN	98.8748	37485	1.1372
[26]	WHISPER tool	94.68	-	4.36
[27]	CNN	67.91	321000	2.15
[28]	RF	92	82810	-
[29]	CNN	91	-	1.93
[30]	BNN	93	-	-

The proposed work is compared with the existing models in SCA detection as shown in Table 4. The existing works used multiple machine learning tools, namely WHISPER, Random Forest (RF), CNN, and Binarized Neural Network (BNN) for classification. The proposed model preprocessed the encrypted signal traces. Then, the features were extracted and reduced for further processing. Finally, the LPRPO-LSTDCNN classifier was used for classifying the SCA. Thus, the proposed model detected the attack in 37485ms with 98.8748% accuracy and 1.1372% FPR. However, [26] could not detect the attack in large data, thus producing a higher FPR of 4.36%. Similarly, [27] and [28] reduced the attack detection with an accuracy of 67.91% and 92%, respectively, since the Public Key was accessed by all the users. In [29], all the extracted features were used for classification, which delayed the attack detection with a FPR of 1.93%. Also, [30] did not secure the data, which reduced the SCA detection with 93% Accuracy. Hence, the proposed model achieved better performance in SCA detection.

5. DISCUSSION

In the proposed framework, the LPRPO-LSTDCNN classifier is modelled to perform SCA detection. With the help of LPRPO-based weight initialization and LS activation function, the proposed LPRPO-LSTDCNN achieved a high accuracy, precision, and recall of 98.87%, 98.86%, and 98.88% which demonstrated the efficiency and reliability of the proposed model. Likewise, DH-ATM-RFICC is employed for secure transactions. Due to the inclusion of RFI-based curves and ATB-based private key generation, the proposed DH-ATM-RFICC obtained a high-security level of 98.65% and low encryption time of 987ms, which proved the enhanced security of the proposed model. Similarly, the IHE-

RESEARCH ARTICLE

SWIFFT was used for generating the hashcode; here, the IHE was modified with SWIFFT for offering collision resistance, thus making the adversaries harder to recover the key from the acquired side channel traces. The proposed IHE-SWIFFT achieved a low hash generation time of 638ms, which proved the low time complexity. The GPBC-EWT achieved a low MSE of 0.0289 and a high SNR of 54.3694, which proved the reliability of the proposed model. Likewise, the BoDF-based feature reduction and DH-based secret key generation provided better performance. But the existing techniques provided low performance in SCA detection and had security leakage. Thus, the results proved that the proposed framework was better compared to conventional techniques.

6. CONCLUSION

This research has proposed an effective framework for side channel attack detection having better computational load which involved evaluating factors like data processing, memory, power consumption, and storage resources. First, the device was installed and the UUID was generated. Then, with the help of the Secret Key, the signal traces were encrypted by using DH-ATM-RFICC within 987ms and with a 98.65% Security Level. The hash code was then generated using IHE-SWIFFT algorithm within 638ms. Finally, the data under attacked was blocked, and the data with no attack was decrypted from the server in 978ms. Further, in our proposed system, the data was preprocessed and features were extracted using GPBC-EWT. The features were then reduced using BoDF and using the hybrid classifier LPRPO-LSTDCNN to detect the side channel attack. The classifier was trained in 37485ms. Thus, the SCA was detected with 98.86% Precision, 98.88% Recall, and 98.87% Accuracy. It is also observed that that the model effectively detected the SCA and adopted the secure data transfer which make harder to recover the secret key. Thus, it was suitable for large-scale deployments due to its scalability for the increased number of data sizes, which showed the excellent real-world applicability and it provided the continuous protection for a large number of devices and adapted to emerging threats.

6.1. Future Scope

Even though the proposed model detected the SCA efficiently, the prevention of Side-Channel Attacks was not considered in this model. Therefore, in the future, SCA mitigation measures will be concentrated to improve the performance of the proposed architecture.

REFERENCES

[1] Griswold-Steiner, Z. LeFevre, and A. Serwadda, “Smartphone speech privacy concerns from side-channel attacks on facial biomechanics,” Comput. Secur., vol. 100, pp. 1–19, 2021, doi: 10.1016/j.cose.2020.102110.

[2] D. Das, J. Danial, A. Golder, S. Ghosh, A. R. Wdhury, and S. Sen, “Deep Learning Side-Channel Attack Resilient AES-256 using Current Domain Signature Attenuation in 65nm CMOS,” Proc. Cust. Integr. Circuits Conf., vol. 2020-March, pp. 2–5, 2020, doi: 10.1109/CICC48029.2020.9075889.

[3] R. M. Tsoupidi, E. Troubitsyna, and P. Papadimitratos, “Thwarting code-reuse and side-channel attacks in embedded systems,” Comput. Secur., vol. 133, pp. 1–14, 2023, doi: 10.1016/j.cose.2023.103405.

[4] A. Johnson and R. Ward, “Introducing the ‘Unified Side Channel Attack - Model’ (USCA-M),” 8th Int. Symp. Digit. Forensics Secur. ISDFS 2020, pp. 1–9, 2020, doi: 10.1109/ISDFS49300.2020.9116291.

[5] R. Benadjila, E. Prouff, R. Strullu, E. Cagli, and C. Dumas, “Deep learning for side-channel analysis and introduction to ASCAD database,” J. Cryptogr. Eng., vol. 10, no. 2, pp. 163–188, 2020, doi: 10.1007/s13389-019-00220-8.

[6] Z. Tong, Z. Zhu, Z. Wang, L. Wang, Y. Zhang, and Y. Liu, “Cache side-channel attacks detection based on machine learning,” Proc. - 2020 IEEE 19th Int. Conf. Trust. Secur. Priv. Comput. Commun. Trust. 2020, pp. 919–926, 2020, doi: 10.1109/TrustCom50675.2020.00123.

[7] M. Salehi, G. De Borger, D. Hughes, and B. Crispo, “NemesisGuard: Mitigating interrupt latency side channel attacks with static binary rewriting,” Comput. Networks, vol. 205, pp. 1–11, 2022, doi: 10.1016/j.comnet.2021.108744.

[8] A. Garg and N. Karimian, “Leveraging deep cnn and transfer learning for side-channel attack,” Proc. - Int. Symp. Qual. Electron. Des. ISQED, vol. 2021-April, pp. 91–96, 2021, doi: 10.1109/ISQED51717.2021.9424305.

[9] A. Barenghi, L. Breveglieri, N. Izzo, and G. Pelosi, “Exploring Cortex-M Microarchitectural Side Channel Information Leakage,” IEEE Access, vol. 9, pp. 156507–156527, 2021, doi: 10.1109/ACCESS.2021.3124761.

[10] A. Akram, M. Mushtaq, M. K. Bhatti, V. Lapotre, and G. Gogniat, “Meet the Sherlock Holmes’ of Side Channel Leakage: A Survey of Cache SCA Detection Techniques,” IEEE Access, vol. 8, pp. 70836–70860, 2020, doi: 10.1109/ACCESS.2020.2980522.

[11] C. Jin and Y. Zhou, “Enhancing non-profiled side-channel attacks by time-frequency analysis,” Cybersecurity, vol. 6, no. 1, pp. 1–26, 2023, doi: 10.1186/s42400-023-00149-w.

[12] J. Galbally, “A new Foe in biometrics: A narrative review of side-channel attacks,” Comput. Secur., vol. 96, pp. 1–17, 2020, doi: 10.1016/j.cose.2020.101902.

[13] T. Miki, N. Miura, H. Sonoda, K. Mizuta, and M. Nagata, “A Random Interrupt Dithering SAR Technique for Secure ADC against Reference-Charge Side-Channel Attack,” IEEE Trans. Circuits Syst. II Express Briefs, vol. 67, no. 1, pp. 14–18, 2020, doi: 10.1109/TCSII.2019.2901534.

[14] S. Liu and W. Yi, “Task parameters analysis in schedule-based timing side-channel attack,” IEEE Access, vol. 8, pp. 157103–157115, 2020, doi: 10.1109/ACCESS.2020.3019323.

[15] A. R. Javed, M. O. Beg, M. Asim, T. Baker, and A. H. Al-Bayatti, “AlphaLogger: detecting motion-based side-channel attack using smartphone keystrokes,” J. Ambient Intell. Humaniz. Comput., vol. 14, no. 5, pp. 4869–4882, 2023, doi: 10.1007/s12652-020-01770-0.

[16] L. Zhang, X. Xing, J. Fan, Z. Wang, and S. Wang, “Multilabel Deep Learning-Based Side-Channel Attack,” IEEE Trans. Comput. Des. Integr. Circuits Syst., vol. 40, no. 6, pp. 1207–1216, 2021, doi: 10.1109/TCAD.2020.3033495.

[17] S. Ghandali, S. Ghandali, and S. Tehranipoor, “Deep K-TSVM: A Novel Profiled Power Side-Channel Attack on AES-128,” IEEE Access, vol. 9, pp. 136448–136458, 2021, doi: 10.1109/ACCESS.2021.3117761.

[18] T. T. Tsai, S. S. Huang, Y. M. Tseng, Y. H. Chuang, and Y. H. Hung, “Leakage-Resilient Certificate-Based Authenticated Key Exchange Protocol,” IEEE Open J. Comput. Soc., vol. 3, pp. 137–148, 2022, doi: 10.1109/OJCS.2022.3198073.

[19] D. Kwon, H. Kim, and S. Hong, “Non-Profiled Deep Learning-Based Side-Channel Preprocessing with Autoencoders,” IEEE Access, vol. 9, pp. 57692–57703, 2021, doi: 10.1109/ACCESS.2021.3072653.

[20] Y. Xiang, Y. Xu, Y. Li, W. Ma, Q. Xuan, and Y. Liu, “Side-Channel

RESEARCH ARTICLE

Gray-Box Attack for DNNs," IEEE Trans. Circuits Syst. II Express Briefs, vol. 68, no. 1, pp. 501–505, 2021, doi: 10.1109/TCSII.2020.3012005.

[21] U. Rioja, L. Batina, J. L. Flores, and I. Armendariz, "Auto-tune POIs: Estimation of distribution algorithms for efficient side-channel analysis," Comput. Networks, vol. 198, pp. 1–19, 2021, doi: 10.1016/j.comnet.2021.108405.

[22] A. A. J. Al-Hchaimi, N. Bin Sulaiman, M. A. Bin Mustafa, M. N. Bin Mohtar, S. L. B. Mohd Hassan, and Y. R. Muhsen, "A comprehensive evaluation approach for efficient countermeasure techniques against timing side-channel attack on MPSoC-based IoT using multi-criteria decision-making methods," Egypt. Informatics J., vol. 24, no. 2, pp. 351–364, 2023, doi: 10.1016/j.eij.2023.05.005.

[23] S. D. P. Tran, B. Seok, and C. Lee, "HANMRE - An authenticated encryption secure against side-channel attacks for nonce-misuse and lightweight approaches," Appl. Soft Comput. J., vol. 97, pp. 1–13, 2020, doi: 10.1016/j.asoc.2020.106663.

[24] S. Paguada, L. Batina, and I. Armendariz, "Toward practical autoencoder-based side-channel analysis evaluations," Comput. Networks, vol. 196, pp. 1–17, 2021, doi: 10.1016/j.comnet.2021.108230.

[25] Y.-S. Won, X. Hou, Dirmanto Jap, J. Breier, and S. Bhasin, "Back to the Basics : Seamless Integration of Side-Channel Pre-Processing in Deep Neural Networks," IEEE Trans. Inf. FORENSICS Secur., vol. 16, pp. 3215–3227, 2021.

[26] M. Mushtaq et al., "WHISPER: A tool for run-time detection of side-channel attacks," IEEE Access, vol. 8, pp. 83871–83900, 2020, doi: 10.1109/ACCESS.2020.2988370.

[27] N. Mukhtar, A. P. Fournaris, T. M. Khan, C. Dimopoulos, and Y. Kong, "Improved hybrid approach for side-channel analysis using efficient convolutional neural network and dimensionality reduction," IEEE Access, vol. 8, pp. 184298–184311, 2020, doi: 10.1109/ACCESS.2020.3029206.

[28] D. Chen et al., "MAGLeak: A Learning-Based Side-Channel Attack for Password Recognition with Multiple Sensors in IIoT Environment," IEEE Trans. Ind. Informatics, vol. 18, no. 1, pp. 467–476, 2022, doi: 10.1109/TII.2020.3045161.

[29] H. Wang and E. Dubrova, "Tandem Deep Learning Side-Channel Attack on FPGA Implementation of AES," SN Comput. Sci., vol. 2, no. 5, pp. 1–12, 2021, doi: 10.1007/s42979-021-00755-w.

[30] Y. S. Won, D. G. Han, D. Jap, S. Bhasin, and J. Y. Park, "Non-Profiled Side-Channel Attack Based on Deep Learning Using Picture Trace," IEEE Access, vol. 9, pp. 22480–22492, 2021, doi: 10.1109/ACCESS.2021.3055833.

Authors

Prasath Vijayan received his B.Tech in Computer Science and Engineering from B.C.E.T, Karaikal, Puducherry and M.Tech in Information Security from Pondicherry Engineering College, Puducherry. He is currently working as Assistant Professor at Perunthalaivar Kamarajar Institute of Engineering and Technology, Puducherry, Karaikal, India. He has 14 years of academic excellence in well-known institutes. He has various publications at recognized international journals and conferences. His research interests include Web service security, Deep learning, Side channel analysis and Application of Machine Learning.

Dr Sudalaimuthu T is a Professor in the Department of Computer Science and Engineering, School of Engineering and Technology at Hindustan Institute of Technology and Science, Chennai, India. From the Hindustan Institute of Technology and Science in Chennai, India, he received his PhD degree. He is a certified Ethical Hacker. He has published more than 50 reputable international journals. He has won numerous honours throughout his career, including the Top Innovator Award and the Pearson Award for Best Teacher. His research interests include machine learning, grid and cloud computing, and cyber network security. He has lifetime memberships in IEEE, ACM, and CSI. He was awarded 10 patents for his inventions.

How to cite this article:

Prasath Vijayan, Sudalaimuthu T, "A Novel LPRPO-LSTDCNN Based Side Channel Attack Detection and Secure Data Transmission Framework Using DH-ATM-RFICC", International Journal of Computer Networks and Applications (IJCNA), 11(6), PP: 803-820, 2024, DOI: 10.22247/ijcna/2024/48.