ELLIPTIC CURVE CRYPTOGRAPHY ALGORITHM WITH RECURRENT NEURAL NETWORKS FOR ATTACK DETECTION IN INDUSTRIAL IOT

Bebin Josey T [1], D.S.Misbha [2]

[1] Department of Computer Science, Nesamony Memorial Christian College Marthandam, Manonmanium Sundaranar University, India.
[2] Department of Computer Applications, Nesamony Memorial Christian College, Marthandam, Manonmanium Sundaranar University ,India.

ABSTRACT

The increasing use of Industrial Internet of Things (IIoT) devices has brought about new security vulnerabilities, emphasizing the need to create strong and effective security solutions. This research proposes a two-layered approach to enhance security in IIoT networks by combining lightweight encryption and RNN-based attack detection. The first layer utilizes Improved Elliptic Curve Cryptography (IECC), a novel encryption scheme tailored for IIoT devices with limited computational resources. IECC employs a Modified Windowed Method (MWM) to optimize key generation, reducing computational overhead and enabling efficient secure data transmission between IIoT sensors and gateways. The second layer employs a Recurrent Neural Network (RNN) for real-time attack detection. The RNN model is trained on a comprehensive dataset of IIoT network traffic, including instances of Distributed Denial of Service (DDoS), Man-in-the-Middle (MitM), ransomware attacks, and normal communications. The RNN effectively extracts contextual features from IIoT nodes and accurately predicts and classifies potential attacks. The effectiveness of the proposed two-layered approach is evaluated using three phases. The first phase compares the computational efficiency of IECC to established cryptographic algorithms including RSA, AES, DSA, Diffie-Hellman, SHA-256 and ECDSA. IECC outperforms all competitors in key generation speed, encryption and decryption time, throughput, memory usage, information loss, and overall processing time. The second phase evaluates the prediction accuracy of the RNN model compared to other AI-based models DNNs, DBNs, RBFNs, and LSTM networks. The proposed RNN achieves the highest overall accuracy of 96.4%, specificity of 96.5%, precision of 95.2%, and recall of 96.8%, and the lowest false positive of 3.2% and false negative rates of 3.1%.

KEYWORDS

Industrial Internet of Things (IIoT), Improved Elliptic Curve Cryptography (IECC), RNN, Cryptographic Algorithms, Attack Detection, Security

1. INTRODUCTION

The Industrial Internet of Things (IIoT) has revolutionized industrial processes by introducing a new level of connectivity and efficiency [1][2]. However, this increased connectivity also presents significant security vulnerabilities, making it imperative to safeguard the integrity and confidentiality of data transmission in IIoT networks [3][4][5]. This research proposes a two-layered architecture designed to address these concerns and strengthen the security of IIoT networks. Traditional cryptographic solutions, particularly those based on Elliptic Curve Cryptography (ECC), face challenges in resource-constrained IIoT devices [6][7]. These challenges include computational complexity, key management issues, and susceptibility to various attacks [8]. Moreover, the constantly changing nature of cyber threats requires advanced

and adaptable detection methods to protect IIoT networks from new intrusions. Ongoing research in IIoT security frequently deals with the trade-off between strong cryptography and the limited resources of industrial devices [9]. Additionally, the effectiveness of attack detection mechanisms varies, with some existing solutions struggling to provide real-time identification and classification of evolving cyber threats [10]. There exists a critical need for an integrated and efficient solution that addresses both the cryptographic and detection aspects of IIoT security.

This research proposes a two-layered approach to strengthen IIoT security. The first layer introduces the Improved Elliptic Curve Cryptography (IECC), a novel encryption scheme personalized for lightweight environments. IECC addresses the limitations of traditional ECC by employing a Lightweight ECC Key Generation process based on the Modified Windowed Method (MWM). This optimization streamlines the scalar multiplication proces, reducing computational overhead and lightening key management challenges. IECC encompasses key exchange, digital signatures, and a secure data transmission scheme to ensure confidentiality and integrity in IIoT communication. The second layer uses a Recurrent Neural Network (RNN)-based attack detection system. This system conducts contextual feature analysis to extract crucial attributes from IIoT nodes. This research utilizes a specially designed dataset containing cases of Distributed Denial of Service (DDoS), Man-in-the-Middle (MitM), ransomware attacks, and normal communications to train and evaluate the RNN model's attack detection capabilities. The model architecture involves layers for input processing, embedding, recurrence, dropout, and output prediction.

The proposed two-layered architecture addresses the critical challenges in IIoT security by providing a holistic solution. Lightweight ECC ensures cryptographic protection in resource-constrained devices, overcoming computational overhead and key management issues. Simultaneously, the RNN-based attack detection layer enhances the resilience of IIoT networks by offering real-time identification and classification of potential attacks. This research contributes significantly to the advancement of industrial cybersecurity, offering an integrated solution for secure data transmission and effective attack detection in IIoT environments. The proposed methodology undergoes extensive validation, as outlined in the results and discussion section. The evaluation showcases the efficiency of IECC in secure data transmission and the accuracy of the RNN model in detecting and classifying various types of attacks. The promising results underscore the potential of the two-layered approach to significantly enhance the security of IIoT networks.

The following section discusses the recently developed attack detection models, Section 3 introduces the proposed two-layer architecture for Industrial Internet of Things (IIoT) security. Section 4 conducts a comparative analysis of the proposed IECC's computational efficiency against popular cryptographic algorithms and evaluates the predictive performance of the proposed method against existing popular attack prediction models. Finally, the research is concluded.

2. LITERATURE REVIEW

Several researchers have proposed deep learning-based IDS models that exhibit superior performance in identifying and classifying various attack types.

Albara Awajan [11] proposed a deep fully connected (FC) network architecture rooted in deep learning principles for IIoT network protection. The proposed IDS achieved an average accuracy of 93.74% in detecting various attacks, including Blackhole, Distributed Denial of Service, Opportunistic Service, Sinkhole, and Workhole.Hakan Can Altunay et al [12] introduced a hybrid IDS specifically designed for IIoT networks. Their proposed IDS utilizes a combination of

Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), and a hybrid CNN+LSTM model. Rigorous testing demonstrated the hybrid CNN+LSTM model's superior accuracy in intrusion detection, achieving rates of 93.21% for binary classification and 92.9% for multi-class classification.Joseph Bamidele Awotunde et al [13] proposed a deep learning model with rule-based feature selection for IIoT network intrusion detection. This approach addresses the challenge of gathering relevant information for intelligent IDS development. The proposed IDS achieved promising results in detecting anomalies in network traffic.Rayeesa Malik et al [14] introduced an enhanced IDS for IoT-based networks in traffic systems. Their proposed IDS utilized a deep belief network (DBN) as its core and demonstrated effectiveness in detecting various attack scenarios.Ho-Myung Kim et al [15] addressed the critical issue of malware detection in smart factories within the IIoT environment. They proposed an innovative solution using edge computing and deep learning techniques to enhance cybersecurity. The proposed malware detection system efficiently processed vast amounts of smart factory IIoT traffic information, demonstrating its potential for practical implementation.Abbas Yazdinejad et [16] al focused on accurate threat detection in edge devices within the IIoT framework. They proposed a parallel ensemble model for threat hunting that employed anomaly detection based on the behavior of IIoT edge devices. This approach demonstrated the feasibility of edge-based threat detection in IIoT environments.

Almaiah [17] offered a lightweight Hybrid Deep Learning-based model for the Industrial Internet of Medical Things. This model consisted of a two-layer security structure integrating blockchain for user and device authentication, and deep learning to predict potential attacks. The Variational AutoEncoder (VAE) technique and a Bidirectional Long Short-Term Memory intrusion detection model were employed for privacy and security respectively. Model training was conducted using the ToN-IoT datasets and IoT-Botnet, expanding the application of hybrid models to medical IIoT scenarios. In a study by Al-Abassi et al [18], a Deep Learning-Based Attack Detection system for IIoT networks was introduced. This innovative model aimed to create balanced representations from imbalanced datasets, using a Decision Tree (DT) and Deep Neural Network (DNN) to identify potential attacks. The model was trained and validated with Gas Pipeline (GP) and Secure Water Treatment (SWaT) datasets, extending the scope of deep learning applications in IIoT security.

While deep learning models have shown promising results in IIoT intrusion detection, the computational overhead associated with these models can pose challenges for resource-constrained IIoT devices.

3. PROPOSED METHODOLOGY

To address the security and integrity of industrial networks, a two-layered architecture is proposed for secure IIoT communication. The first layer, the lightweight secured data transmission layer, utilizes a novel encryption scheme called IECC to ensure secure data transfer to the cloud layer. The second layer, the RNN-based attack detection layer, employs an RNN model to predict potential network attacks and store attack details in the cloud server for future analysis. This comprehensive approach provides a robust security solution for IIoT networks, safeguarding sensitive data and ensuring reliable operation. The overall architecture of the proposed methodology is represented in the figure 1.

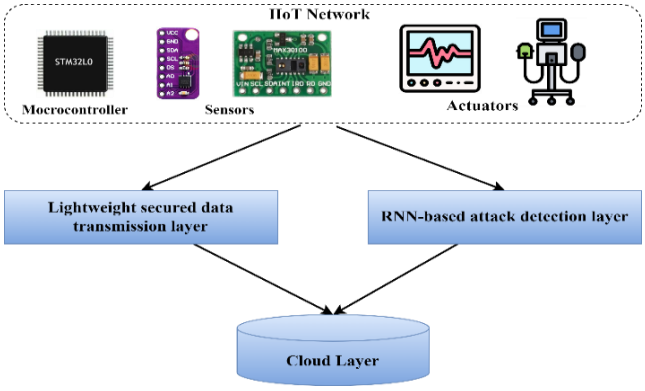

Figure 1.Overall process flow proposed a lightweight attack detection model for IIoT.

3.1 Lightweight secured data transmission layer

ECC has become a widely adopted cryptographic algorithm in IIoT systems [19]. However, the implementation of ECC in IIoT devices poses several challenges and limitations [20]. One of the primary concerns is the computational overhead associated with ECC operations. Resource-constrained IIoT devices often lack the processing power to handle the complex mathematical computations required for ECC, leading to performance degradation and increased energy consumption. Moreover, the implementation of ECC on these devices often requires specialized hardware or software libraries, which can further complicate the development and deployment process. Another challenge lies in the key management and generation process for ECC. IoT networks involve a large number of devices, each with its own public-private key pair. Managing and securing these keys becomes increasingly complex as the network grows. Following are the steps involved in the existing ECC.

3.1.1. Elliptic Curve Cryptography (ECC)

Elliptic Curve Cryptography (ECC) involves several key steps, beginning with the selection of an elliptic curve over a finite field [21]. The curve is typically represented by the equation $y^2 = x^3 + ax + b$, where a and b are constants. A base point G on the curve is chosen with a prime order n, such that $n \times G = \mathcal{O}$, the point at infinity. Following this, a private key d is randomly chosen, and the corresponding public key Q is computed as $Q = d \times G$. This establishes the foundation for key generation in ECC.

3.1.2. Key Generation

Firstly, a specific elliptic curve over a finite field is selected, characterized by parameters like the prime modulus (p), coefficients (a and b), and a base point (G(x, y)). This elliptic curve serves as the foundation for the key generation process [22][23]. The next step is to randomly generate a private key (d) within the range [1, n-1], where n represents the order of the base point G. This private key remains confidential and is fundamental to the security of the ECC system. Subsequently, the corresponding public key (Q) is computed using scalar multiplication, where Q = d * G. This process involves repeated additions of the base point G to itself d times, resulting in a point on the elliptic curve. Optionally, the public key can be compressed for more efficient storage by transmitting only the x-coordinate and a parity bit indicating the y-coordinate's parity. The final key pair, consisting of the private key (d) and the compressed or uncompressed public key (Q), is then utilized for secure cryptographic operations. The security of ECC relies on the complexity of the Elliptic Curve Discrete Logarithm Problem (ECDLP), making the private key

difficult to deduce from the public key. Proper implementation and careful validation of key properties are crucial for the robustness of ECC key generation.

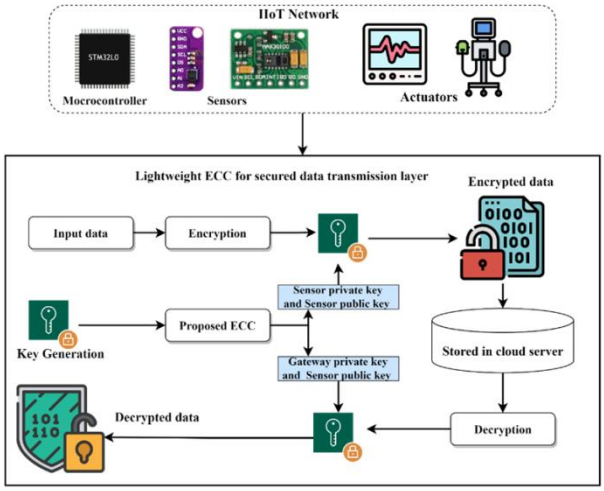

Figure 2. Process flow of secured data transmission

3.1.3. Key Exchange

For key exchange using ECC, each party generates a session key by selecting a random integer (k for Alice and l for Bob) and computing a point (P for Alice and Q for Bob). The shared secret is then independently calculated by both parties, involving themultiplication of the other party's point with their private key. This shared secret can be used as a symmetric key for secure communication.

3.1.4. Digital Signatures

Digital signatures in ECC involve a signing party (e.g., Alice) and a verifying party (e.g., Bob) [24]. Alice first hashes the message she wants to sign ($m = H(\text{message})$) and chooses a random integer k. She then computes a point $P = k \times G$ and calculates $r \equiv x(P) \pmod{n}$. The signing party ensures that r is not zero, recalculating k if needed. Subsequently, she computes $s \equiv k^{-1} \times (m + d \times r) \pmod{n}$, and the signature is formed as (r, s).

On the verifying side, Bob receives the message and signature and computes the hash of the message ($m = H(\text{message})$). He checks that r and s are within the appropriate range and then calculates $w \equiv s^{-1} \pmod{n}$. Bob further computes $u_1 = m \times w \pmod{n}$ and $u_2 = r \times w \pmod{n}$. By calculating a point $P = u_1 \times G + u_2 \times Q$, where Q is the public key of the signing party, Bob verifies if $r \equiv x(P) \pmod{n}$. If this holds, the signature is deemed valid.

To address the computational cost in resource-constrained like IIoT this research introduces a Lightweight ECC Key Generation based on the Modified Windowed Method (MWM). By using the MWM, the algorithm optimizes the scalar multiplication process, reducing computational complexity while maintaining the cryptographic integrity of the generated key pairs. The algorithm's significance lies in its applicability to a wide range of lightweight cryptographic scenarios, including IoT deployments and embedded systems. Algorithm 1 explains the process flow of MWM. In this method, multiples of the base point G are precomputed and stored to expedite the scalar multiplication. The algorithm initializes a result point R to the point at infinity and converts the scalar d into its binary representation. Subsequently, it performs the MWM, where the binary representation is processed in windows, and for each window, the algorithm efficiently adds the corresponding pre-computed multiple to the resulting point R based on the

extracted bits. This approach significantly reduces the number of point additions required during scalar multiplication, enhancing the computational efficiency of ECC key generation, especially in resource-constrained environments. The final result, R, represents the computed point on the elliptic curve and serves as the public key in the ECC key pair.

Algorithm 1 Scalar Multiplication using Modified Windowed Method

```
function scalar multiply(d,G,window size)
    Precompute multiples of the base point G
    precomputed ← [G]
    for i← 1 to 2window size − 1 do
        precomputed.append(add(precomputed[−1],G))
    end for
    Initialize R to the point at infinity
    R ← infinity point()
    Convert d to binary form
    binary d ← bin(d)[2 :]
    Perform modified windowed method
    for i← 0 to len(binary d) by window size do
window ← binary d[i: i+ window size][:: −1] ▷Extract a window of bits
        window value ← int(window,2)
        if window value != 0 then
            R ← add(R,precomputed[window value//2])
        end if
    end for
    return R
end function
```

The proposed lightweight ECC key generation process using the MWM involves several key steps to optimize the generation of key pairs in resource-constrained environments. Figure 2 shows the process flow of the proposed lightweight secure data transmission. Firstly, a lightweight elliptic curve with parameters (p, a, b, G) is carefully chosen to suit the constraints of the environment. Then, a private key (d) is generated using a lightweight pseudorandom number generator within the range [1, n-1], where n is the order of the base point G. The core of the algorithm lies in the Point Generation using the MWM for scalar multiplication. Precomputed multiples of the base point G are generated, and the private key is represented in binary form. For each window of bits, the algorithm efficiently adds the corresponding precomputed multiple to the result point R, significantly reducing the computational complexity of scalar multiplication. Optionally, the public key is compressed to minimize storage and transmission overhead. The final output consists of the key pair (d, Q), where Q is the computed public key, and adherence to ECC security requirements is emphasized throughout the process. This algorithm addresses the need for an efficient ECC key generation method tailored for lightweight environments while maintaining the security standards of ECC. Following are the steps involved in the proposed Lightweight ECC key generation process using MWM.

Algorithm 2 Lightweight ECC Key Generation Process using Modified Windowed Method

1. **Curve Selection:** Choose a lightweight elliptic curve with parameters (p, a, b, G) suitable for constrained environments.
2. **Random Key Generation:** Generate a private key (d) using a lightweight pseudorandom number generator in the range $[1, n-1]$, where n is the order of the base point G.
3. **Point Generation (Modified Windowed Method):** Compute the public key (Q) using the modified windowed method for scalar multiplication:
   a. Initialize R to the point at infinity (O).

b. Precompute multiples of the base point G: $G, 2G, 3G, \ldots, 2^{w-1}G$, where w is the window size.
c. Represent d in binary form: $d_{k-1}d_{k-2} \ldots d_1 d_0$.
d. For each window, starting from the most significant bits:
e. Extract the bits $d_i, d_{i-1}, \ldots, d_{i-w+1}$.
f. Multiply R by 2^w and add the appropriate precomputed multiple based on the extracted bits.
g. The final R is the computed point Q.

4. **Public Key Compression:** Compress the public key to reduce storage and transmission overhead:

Compressed public key: $Q_{\text{compressed}} = \left(x_Q, \text{parity}(y_Q)\right)$.

Uncompressed public key: $Q_{\text{uncompressed}} = (x_Q, y_Q)$.

5. **Output Key Pair:** The key pair is (d, Q). Ensure that the generated keys adhere to ECC security requirements.

3.1.5. Encryption Process

In the proposed IoT sensor data transmission scheme, the encryption process begins with the generation of a Lightweight ECC key pair by the IoT sensor, consisting of a private key (d_{sensor}) and a public key (Q_{sensor}), utilizing the MWM. The sensor securely transmits its public key to the gateway. Upon reception, the gateway generates its ECC key pair ($d_{\text{gateway}}, Q_{\text{gateway}}$) using the same Lightweight ECC with MWM. The shared secret (K_{shared}) is then calculated through scalar multiplication of the sensor's public key and the gateway's private key. Subsequently, a symmetric encryption key ($K_{\text{symmetric}}$) is derived from the shared secret using a key derivation function (KDF). The IoT sensor employs this symmetric key to encrypt its sensor data (M) using an encryption algorithm, resulting in the ciphertext (C). The encrypted data is then securely transmitted to the gateway, ensuring confidentiality and integrity during the communication.

Algorithm 3 IIoT Sensor Data Transmission Encryption

1: Input: Lightweight ECC key pair parameters, Sensor private key d_{sensor}, Sensor public key Q_{sensor}, MWM window size, Gateway public key Q_{gateway}
2: **Output:** Encrypted data C
3: **Sensor Side:**
4: Generate Lightweight ECC key pair: $(d_{\text{sensor}}, Q_{\text{sensor}}) \leftarrow$ MWMKeyGeneration()
5: Securely transmit $Q_{\text{sensor}}$to the gateway
6: **Gateway Side:**
7: Generate Lightweight ECC key pair: $(d_{\text{gateway}}, Q_{\text{gateway}}) \leftarrow$ MWMKeyGeneration()
8: Calculate shared secret: $K_{\text{shared}} \leftarrow$ ScalarMultiply($d_{\text{gateway}}, Q_{\text{sensor}}$,MWM window size)
9: Derive symmetric key: Ksymmetric ← KDF(Kshared)
10: **Encryption:**
11: Encrypt sensor data: $C \leftarrow$ Encrypt($M, K_{\text{symmetric}}$)
12: **Transmission:**
13: Transmit encrypted data C securely to the gateway

3.1.6. Decryption Process

Upon receiving the encrypted data (C), the gateway initiates the decryption process. The gateway uses its private key (d_{gateway}) and the sensor's public key (Q_{sensor}) in conjunction with the MWM for scalar multiplication to recompute the shared secret (K_{shared}). Using the same key derivation

function, the gateway derives the symmetric encryption key ($K_{symmetric}$). With the obtained symmetric key, the gateway decrypts the received ciphertext (C) utilizing a decryption algorithm, revealing the original sensor data (M). This comprehensive encryption and decryption process ensures secure and efficient communication between the IoT sensor and the gateway, using lightweight ECC with MWM for key generation and encryption. Regular key rotation further strengthens the security of the communication channel, addressing the dynamic nature of IoT security considerations.

Algorithm 4: IIoT Sensor Data Transmission Decryption

1: **Input:** Encrypted data C, Gateway private key $d_{gateway}$, Sensor public key Q_{sensor}, MWM window size
2: **Output:** Decrypted sensor data M
3: **Gateway Side:**
4: Calculate shared secret: $K_{shared} \leftarrow$ ScalarMultiply($d_{gateway}$,Q_{sensor},MWM window size)
5: Derive symmetric key: Ksymmetric $\leftarrow$ KDF(Kshared)
6: **Decryption:**
7: Decrypt encrypted data: $M \leftarrow$ Decrypt(C,$K_{symmetric}$)

3.2. Attack Prediction Model

3.2.1. Contextual feature analysis for attack detection in IIoT nodes

IIoT nodes typically possess various attributes, such as node address, location information, storage capacity, and processing power. Additionally, nodes engage in communication by exchanging packets, which contain timestamps, routing information, and payload data. By analyzing these attributes and communication patterns, contextual features can be extracted to characterize the behavior of each node.Each contextual feature exhibits a normal range of values for healthy nodes. Deviations from these normal ranges can indicate potential abnormalities or attacks. For instance, a node with unusually high computational cost, extended packet transmission times, or a significantly higher hop count compared to its peers may be compromised. To identify attacked nodes, threshold values are established for each contextual feature. These thresholds are typically determined based on a statistical analysis of expert knowledge of the IIoT network. If a node's contextual feature value falls outside its corresponding threshold, it is flagged as a potential attack candidate. By aggregating and analyzing the contextual features of all nodes in the IIoT network, a comprehensive assessment of network health can be obtained. Nodes with multiple contextual features deviating from their respective thresholds are considered highly likely to be compromised and require further investigation. After identifying the attack nodes, the RNN is used to classify the attacks. Figure 3 shows the process flow of theproposed attack prediction model.

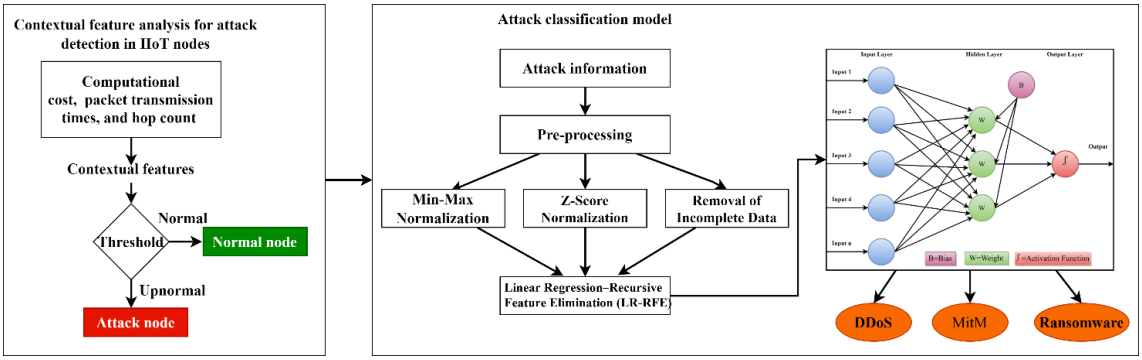

Figure 3. Process flow of proposed attack prediction model.

3.2.2. Classifying the Different Types of Attacks

Once attack nodes have been identified using the previous module, the next step is to classify the type of attack being perpetrated. This involves utilizing a trained RNN model to analze the contextual features and communication patterns of the attack nodes. Before employing the RNN model, the attack node data undergoes preprocessing to ensure its quality and suitability for machine learning. Normalization is a crucial step in preparing data for machine learning algorithms. It scales the features of the attack node data to a common range, preventing features with larger magnitudes from dominating the model's training process. After preprocessing, feature selection is performed to eliminate redundant and uncorrelated features that may introduce noise and degrade the classification accuracy of the RNN model. This process involves correlation analysis. Correlation analysis measures the strength and direction of the linear relationship between pairs of features. Features that exhibit high correlations with each other are considered redundant and can be eliminated, as they provide similar information. The pre-processed and feature-selected attack node data is then fed into the trained RNN model for classification. The RNN model, with its ability to capture temporal dependencies in the data, effectively analyses the sequential patterns and relationships within the contextual features, enabling it to distinguish between different types of attacks.

3.2.3. Data Preprocessing

Data preprocessing is a crucial step in preparing data for machine learning algorithms. In the context of classifying attack types in IIoT networks, data preprocessing involves several essential steps:

Min-Max Normalization: This technique scales the data to a range between 0 and 1. The formula for min-max normalization is:

$$x_i = \frac{\big(x_i - min(x)\big)}{\big(max(x) - min(x)\big)} \quad (1)$$

where x_i is the value of the ith feature, min(x) is the minimum value of the feature, and max(x) is the maximum value of the feature.

Z-Score Normalization: This method normalizes the data to have a mean of 0 and a standard deviation of 1. The formula for z-score normalization is:

$$x_i = \frac{\big(x_i - \mu(x)\big)}{\sigma(x)} \quad (2)$$

where x_i is the value of the ith feature, μ(x) is the mean of the feature, and σ(x) is the standard deviation of the feature.

Removal of Incomplete Data: This step involves identifying and removing data points with missing values. The equation for this process can be represented as:

$$x_{new} = \{x_i \in x \,|\forall j: x_{\{ij\}} \neq NaN\} \quad (3)$$

where x_{new} is the new dataset without missing values, x_i is the ith data point in the original dataset, and $x_{\{ij\}}$ is the jth feature value of the ith data point. These preprocessing steps play a critical role in ensuring the quality and consistency of the data, ultimately improving the performance of the attack classification model.

3.2.4. Feature Selection

The feature selection module of this research employs Linear Regression–Recursive Feature Elimination (LR-RFE) to enhance the efficiency and interpretability of the RNN model used for attack detection in IIoT networks. Initialization involves the creation of a linear regression model, denoted as $(\overline{LR})$, utilizing the complete set of features $(\bar{X})$and the target variable $(\bar{Y})$. Subsequently, the feature ranking process assigns weights to each feature based on their contributions to the linear regression model, facilitating the identification of their importance. Through recursive feature elimination, the module iteratively removes the least important features, refitting the linear regression model at each step. The process continues until the desired subset of features is obtained. The final iteration yields the optimal feature subset. Algorithm 5 explains the feature selection process.

Algorithm 5 Feature Selection Using LR-RFE

1:**Input:** Feature matrix X, Target variable Y
2: Initialize linear regression model: $LR \leftarrow$ LinearRegression(X,Y)
3: **Initialize:** All features: $F \leftarrow$ AllFeatures(X)
4: **while** thestopping criterion is not met **do**
5: Train LR on X using features in F
6: Obtain feature weights: $W \leftarrow$ FeatureWeights(LR)
7: Rank features based on weights: Franked $\leftarrow$ RankFeatures(W)
8: Remove least important feature(s): $F \leftarrow F$ranked [1 num features to keep]
9: Evaluate performance metrics of LR
10: **end while**
11: **Output:** Optimal feature subset: F

3.2.5. RNN Model Development

The proposed RNN architecture for attack detection in IIoT systems comprises several key layers designed to efficiently capture and analyse the attacks in IIoT. At the input layer, pre-processed and feature-selected data representing various contextual parameters, such as computational cost, packet transmission times, and hop count, is fed into the network. Following the input layer, an embedding layer transforms the input features into a format suitable for the RNN, facilitating the capture of relationships and dependencies between different features. The core of the proposed architecture lies in the recurrent layer, which consists of recurrent units enabling the network to retain the memory of previous inputs. This layer is instrumental in capturing temporal patterns, essential for identifying potential attacks that may exhibit specific sequences or patterns over time. To enhance the robustness of the model and prevent overfitting, a dropout layer is introduced, randomly deactivating a fraction of neurons during training. A dense layer, maps the learned features from the recurrent layer to the output layer. This layer combines the temporal information captured by the recurrent layer to make predictions regarding the presence or absence of attacks. The output layer, typically utilizing sigmoid activation functions for multiclass classification, respectively, produces the final predictions.

In this research, a custom dataset was manually developed to train and evaluate the proposed attack detection model. The dataset comprises a total of 30,000 instances, categorized into three types of attacks and normal communications. The attacks included in the dataset are Distributed Denial of Service (DDoS), Man-in-the-Middle (MitM), and ransomware. To maintain a balanced distribution, each type of attack consists of 25 instances, summing up to 75 attack instances, and the remaining instances are labeled as normal communications. To create a balanced dataset, the distribution is as follows: 25 instances of DDoS attacks, 25 instances of MitM attacks, 25 instances of ransomware attacks, and the remaining instances labelled as normal communications.

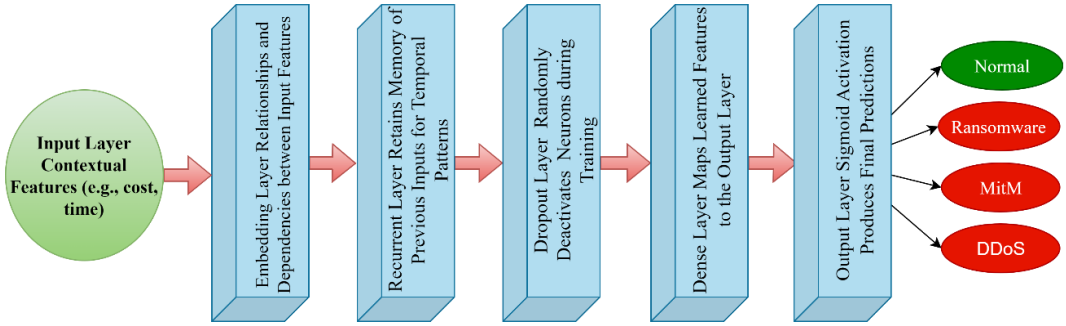

Figure 4. Layer details of proposed RNN.

For model training and evaluation, 80 randomly selected instances are used for training the proposed RNN, and the remaining instances are utilized for testing the model. The training process of the proposed RNN is explained in detail in Algorithm 6.

Algorithm 6 Recurrent Neural Network (RNN) for Attack Detection

1: **Input:** Preprocessed and feature-selected data: $X_{\text{preprocessed}}$, Target labels: Y
2: Initialize RNN model: $RNN \leftarrow$ CreateRNNModel()
3: **Split data:** Training set: $X_{\text{train}}, Y_{\text{train}}$, Testing set: $X_{\text{test}}, Y_{\text{test}}$
4: Train RNN on X_{train} with corresponding labels Y_{train}
5: **Classify attack types:** Predicted labels: $Y_{\text{pred}} \leftarrow$ RNNPredict(RNN, X_{test})
6: Evaluate performance metrics: Accuracy, Precision, Recall, F1-score, etc.

4. RESULTS AND DISCUSSION

This section evaluates the performance of the proposed IECC and RNN models in three phases. The first phase compares the computational efficiency of the proposed IECC to popular cryptographic algorithms. The second phase evaluates the computational efficiency of the proposed IECC across different data sizes. The third phase compares the attack prediction efficiency of the proposed RNN to existing AI-based prediction models.

4.1. System Setup

To evaluate the effectiveness of the proposed IECC-RNN, a comprehensive experimental setup was established. The implementation utilized Python 3.7 along with dedicated networking packages. The system infrastructure comprised a powerful configuration featuring 32 GB RAM, a 2.75 GHz processor, an Intel B85 series motherboard, 1 TB SATA storage, and an NVIDIA 730 8GB graphics card. To simulate real-world IIoT attack scenarios, an artificial network environment was constructed, as depicted in Figure 5. This network encompassed both normal and malware nodes, allowing for realistic simulations of intrusion attempts. Contextual features extracted from this IIoT network setup, including normal computational cost, sending/receiving packet time, hop count, and others, served as crucial indicators for identifying compromised nodes. Nodes exhibiting contextual feature values that deviated significantly from the norm were classified as potentially compromised nodes. Based on these contextual features, the research categorized IIoT nodes into normal and abnormal groups. To thoroughly assess the RNN's performance, 80% of the available data was allocated for training the model, while the remaining 120% was reserved for testing. Figure 5b illustrates the classification of normal and abnormal nodes, with normal nodes represented by blue circles and abnormal nodes represented by red circles.

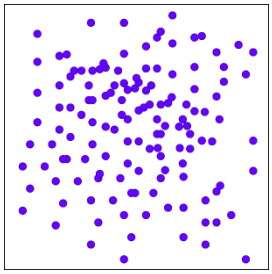

Figure 5. (a) Spatial distribution of nodes.

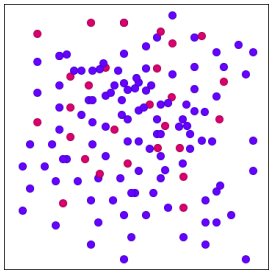

Figure 5. (b) Classification of normal and abnormal nodes based on contextual features.

4.2. Performance Analysis of Improved Elliptic Curve Cryptography (IECC)

In the comparative analysis of this research, the computational efficiency of the proposed Improved Elliptic Curve Cryptography (IECC) is assessed against existing popular cryptographic algorithms. The evaluation aims to provide insights into the performance and efficiency of IECC in comparison to well-established cryptographic techniques. The algorithms selected for this comparison include RSA (Rivest-Shamir-Adleman), AES (Advanced Encryption Standard), DSA (Digital Signature Algorithm), Diffie-Hellman, SHA-256 (Secure Hash Algorithm 256-bit), and ECDSA (Elliptic Curve Digital Signature Algorithm).The computational efficiency of the proposed IECC will be evaluated in terms of key generation speed, encryption and decryption time, throughput, memory usage, information loss, and overall processing time in comparison to these popular cryptographic algorithms. This analysis will contribute to understanding the strengths and potential advantages of IECC in IIoT security.

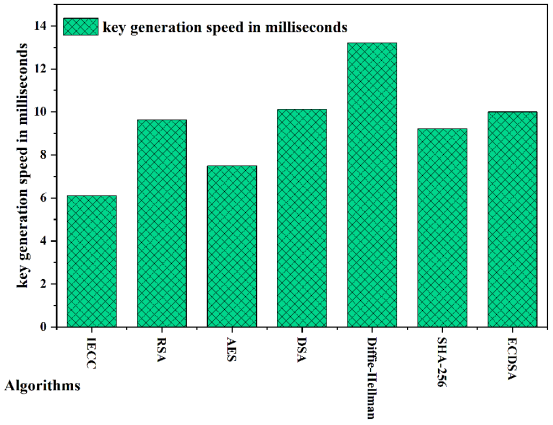

Figure 6.Key generation speed analysis of proposed IECC with existing encryption algorithms.

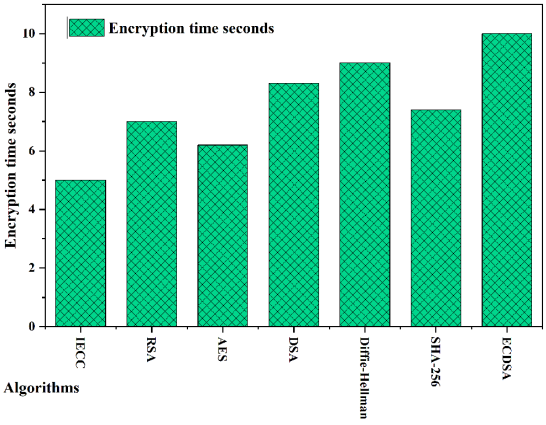

Figure 7. Encryption time analysis of proposed IECC with existing encryption algorithms.

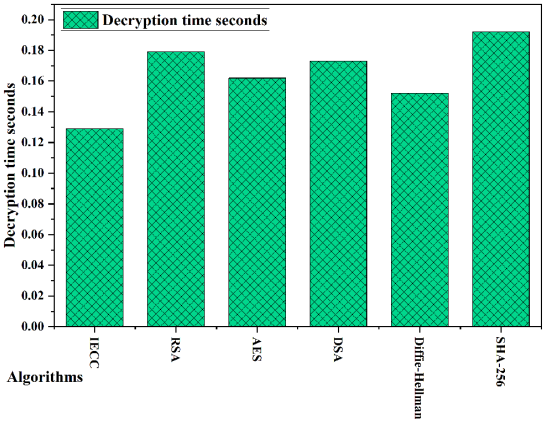

Figure 8. Decryption time analysis of proposed IECC with existing encryption algorithms

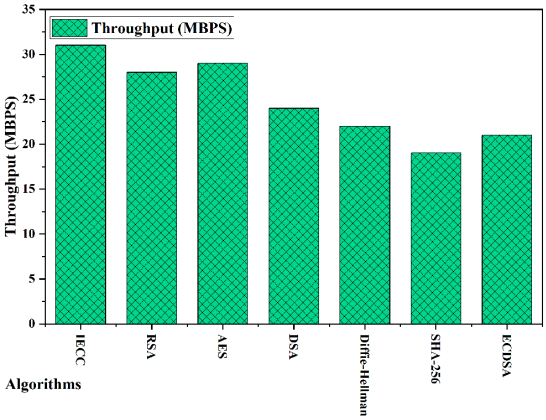

Figure 9. Throughput analysis of proposed IECC with existing encryption algorithms.

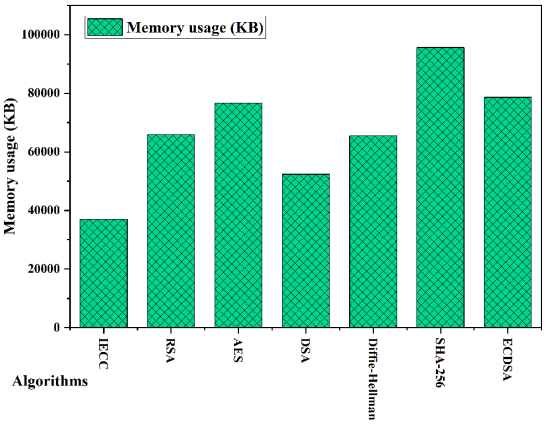

Figure 10. Memory usage analysis of proposed IECC with existing encryption algorithms.

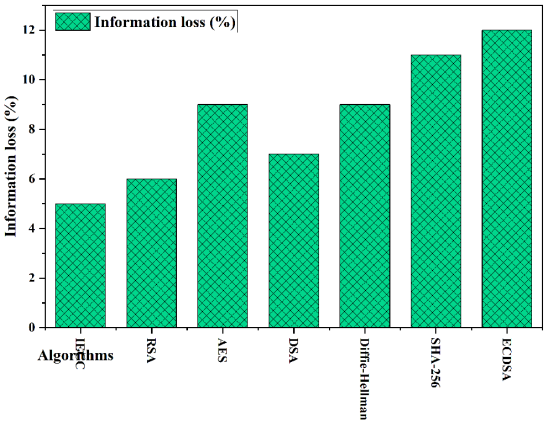

Figure 11. Information loss analysis of proposed IECC with existing encryption algorithms.

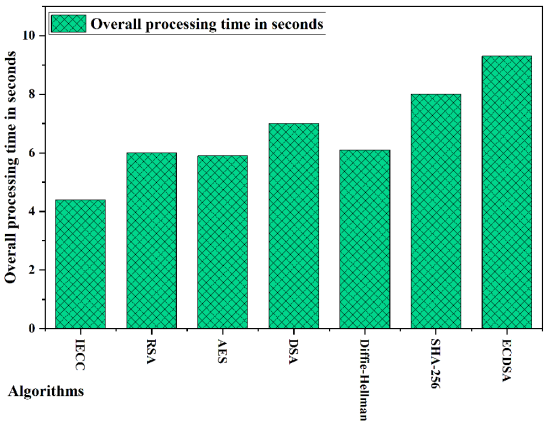

Figure 12. Overall processing time analysis of proposed IECC with existing encryption algorithms.

Figure 6 demonstrates the superior key generation speed of IECC, achieving a remarkable 6.1 milliseconds, outperforming RSA, DSA, and Diffie-Hellman by a significant margin. While AES and SHA-256 also exhibit competitive speeds, IECC stands out as the most efficient solution for key generation. Figure 7 illustrates IECC's strong performance in encryption time, with a swift processing time of 5 seconds. AES closely follows, showcasing its efficiency in securing data.

RSA and Diffie-Hellman, while still performing well, exhibit slightly longer encryption times. This emphasizes IECC's effectiveness in real-time encryption scenarios. Figure 8 highlights IECC's excellence in decryption time, achieving an impressively low processing time of 0.129 seconds. RSA and AES demonstrate competitive decryption speeds, while Diffie-Hellman and ECDSA also perform well. IECC's efficiency in decryption contributes to its overall appeal in resource-constrained environments. Figure 9 presents the throughput analysis, measured in Megabytes per second (MBPS), reflecting the efficiency of data processing. IECC attains a throughput of 31 MBPS, outpacing RSA and AES. This reinforces IECC's suitability for scenarios where data throughput is a crucial consideration, such as in industrial applications. Figure 10 showcases IECC's lightweight nature by consuming only 37,000 KB of memory. This is significantly lower than RSA and SHA-256, making IECC an attractive choice for memory-constrained environments. The memory efficiency of IECC is vital for IIoT devices with limited resources. Figure 11 illustrates IECC's effectiveness in preserving data integrity by maintaining a low information loss rate of 5%. This is a notable advantage over other algorithms, contributing to the reliability of IECC in secure communications. Figure 12 presents the overall processing time analysis, considering the combined impact of key generation, encryption, and decryption. IECC emerges as a well-balanced cryptographic solution with an overall processing time of 4.4 seconds. It outperforms competitors like RSA, DSA, and SHA-256, demonstrating its efficiency in executing end-to-end cryptographic operations. Overall, the proposed IECC consistently outperforms existing cryptographic algorithms in terms of key generation speed, encryption time, decryption time, throughput, memory usage, and information loss. Its overall processing time is also competitive, making it a well-rounded solution for resource-constrained environments like IIoT devices. The combination of efficiency and security makes IECC a promising candidate for various applications that demand secure data transmission and storage.

4.3. Comparative Performance Analysis of IECC and Existing Algorithms Across Different File Sizes

The proposed IECC algorithm has demonstrated superior performance in key generation speed, encryption time, decryption time, throughput, memory usage, and information loss compared to existing cryptographic algorithms. To further evaluate IECC's effectiveness, a comprehensive performance analysis was conducted across different file sizes, namely 3 MB, 6 MB, and 9 MB. This analysis aimed to assess the impact of file size on the performance of IECC and compare its efficiency against popular cryptographic algorithms such as RSA, DSA, Diffie-Hellman, AES, and SHA-256.

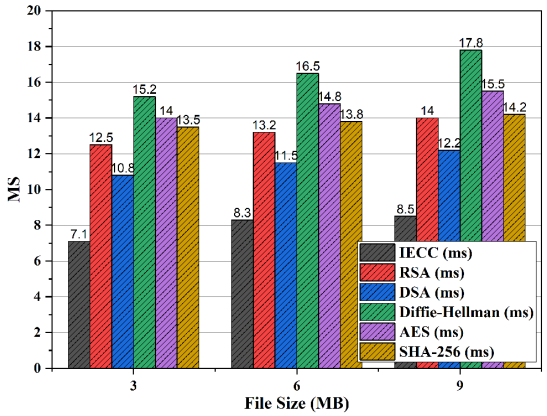

Figure 13. Key generation speed analysis of proposed IECC with existing encryption algorithms across different file sizes

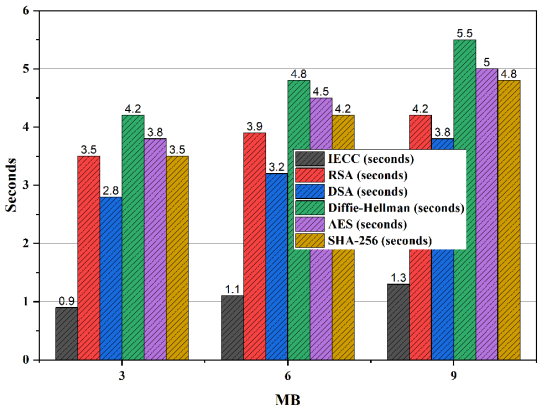

Figure14. Encryption speed analysis of proposed IECC with existing encryption algorithms across different file sizes

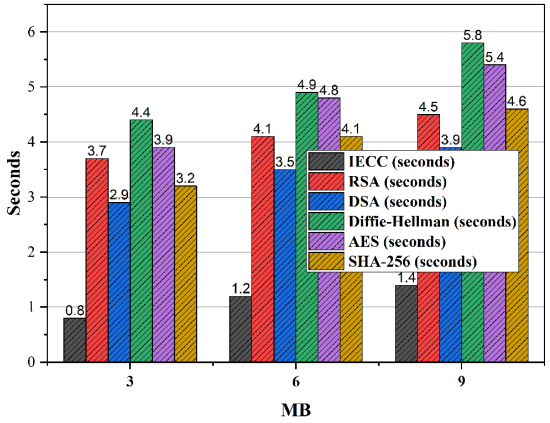

Figure 15. Decryption speed analysis of proposed IECC with existing encryption algorithms across different file sizes

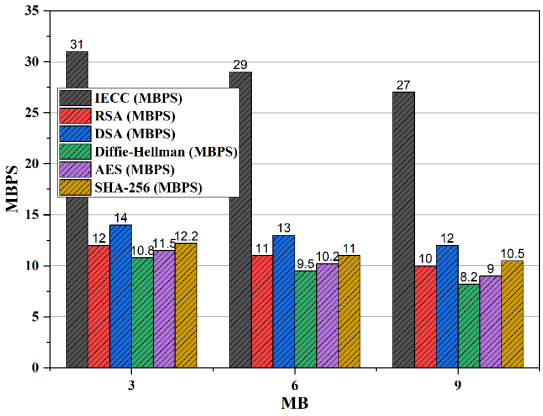

Figure 16. Throughput analysis of proposed IECC with existing encryption algorithms across different file sizes

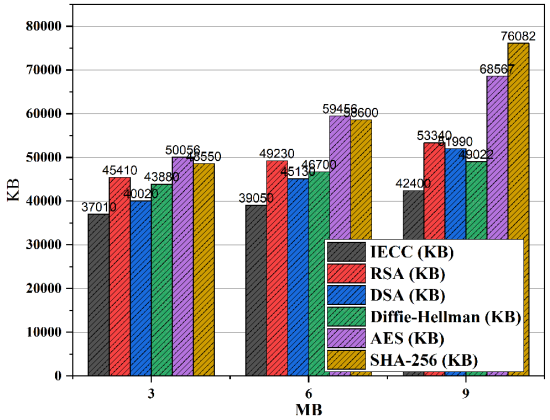

Figure 17. Memory usage analysis of proposed IECC with existing encryption algorithms across different file sizes

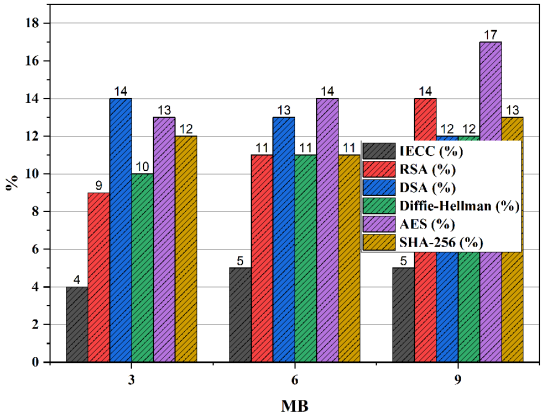

Figure 18. Information loss analysis of proposed IECC with existing encryption algorithms across different file sizes

The key generation speed analysis (Figure 13) highlights IECC's superior performance across various file sizes (3 MB, 6 MB, and 9 MB). IECC consistently outperforms RSA, DSA, Diffie-Hellman, AES, and SHA-256 in milliseconds, demonstrating its efficiency in generating cryptographic keys. For instance, at 3 MB, IECC achieves a key generation speed of 7.1 ms, while RSA, DSA, Diffie-Hellman, AES, and SHA-256 exhibit 12.5 ms, 10.8 ms, 15.2 ms, 14.0 ms, and 13.5 ms, respectively. Similar trends are observed at 6 MB and 9 MB, further strengthening IECC's efficiency. IECC's superior performance extends to encryption speed, as demonstrated in figure 14. Across different file sizes, IECC consistently achieves faster encryption times in seconds compared to RSA, DSA, Diffie-Hellman, AES, and SHA-256. At 3 MB, IECC encrypts data in 0.9 seconds, while RSA, DSA, Diffie-Hellman, AES, and SHA-256 require 3.5 seconds, 2.8 seconds, 4.2 seconds, 3.8 seconds, and 3.5 seconds, respectively. This efficiency is maintainedat 6 MB and 9 MB, further establishing IECC as a highly efficient encryption solution. The decryption speed analysis presented in figure 15 further strengthens IECC's efficiency in decrypting data. IECC consistently outperforms RSA, DSA, Diffie-Hellman, AES, and SHA-256 in seconds across different file sizes. This efficiency makes IECC suitable for real-time decryption for resource limited devices. Figure 16 illustrates IECC's superior throughput, measured in Megabytes per second (MBPS), across various file sizes. IECC consistently achieves higher throughput values compared to RSA, DSA, Diffie-Hellman, AES,

and SHA-256. This emphasizes IECC's suitability for scenarios where data throughput is crucial, such as in industrial applications. IECC exhibits efficient memory usage (figure 17), consuming significantly less memory (in kilobytes) compared to RSA, DSA, Diffie-Hellman, AES, and SHA-256. This lightweight nature makes IECC an attractive choice for memory-constrained environments like IoT devices. The information loss analysis (figure 18) demonstrates IECC's effectiveness in preserving data integrity with low information loss percentages. IECC consistently outperforms RSA, DSA, Diffie-Hellman, AES, and SHA-256, contributing to the reliability of IECC in secure communications.

4.4. Prediction Efficiency Analysis

This section presents a comprehensive evaluation of the prediction efficiency of the proposed RNN-based attack prediction model. To establish a benchmark, we compare the performance of the proposed model against a range of established AI-based attack prediction models, including deep neural networks (DNNs), deep belief networks (DBNs), radial basis function networks (RBFNs), and long short-term memory (LSTM) networks. Table 1 lists the stimulation parameters for theabove AI models. To objectively assess the prediction efficiency of these models, we employ a variety of performance metrics, including accuracy, specificity, precision, recall, false positive rate (FPR), and false negative rate (FNR). Accuracy measures the overall correctness of the model's predictions, while specificity and precision indicate the model's ability to correctly identify normal traffic and attack traffic, respectively. Recall measures the proportion of attack traffic correctly identified by the model, while FPR and FNR indicate the model's tendency to incorrectly classify normal as attack and vice versa, respectively. Which are calculated by using the following formulas.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (4)$$

$$Specificity = \frac{TN}{TN + FP} \quad (5) \quad Precision = \frac{TP}{TP + FP} \quad (6)$$

$$Recall = \frac{TP}{TP + FN} \quad (7)$$

$$FPR = \frac{FP}{FP + TN} \quad (8)$$

$$FNR = \frac{FN}{FN + TP} \quad (9)$$

where TP is True Positive, TN is True Negative, FP is False Positive, and FN is False Negative.

Table 1. Stimulation Parameters for AI Models.

Parameter	Proposed RNN
Primary Learning Rate	0.001
Loss Function	MSE
Optimizer	Adam
Dropout	0.2
State Activation Function	tanh
Learn Rate Drop Factor	0.5
Gradient Threshold	0.5
Gate Activation Function	Sigmoid
Number of Layers	3
Maximum Epochs	300
Batch Size	128

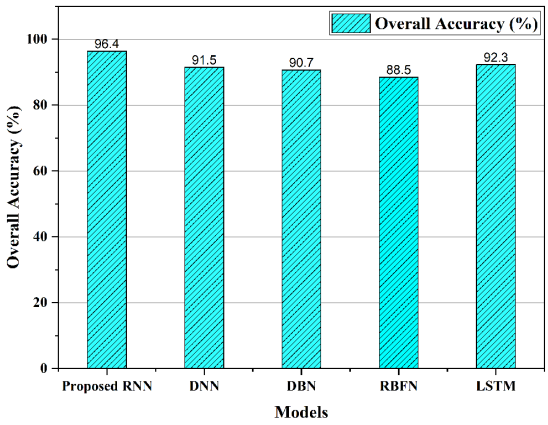

Figure 19.Overall accuracy comparison of proposed RNN with existing AI-based attack prediction models.

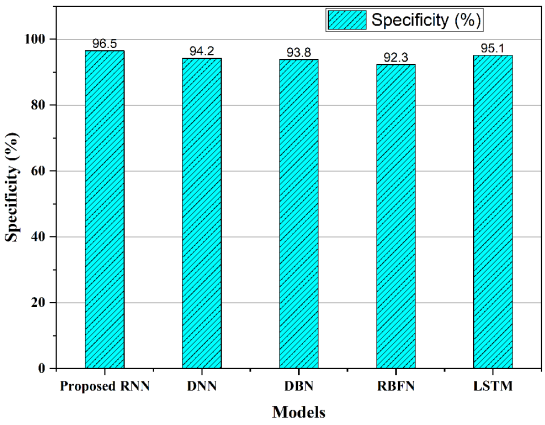

Figure 20.Specificity comparison of proposed RNN with existing AI-based attack prediction models.

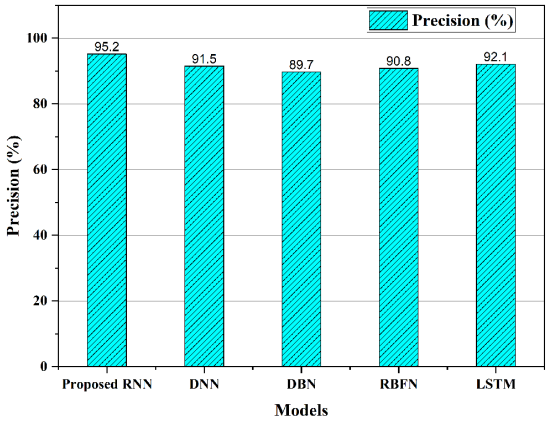

Figure 21.Precision comparison of proposed RNN with existing AI-based attack prediction models.

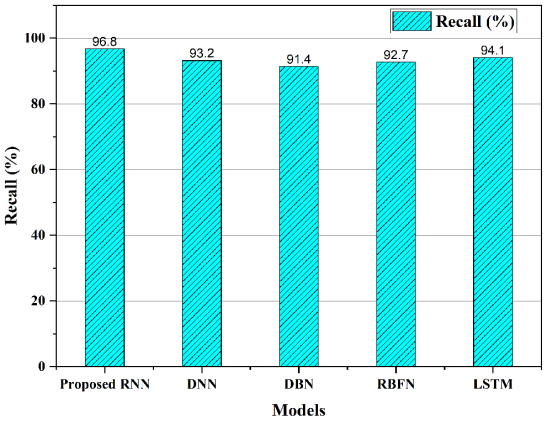

Figure 22.Recall the comparison of the proposed RNN with existing AI-based attack prediction models.

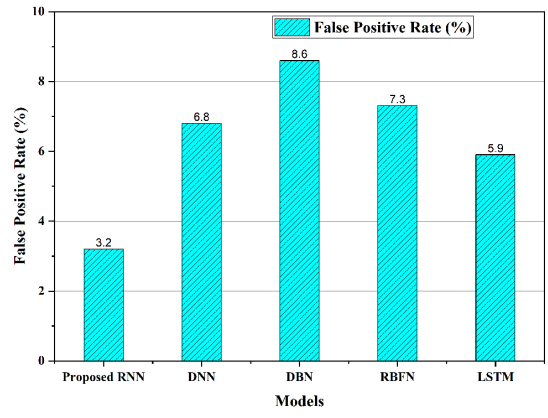

Figure23. FPR comparison of proposed RNN with existing AI-based attack prediction models.

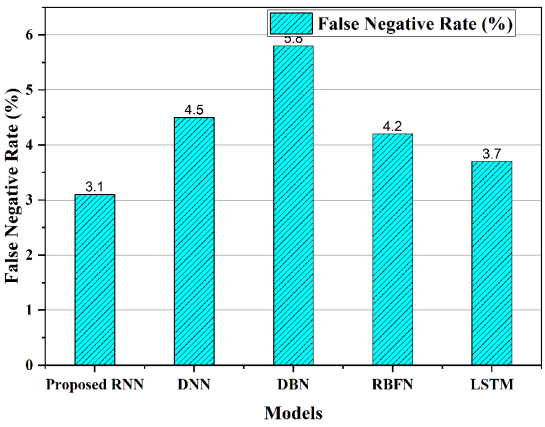

Figure24. FNR comparison of proposed RNN with existing AI-based attack prediction models.

Figure 19 illustrates the superior overall accuracy of the proposed RNN model compared to other AI-based attack prediction models. The proposed RNN stands out with the highest overall accuracy at 96.4%, surpassing DNN (91.5%), DBN (90.7%), RBFN (88.5%), and LSTM (92.3%). Figure 20 demonstrates the high specificity of the proposed RNN model. Specificity measures the model's ability to correctly identify normal traffic. The proposed RNN again leads with a specificity of 96.5%, while DNN (94.2%), DBN (93.8%), RBFN (92.3%), and LSTM (95.1%) follow closely. This suggests that the proposed RNN excels in distinguishing normal network behaviour, effectively reducing false positives. Figure 21 highlights the precision of the proposed RNN model. The proposed RNN exhibits superior precision with a value of 95.2%, outperforming DNN (91.5%), DBN (89.7%), RBFN (90.8%), and LSTM (92.1%). This indicates that the proposed RNN effectively distinguishes genuine attacks from normal traffic. Figure 22 demonstrates the high recall of the proposed RNN model. Recall measures the proportion of correctly identified attack traffic. The proposed RNN achieves the highest recall score at 96.8%, outperforming DNN (93.2%), DBN (91.4%), RBFN (92.7%), and LSTM (94.1%). This indicates that the RNN effectively captures instances of actual attacks, minimizing false negatives. Figure 23 showcases the low false positive rate of the proposed RNN model. The false positive rate (FPR) indicates the model's tendency to classify normal as anattack. The proposed RNN again demonstrates superior performance with a low FPR of 3.2%. This suggests that the RNN effectively avoids labelling normal as attacks, reducing false alarms. Figure 24 indicates the low false negative rate of the proposed RNN model. Similarly, in terms of false negative rate (FNR), representing the model's tendency to incorrectly classify attacks as normal, the proposed RNN exhibits a minimal FNR of 3.1%. This indicates that the RNN effectively identifies genuine attacks, minimizing missed detections. The Receiver Operating Characteristic (ROC) chart for the research project demonstrates a high level of performance, achieving an impressive area under the curve (AUC) value of 0.988. This indicates the strong ability of the proposed RNN-based attack detection model to distinguish between true positive and false positive rates effectively. In Figure 25, the ROC chart visually represents the trade-off between the true positive rate (sensitivity) and the false positive rate.

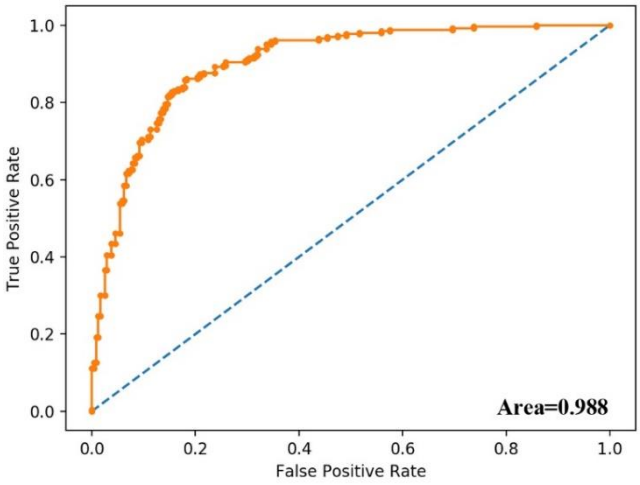

Figure25. ROC chart of proposed RNN.

4.5. Discussions

The results of the computational efficiency analysis demonstrate the superiority of the Improved Elliptic Curve Cryptography (IECC) over established cryptographic algorithms. IECC outperforms in key areas such as key generation speed, encryption and decryption time, throughput, memory usage, information loss, and overall processing time. This suggests that

IECC is well-suited for resource-constrained IIoT devices, addressing challenges related to computational complexity and key management. The lightweight nature of IECC, as indicated by its minimal memory usage, positions it as an efficient cryptographic solution for IIoT environments.
The evaluation of IECC across different file sizes strengthens its consistent efficiency in key generation, encryption, decryption, throughput, memory usage, and information loss. This robust performance across varied data sizes underscores the adaptability of IECC, making it a reliable choice for securing data transmission in IIoT networks, regardless of the file size.

The RNN exhibits outstanding performance in accurately identifying and classifying various types of attacks, including Distributed Denial of Service (DDoS), Man-in-the-Middle (MitM), and ransomware attacks. The high precision, recall, and overall accuracy of the RNN contribute to its effectiveness in real-time attack detection. The low false positive and false negative rates further enhance the reliability of the RNN, minimizing both false alarms and missed detections.

The integration of IECC and RNN in the proposed two-layered architecture emerges as a comprehensive solution for IIoT security. IECC ensures secure data transmission by addressing cryptographic challenges in resource-constrained devices, while the RNN enhances the network's resilience through accurate and timely attack detection. The combination of these two layers addresses the inherent trade-off between cryptographic strength and resource constraints, providing a balanced approach to IIoT security. The research significantly contributes to the field of industrial cybersecurity by presenting an integrated solution. The two-layered architecture not only improves cryptographic protection but also introduces an effective attack detection mechanism. This integrated approach aligns with the evolving landscape of cyber threats and the need for adaptive security measures in IIoT networks.

Despite the benefits of the proposed two-layered architecture, several drawbacks must be considered. First, while the method introduces minimal overhead, the cryptographic demands of the IECC implementation can still impose a significant computational burden, particularly for resource-constrained IIoT devices. Second, the system requires specialized expertise to deploy and manage encryption and RNN-based detection mechanisms effectively, leading to increased operational costs and added complexity. Third, the performance of the RNN is highly reliant on the quality of the training dataset; inadequate and biased data can severely compromise detection accuracy.

5. CONCLUSION

This research introduces a two-layered security architecture designed to enhance the security and integrity of IIoT networks. The first layer incorporates the IECC, addressing challenges related to computational complexity and key management in resource-constrained IIoT devices. IECC's lightweight nature and superior efficiency in key generation, encryption, and decryption processes position it as a promising solution for secure data transmission in industrial environments. The second layer of the architecture employs an RNN-based attack detection system, showcasing high accuracy, specificity, precision, and recall in identifying and classifying potential network attacks. The RNN's ability to minimize false positives and false negatives enhances its reliability in real-time attack detection scenarios. The integration of IECC and RNN in this two-layered approach offers a balanced solution, effectively addressing the inherent trade-off between cryptographic strength and the resource constraints of industrial devices. The performance analysis of IECC reveals several key findings. IECC consistently outperforms popular cryptographic algorithms in terms of key generation speed, encryption and decryption times, throughput, memory usage, information loss, and overall processing time. Notably, IECC showcases remarkable speed in key generation (6.1 milliseconds), swift encryption time (5

seconds), and impressively low decryption time (0.129 seconds). Its efficiency in resource-constrained environments is evident in its low memory usage (37,000 KB) and low information loss rate (5%). IECC emerges as a well-balanced cryptographic solution with an overall processing time of 4.4 seconds, outperforming its competitors across various file sizes (3 MB, 6 MB, and 9 MB). The evaluation of the RNN-based attack prediction model further strengthens the research findings. The RNN exhibits superior overall accuracy (96.4%), specificity (96.5%), precision (95.2%), and recall (96.8%) when compared to existing AI-based models. Its low false positive rate (FPR) of 3.2% and minimal false negative rate (FNR) of 3.1% underscore its effectiveness in identifying and classifying potential network attacks.

CONFLICTS OF INTEREST

The authors declare no conflict of interest.

REFERENCES

[1] M. Javaid, A. Haleem, R. Pratap Singh, S. Rab, and R. Suman, "Upgrading the manufacturing sector via applications of Industrial Internet of Things (IIoT)," Sensors International, vol. 2, p. 100129, 2021.

[2] W. Z. Khan, M. H. Rehman, H. M. Zangoti, M. K. Afzal, N. Armi, and K. Salah, "Industrial internet of things: Recent advances, enabling technologies and open challenges," *Computers & Electrical Engineering*, vol. 81, p. 106522, Jan. 2020.

[3] N. N. Hurrah, S. A. Parah, J. A. Sheikh, F. Al-Turjman, and K. Muhammad, "Secure data transmission framework for confidentiality in IoTs," Ad Hoc Networks, vol. 95, p. 101989, Dec. 2019.

[4] A. Mosteiro-Sanchez, M. Barcelo, J. Astorga, and A. Urbieta, "Securing IIoT using Defence-in-Depth: Towards an End-to-End secure Industry 4.0," Journal of Manufacturing Systems, vol. 57, pp. 367–378, Oct. 2020.

[5] L. Tawalbeh, F. Muheidat, M. Tawalbeh, and M. Quwaider, "IoT Privacy and Security: Challenges and Solutions," *Applied Sciences*, vol. 10, no. 12, p. 4102, Jun. 2020.

[6] A. Lohachab and Karambir, "ECC based inter-device authentication and authorization scheme using MQTT for IoT networks," Journal of Information Security and Applications, vol. 46, pp. 1–12, Jun. 2019.

[7] Y. Yang, S.-H. Lee, J. Wang, C.-S. Yang, Y.-M. Huang, and T.-W. Hou, "Lightweight Authentication Mechanism for Industrial IoT Environment Combining Elliptic Curve Cryptography and Trusted Token," *Sensors*, vol. 23, no. 10, pp. 4970–4970, May 2023.

[8] S. Ullah and R. Zahilah, "Curve25519 based lightweight end-to-end encryption in resource constrained autonomous 8-bit IoT devices," Cybersecurity, vol. 4, no. 1, Mar. 2021.

[9] K. Tsiknas, D. Taketzis, K. Demertzis, and C. Skianis, "Cyber Threats to Industrial IoT: A Survey on Attacks and Countermeasures," *IoT*, vol. 2, no. 1, pp. 163–186, Mar. 2021.

[10] P. Williams, I. K. Dutta, H. Daoud, and M. Bayoumi, "A Survey on Security in Internet of Things with a Focus on the Impact of Emerging Technologies," *Internet of Things*, vol. 19, p. 100564, Jul. 2022.

[11] A. Awajan, "A Novel Deep Learning-Based Intrusion Detection System for IoT Networks," Computers, vol. 12, no. 2, p. 34, Feb. 2023.

[12] H. C. Altunay and Z. Albayrak, "A hybrid CNN+LSTM-based intrusion detection system for industrial IoT networks," Engineering Science and Technology, an International Journal, vol. 38, p. 101322, Feb. 2023.

[13] J. B. Awotunde, C. Chakraborty, and A. E. Adeniyi, "Intrusion Detection in Industrial Internet of Things Network-Based on Deep Learning Model with Rule-Based Feature Selection," Wireless Communications and Mobile Computing, vol. 2021, p. e7154587, Sep. 2021.

[14] R. Malik, Y. Singh, Z. A. Sheikh, P. Anand, P. K. Singh, and T. C. Workneh, "An Improved Deep Belief Network IDS on IoT-Based Network for Traffic Systems," Journal of Advanced Transportation, vol. 2022, p. e7892130, Apr. 2022.

[15] H. Kim and K. Lee, "IIoT Malware Detection Using Edge Computing and Deep Learning for Cybersecurity in Smart Factories," *Applied Sciences*, vol. 12, no. 15, p. 7679, Jul. 2022.

[16] A. Yazdinejad, B. Zolfaghari, A. Dehghantanha, H. Karimipour, G. Srivastava, and R. M. Parizi, "Accurate threat hunting in industrial internet of things edge devices," *Digital Communications and Networks*, Sep. 2022.
[17] Almaiah, M.A.; Ali, A.; Hajjej, F.; Pasha, M.F.; Alohali, M.A. A Lightweight Hybrid Deep Learning Privacy Preserving Model for FC-Based Industrial Internet of Medical Things. Sensors 2022, 22, 2112.
[18] A. Al-Abassi, H. Karimipour, A. Dehghantanha and R. M. Parizi, "An Ensemble Deep Learning-Based Cyber-Attack Detection in Industrial Control System," in IEEE Access, vol. 8, pp. 83965-83973, 2020, doi: 10.1109/ACCESS.2020.2992249.
[19] H. AlMajed and A. AlMogren, "A Secure and Efficient ECC-Based Scheme for Edge Computing and Internet of Things," Sensors, vol. 20, no. 21, p. 6158, Oct. 2020.
[20] B. Hammi, A. Fayad, R. Khatoun, S. Zeadally and Y. Begriche, "A Lightweight ECC-Based Authentication Scheme for Internet of Things (IoT)," in IEEE Systems Journal, vol. 14, no. 3, pp. 3440-3450, Sept. 2020.
[21] U. Gulen and S. Baktir, "Elliptic Curve Cryptography for Wireless Sensor Networks Using the Number Theoretic Transform," Sensors, vol. 20, no. 5, p. 1507, Mar. 2020.
[22] A. Subashini and P. Kanaka Raju, "Hybrid AES model with elliptic curve and ID based key generation for IOT in telemedicine," Measurement: Sensors, vol. 28, p. 100824, Aug. 2023.
[23] M. A. Shaaban, A. S. Alsharkawy, M. T. AbouKreisha, and M. A. Razek, "Efficient ECC-Based Authentication Scheme for Fog-Based IoT Environment," International journal of Computer Networks & Communications, vol. 15, no. 04, pp. 55–71, Jul. 2023.
[24] S. Bose, T. Arjariya, A. Goswami, and S. Chowdhury, "Multi-Layer Digital Validation of Candidate Service Appointment with Digital Signature and Bio-Metric Authentication Approach," International journal of Computer Networks & Communications, vol. 14, no. 5, pp. 81–100, Sep. 2022.