ORIGINAL RESEARCH OPEN ACCESS

CIDER: Cyber-Security in Industrial IoT Using Deep Learning and Ring Learning with Errors

Siu Ting Tsoi | Anish Jindal

Department of Computer Science, Durham University, Durham, UK

Correspondence: Anish Jindal (anishjindal90@gmail.com, anish.jindal@durham.ac.uk)

Received: 2 June 2024 | **Revised:** 5 November 2024 | **Accepted:** 31 March 2025

Handling Editor: Subham Sahoo

Funding: This work was supported by the Durham University start-up grant.

Keywords: cryptography | cyber-physical systems | Internet of Things

ABSTRACT

Traditional security measures such as access control and authentication need to be more effective against ever-evolving threats. Moreover, security concerns increase as more industries shift towards adopting the industrial Internet of things (IIoT). Therefore, this paper proposes secure measures using deep machine learning-based intrusion detection and advanced encryption schemes based on lattice-based cryptography on three-layered cloud-edge-fog IIoT architecture. The novelty of the paper is an integrated security framework for IIoT that combines deep learning-based intrusion detection system (IDS) with lightweight cryptographic protocols. For deep learning, multi-layer perception (MLP), convolutional neural network (CNN), and TabNet were implemented for intruder detection systems from edge to cloud layer, and ring learning with error (RLWE) was proposed for homomorphic encryption to communicate data between fog and edge layer. The evaluation experiments were performed on the Ton_IoT dataset and the results show that the deep learning models have a very good accuracy of around 92% for multiclass attack classification. Moreover, RLWE results show improved encryption time and reduced ciphertext size against standard Learning With Error encryption.

1 | Introduction

The industrial Internet of things (IIoT) provides efficient and reliable operation of smart devices in real-time, reducing the need for human interactions. There are now more than 50 billion IoT devices connected to the Internet [1]; as such, more companies are shifting towards the use of the IIoT industry, allowing the industries to drive productivity with a reliable IoT infrastructure. IIoT is widely used in many sectors requiring heavy machinery processing, including medicine, factories, energy management, and smart cities. Although the IIoT can provide reliable communication between devices, it faces several security challenges. Most noticeably, with the increased number of malicious attacks, hackers can access the whole industrial infrastructure through network attacks. Not only can sensitive information be leaked, but this could also lead to system failures. Such attacks can be classified into different types: device attacks, application service attacks, network attacks, web interface attacks, and data integrity attacks [2]. Cyberattacks can target the electrical grid in the winter, shutting down power, or tuning down cooling towers in nuclear fusion reactors; cyberterrorism is said to have caused trillions of losses in countries' economies [3]. With the growth of AI-generated attacks [4], more than existing measurements may be needed.

As the roles and functionalities of IIoT sensors are independent, there's no one-way solution for security problems. Each layer of the architecture needs to be evaluated for any weaknesses it may contain and provide a remedial security solution. The IIoT architecture can be divided into multiple layers, including the basic three-layer, derived four-layer, or the detailed five-layer architecture [5]. Each layer has its responsibilities and is crucial in maintaining the whole IIoT infrastructure; the following shows the functionalities of each layer:

- Sensing layer: Includes sensors and devices that collect information such as objects and data in the surroundings; it then transmits the information to the network.
- Network layer: This layer transmits data from the sensor layer to the Internet using gateways [6]. It comprises different wireless sensor network (WSN) technologies, such as 3G/4G, WIFI, Bluetooth etc. It acts as a middleman, collecting, filtering, and transporting information to other layers.
- Middleware layer: This layer enables components of different technologies to work together; exchanged data is managed and operated here.
- Application layer: This is located in the cloud, where data is presented, and functionalities are provided; long-term data is stored here.
- Business layer: It is responsible for ensuring that all system protocols and logic are followed to meet the goals, including producing secure devices against attacks [7].

IIoT faces multiple issues with implementing security measurements, most noticeably the lack of computation power and low bandwidth in IoT devices [8]. This means measures that require high computational power may not be implemented; new methodologies have to meet the requirements of the devices [9], causing a bottleneck in developing secure devices. The lack of organisational support further enlarges this issue, as corporations favor implementing better technological functions in IoT devices rather than security measurements to please investors and users [10].

1.1 | Motivation

With advancements in artificial intelligence (AI), more AI-driven attacks are happening to break down the underlying system which makes it challenging to detect such types of attacks. Attackers often have dedicated infrastructure to carry out such attacks, while IIoT devices are computationally weak, thereby making them the soft target for attackers. While many studies have used machine learning (ML) and emphasise on the accuracy of models in identifying the correct attack/type when constructing an intruder detection system (IDS), the most critical factor for an IDS is its ability to promptly detect an attack. This is even more important in IIoT environment where identifying the attack promptly is crucial to stop it from hampering the whole industrial process. Apart from accuracy, the current security measures and protocols which may be enough against common attack surfaces, may not be as useful for the IIoT applications. As the operations and uses of IIoT increase, these strategies need to be as efficient as possible, allowing IIoT computational power to focus on more complex problems, such as detecting major system failures. Existing security solutions which are computationally expensive put burden on decision-making in IIoT applications which rely on the timeliness. Hence, the IIoT security protocols must be lightweight, making them feasible to implement in practice which is crucial for smooth industrial operations. Therefore, in this work, we focus on addressing these challenges by developing an integrated security framework combining deep learning-based IDS and lightweight cryptographic protocols specifically for IIoT applications.

1.2 | Contributions

Based on the challenges highlighted above, this paper proposes lightweight security measures to provide a secure environment for IIoT in general; the major contributions of the work are as follows:

- A secure key exchange authentication system when transferring readings from IIoT devices for data analysis.
- A lattice-based post-quantum homomorphic encryption system using Ring Learning With Error for data transmission.
- A deep learning-based intruder-detection system to validate the network packets transferring from the edge layer to the cloud layer.

1.3 | Organisation

The remaining paper is organised as follows: Section 2 discusses the related works; Section 3 illustrates the IIoT system model; Section 4 shows the proposed methodologies; Section 5 presents the experimentation details and evaluation of the proposed schemes; Section 6 concludes the work and suggests future contribution that can be added to the field.

2 | Related Works

Encryption has been heavily used when sending data across different layers in the IIoT architecture. Many IoT devices use symmetric encryption where the same key is used for encrypting and decrypting data [11]; this gives some benefits as small symmetric keys can achieve a high level of security and are easily manageable, requiring only one key. However, this creates risks for the IoT devices. Firstly, sharing a key is complicated as a "man-in-the-middle" attack can intercept it; furthermore, when a device is compromised, the encryption key can be obtained and used on other devices, thereby endangering the whole system. RSA is widely utilised in various encryption domains due to its efficiency and security; one example was shown on the smart manufacturing system [12]. It mainly facilitates key exchange and data integrity verification rather than direct data encryption. Conversely, LWE can be a better alternative as it can perform homomorphic encryption. Encrypted data can perform arithmetic operations such as addition without compromising any confidentiality about the plaintext. This attribute is advantageous for the architecture of IIoT, as it can preserve privacy during data processing.

Another type of encryption is asymmetric key encryption, divided into a pair: a private key for encryption and a public key for decryption. While this is less manageable than a symmetric key, having two keys allows for more flexible data encryption, as anyone can encrypt data using the public key. Elliptic-curve cryptography (ECC) [13] is an example of public key cryptography, which utilises the algebraic structure of elliptic curves over a finite field and is recognised for computational efficiency. Another example is learning with errors (LWE) cryptography, based on the idea of hiding secret information with the use of small errors; ring learning with error (RLWE) is a variant of LWE leveraging polynomial rings instead [14], adopting the properties of RLWE can provide better performance efficiency. LWE is acknowledged as post-quantum encryption, resistant against quantum computer attacks, which may be expected in the coming years. The combination of these schemes has also been utilised in the literature to improve the overall security of the system. For instance, in the investigation of E-healthcare authentication and data integrity schemes [15], RLWE was proposed for key exchange, and ECC was proposed for checking data integrity. While using post-quantum cryptography will be beneficial when quantum attacks are common in the future, it is relatively slower than other public key encryptions. This paper proposes using ECC for key exchange (as elliptic curve Diffie-Hellman) as it offers significantly less computation time than different schemes. A critical issue in elliptic curve cryptography is that choosing a correct elliptic curve is vital as known algorithms break "weak" curves [16]; however, it was not mentioned in the paper above.

Apart from these, ML and AI have been used widely in different aspects of information security as it can find patterns of data where it is not easily identifiable by humans. AI has recently become an enormous topic due to the rise of large language models (LLMs) such as ChatGPT. Firewalls block unauthorised access through the internet; however, using AI generated attacks, attackers can easily bypass firewalls. Hence, ML, and by extension deep learning, is an excellent tool for attack recognition, as it identifies any anomaly in the network packet, such as large packet size or high connection duration, which are usually difficult to spot by human analyst or a firewall logic. Standard deep learning models are multi-layer perception (MLP) that use linear layers and convolutional neural networks (CNN) which use convolutional layers to extract and understand the input features. TabNet [17] is another state-of-the-art model, using sequential attention layers to choose which features to focus on the input for tabular data, and has shown outstanding results compared to other neural networks. Implementing state-of-the-art deep learning algorithms and advanced encryption methods concurrently can significantly elevate the level of security while addressing the challenges in IIoT.

Research on Densely-ResNet [18] performed on the network dataset ToN_IoT had shown most of the attacks have a 100% detection rate (DR, or recall for intruder detection) with no false alarm rate (FAR), 99.95% DR for DDoS attacks without any FAR and 99.95% DR for XSS attacks with 0.03% FAR. Finally, 100% DR on Ransomware attacks but with 0.06% DR. Another research paper focused on Inception Time [19], and the results were similar when performed on the dataset Edge_IoT, with most types of attacks with close to 100% accuracy. The two lowest were password and cross-site scripting (XSS) attacks, with 59% and 63% accuracy, respectively. Other datasets that the researchers experimented with have some limitations. The CICIDS2017 dataset [21] doesn't contain any ground truth in the network packets; any attacks are unclear. Furthermore, new IoT attacks were not included in the dataset and are considered outdated [20]. The UNSW-NB15 dataset [22] only contains generic attacks against the edge layer without specific attacks on IIoT, such as XSS and password attacks. Table 1 shows the summary of the experimental works.

3 | System Model

This proposed scheme provides general security methodologies for the three-layered IIoT architecture, which contains the three layers: cloud, edge, and fog. The system model is presented in Figure 1. The structure is kept to a minimum so that it can be adapted to numerous industrial contexts while maintaining security standards.

The detailed description of the layers is as follows:

1. *Cloud Layer:* Provide end user interface and stores the data for long-term analysis.
2. *Edge Layer:* An intermediate layer for processing data collected from the fog layers as well as pass the required data to the cloud layer.
3. *Fog Layer:* Stationed next to IoT devices and is responsible for collecting and transmitting data.

The model in Figure 1 consists of three main security methodologies: Key exchange authentication and homomorphic encryption between fog and edge layers, as well as deep learning-based intrusion detection systems between edge and cloud layers. The steps involving the entire process of the system model are described as follows:

Step 1: The fog layer requests to send information to the edge layer.

Step 2: The edge layer initiates the authentication process. Key exchange is performed between the edge and fog layer.

Step 3: A secure channel is established When the authentication process is successful.

Step 4: Once the channel is secured, both layers can safely communicate. The fog layer will use the public key sent from the edge layer to encrypt the data, which is sent back to the edge layer for processing.

Step 5: Data will be sent to the cloud layer as network packets after decryption and processing.

Step 6: Finally, the IDS will detect if the network packets behave normally in the cloud layer; if everything works correctly, the data will be stored in the cloud server or for further long-term processing.

TABLE 1 | Survey summary for experimental works.

Reference	Year	Methodology	Dataset/Environment	Results
[12]	2020	Fog assisted task allocation (FATA), QCI-NN, RSA variant, Softmax-DNN	Smart manufacturing	Encryption and decryption time less than a quarter than FATA with 2048 key bits, and Softmax-DNN achieved 92.45% sensitivity
[15]	2018	Lattice and ECC-based authentication, integrity verification scheme	E-healthcare services	Latency for RSA is 10-folds more expensive than ECC
[17]	2021	TabNet	Forest cover type, poker hand induction, sarcos, Higgs boson, rossmann store sales	Outperforms other works on tabular datasets with the highest of 99.2% test accuracy on poker hand induction dataset
[18]	2020	Densely-ResNet	ToN-IoT, UNSW-NB15	99.93% accuracy on ToN-IoT dataset and 73.93% accuracy on UNSW-NB15 benchmark
[19]	2022	DenseNet, inception time	ToN-IoT, UNW-NB15, Edge-IIoT	Highest with 99.9% accuracy with DenseNet and 100% accuracy with inception time
[20]	2021	GBM, RF, NB, DNN	ToN-IoT	Training accuracy of RF with 99.98% and DNN with 99.92%

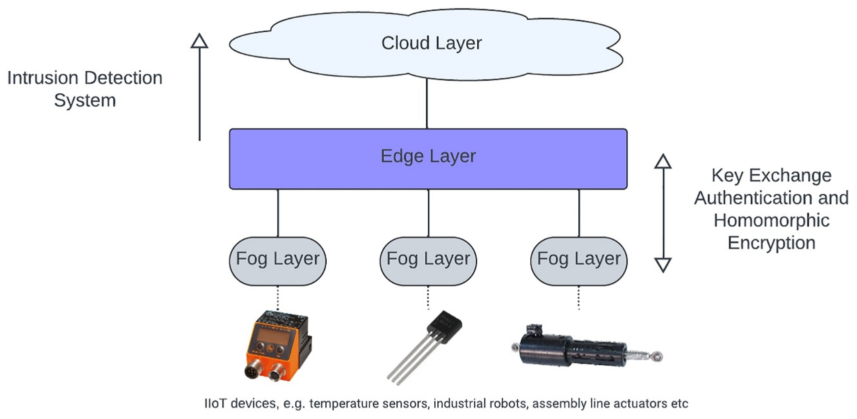
Cloud Layer
Intrusion Detection System
Edge Layer
Key Exchange Authentication and Homomorphic Encryption
Fog Layer
Fog Layer
Fog Layer
IIoT devices, e.g. temperature sensors, industrial robots, assembly line actuators etc

FIGURE 1 | Proposed system model for general three-layer architecture IIoT.

The detailed procedures for key exchange authentication, homomorphic encryption, and IDS are discussed in the next section.

4 | Proposed Methodology

The proposed scheme's operation is divided into three phases. The first phase is key exchange authentication to secure the transmission channel; the second phase is homomorphic encryption for data transmission; and the final phase is an IDS using deep learning models.

4.1 | Secure Authentication Key Exchange

The secure authentication key exchange (SAKE) is responsible for establishing a secure channel for data transmission. The entire process of the SAKE mechanism is shown in Figure 2 between the edge and fog layers. Initially, both layers communicate in a non-trusted channel; neither channel will transmit any data to each other until the channel is secured. The elliptic curve Diffie–Hellman (ECDH) is a lightweight scheme for this SAKE principle. ECDH combines two public key cryptography: Elliptic curve cryptography and Diffie–Hellman key exchange. This scheme is a variant of the Diffie–Hellman key exchange protocol; instead of operating on traditional numeric fields, it leverages the algebraic structure of elliptic curves to operate on elliptic curve fields while providing the same functionalities of key exchange. The Diffie–Hellman protocol leverages the hardness of the discrete log problem, and operating on an elliptic curve is believed to be harder to solve, as group operations such as point additions and multiplications involve algebraic formulas that are more complex than simple modular arithmetic presented in the classic DH. This property makes ECDH a more secure protocol, as no effective algorithms can solve the discrete log problem on an elliptic curve other than brute force, unlike the classic DL.

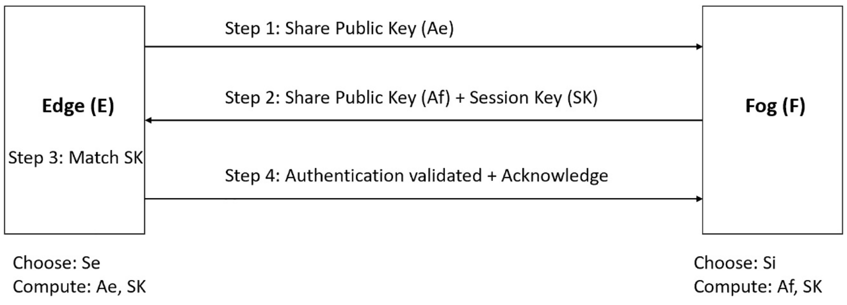
Edge (E)
Fog (F)
Step 1: Share Public Key (Ae)
Step 2: Share Public Key (Af) + Session Key (SK)
Step 3: Match SK
Step 4: Authentication validated + Acknowledge
Choose: Se
Compute: Ae, SK
Choose: Si
Compute: Af, SK

FIGURE 2 | Key exchange authentication process.

The elliptic curve is defined over a $\boldsymbol{K}$, where $\boldsymbol{K}$ is a finite field of complex or real numbers. It consists of a collection of points $(\mathbf{x}, \mathbf{y})$ over the Cartesian product of $\boldsymbol{K}$, $\boldsymbol{K}^2$ with the equation $\mathbf{y}^2 = \mathbf{x}^3 + \mathbf{ax} + \mathbf{b}$ for some $\mathbf{a}, \mathbf{b} \in \boldsymbol{K}$ such that $\mathbf{4a}^3 + \mathbf{27b}^2 \neq \mathbf{0}$ to ensure there are no singular points on the curve. A special point inf is the group operation's identity element. The addition of two points $\boldsymbol{P} = (\mathbf{x_1}, \mathbf{y_1})$ and $\boldsymbol{Q} = (\mathbf{x_2}, \mathbf{y_2})$, where $\boldsymbol{P} \neq \pm \boldsymbol{Q}$, then $\boldsymbol{P} + \boldsymbol{Q} = (\mathbf{x_3}, \mathbf{y_3})$ is defined as follows:

$$x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2 \tag{1}$$

$$y_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)(x_1 - x_3) - y_1 \tag{2}$$

Doubling a point in an elliptic curve can be performed in one operation. For any $\boldsymbol{P} = (\mathbf{x_1}, \mathbf{y_1})$ and $\boldsymbol{P} \neq -\boldsymbol{P}$, then $\mathbf{2}\boldsymbol{P} = (\mathbf{x_3}, \mathbf{y_3})$, where

$$x_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)^2 - 2x_1 \tag{3}$$

$$y_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)(x_1 - x_3) - y_1 \tag{4}$$

The double and add method is used in elliptic curves, which can effectively perform scalar multiplication by performing operations on the binary representation of the scalar number, shown in algorithm 1. The point-doubling operation of the elliptic curve makes this process efficient.

ALGORITHM 1 | Apply Double-and-Add Method for Elliptic Curves.

```
Input: $G$: Base point on the elliptic curve, $k$: Scalar multiplier,
$p$: Prime number defining the finite field
Output: Returns $kG$: The result of scalar multiplication of $G$ by
$k$
 1: Initialize $targetPoint \leftarrow G$
 2: Convert $k$ to binary and store in $kBinary$
 3: for $i \leftarrow 1$ to $length(kBinary) - 1$ do
 4:     $currentBit \leftarrow kBinary[i]$
 5:     $targetPoint \leftarrow doublePoint(targetPoint, p)$
 6:     if $currentBit = 1$ then
 7:         $targetPoint \leftarrow addPoints(targetPoint, G, p)$
 8:     end if
 9: end for
10: return $targetPoint$
```

Edge Layer (E)	Fog Layer (F)
Input $\{a, b, G, n, q\}$	
Choose Secret Integer	
$S_e \leftarrow CSPRNG$	
Compute Public Key	*Input* $\{a, b, G, n, q\}$
$\mathbf{A_e} = \mathbf{G} \times S_e(modq)$	**Choose Secret Integer**
$\underbrace{A_e}_{\text{public key}} \quad \xrightarrow{\qquad\qquad}$	$S_f \leftarrow CSPRNG$
	Compute Public Key
	$\mathbf{A_f} = \mathbf{G} \times S_f(modq)$
	Compute Session Key
	$SK_f = \mathbf{A_e} \times S_f(modq)$
	$\xleftarrow{\qquad\qquad} \quad \underbrace{A_f, SK_f}_{\text{public, session key}}$
Compute Session Key	
$SK_e = \mathbf{A_f} \times S_e(modq)$	
$SK_e = SK_f$ (key matched)	
$\underbrace{ACK}_{\text{authentication validated}} \quad \xrightarrow{\qquad\qquad}$	

FIGURE 3 | Secure authentication key exchange (SAKE) mechanism.

The full process of the SAKE mechanism between the edge (E) and fog (F) layer is in Figure 3 and is explained below:

Step 1: The parameters used for the elliptic curve are initially agreed upon by both parties as $(a, b, \boldsymbol{G}, \mathbf{n}, \mathbf{q})$. The following describes the parameters: a, b are the parameters that define the curve in the formula $y^2 = x^3 + ax + b$; $\boldsymbol{G}$ is the base point of the curve; n is the smallest possible integer such that nG equals to the identity element of the group formed by the points on the curve; q is the modulo prime number. The secret key S_e is randomly generated using a Cryptographically Secure Random Number Generator (CSPRNG) to ensure the key is as unpredictable as possible. The shared public key $\mathbf{A_e}$ is derived by $\mathbf{G} \times S_e$ using the double-and-add method and is sent to the fog layer.

Step 2: The fog layer receives the shared key $\mathbf{A_e}$ from the edge layer. It does a similar process, generates its own secret S_f from a CSPRNG, then calculates its own shared key $\mathbf{A_f}$ by $\mathbf{G} \times S_f$. A new session key SK is calculated using the shared public key and its own secret key $\mathbf{A_e} \times S_f$. The new session key SK and the fog layer's shared key A_f are returned to the edge layer.

Step 3: The edge layer then authenticates the edge layer by validating the session key. If the shared key times the secret key $A_f \times S_e$ is equal to the session key SK, the authentication process is successful, and the request is considered authentic, or else the request is invalid. Finally, an acknowledgement is sent back to the fog layer.

Step 4: Once the fog layer receives the acknowledgement, a secure channel is established, and both layers can start exchanging messages.

4.2 | Homomorphic Encryption

After successful authentication, a secure channel is established, and both parties can start data transmission. The proposed scheme uses public key encryption before the transmission of data. The full process is described below:

Step 1: The edge layer sends the public key to the fog layer.

Step 2: The fog layer uses the public key to encrypt the message, and the ciphertext is returned to the edge layer.

Step 3: The edge layer receives and decrypts the ciphertext back to the message, which can then be used for analysis or processing.

Ring Learning With Error (RLWE), which is a variant of LWE, is used for the above process. RLWE utilises polynomial rings over a finite field, moving from integers $\mathbb{Z}_q^n$ to polynomial quotient rings $\left(\mathbb{Z}_q[x]/\langle x^n + 1\rangle\right)$. This means the operations are done in polynomials instead of matrices in traditional LWE. By utilising the number-theoretic transform (NTT) [23], which effectively multiplies two polynomials with their integer coefficients, the operations of RLWE are more efficient than traditional LWE. Furthermore, RLWE can directly encrypt plaintext by turning the plaintext into a plaintext polynomial; these properties make RLWE a better approach than LWE in practice.

The parameters for RLWE are the polynomial degree n, modulo prime number q, the error distribution $\mathcal{X}$ over q, and plaintext modulo t. The key generation process (as depicted in algorithm 2) creates a set of equations, with matrices (A, b) as the public key and secret vector s as the secret key, with the operations done in polynomials.

ALGORITHM 2 | RLWE Key Generation.

```
1) Choose $A$ uniformly at random from $\mathbb{Z}_q^n$
2) Sample $s$ from $\chi^n$
3) Sample $e$ from $\chi^n$
4) Calculate $b = NTT(A, s) + e \mathrm{mod} q$
5) Return $(A, b)$ as the public key
6) Return $s$ as the private key
```

Encryption in RLWE is illustrated in Algorithm 3. At first, the message is encoded and scaled into a plaintext polynomial, and then two new errors are sampled from the distribution. The ciphertext is finally generated using these errors and a random vector r.

ALGORITHM 3 | RLWE Encryption.

```
1) Calculate $m = pt + 0 \times n - 1 \bmod t$
2) Calculate $\delta = q \bmod t$
3) Calculate $scaled_m = \delta \times m \bmod q$
4) Sample $e_1$ from $\chi^n$
5) Sample $e_2$ from $\chi^n$
6) Choose $r$ uniformly at random from $\mathbb{Z}_q^{n\times 2}$
7) Calculate $a' = NTT(A, r) + e_1 + NTT(\delta, scaled_m) \bmod q$
8) Calculate $b' = NTT(b, r) + e_2 \bmod q$
9) Return $(a', b')$ as the ciphertext
```

RLWE decryption is presented in algorithm 4 with the plaintext polynomial transformed back to the plaintext using the plaintext modulus. The decryption may fail, returning the wrong integer if the error sampled is too large, as this will lead to the calculated value being over the decoding range. Hence, the noise must be carefully balanced for the cryptosystem to work successfully.

ALGORITHM 4 | RLWE Decryption.

```
1) Calculate $scaled_{pt} = NTT(b', s) + a' \bmod q$
2) Calculate $pt = round\left(\frac{scaled_{pt} \times t}{q}\right) \bmod t$
3) Return $pt$
```

RLWE supports homomorphic operations, which means that the operations can be done on the encrypted data, as with pure data, without exposing any information about the actual data. More formally, any encryption scheme is called homomorphic if it satisfies the following [24]:

$$Dec\left(Enc(m_1, pk) \blacklozenge Enc(m_2, pk), sk\right) = m_1 \circ m_2 \tag{5}$$

Dec and Enc are decryption and encryption functions; $\blacklozenge$ and $\circ$ represent the operation in the corresponding ciphertext and plaintext space. Both messages are encrypted using the same key, as operations can only be done on the ciphertexts encrypted from the same public key. This property gives a huge advantage in IIoT, as security can be maintained while performing operations on the data. The actual data do not need to be shown in the layer that processes it; if the layer is compromised, sensitive information will not be leaked as it will remain as ciphertext; only the layers that contain the secret key or collect the data will have access. Two example use cases of homomorphic operations are detailed below:

In use case 1, as shown in Figure 4, the edge layer can distribute the same public key to all fog layers. All the fog layers can then

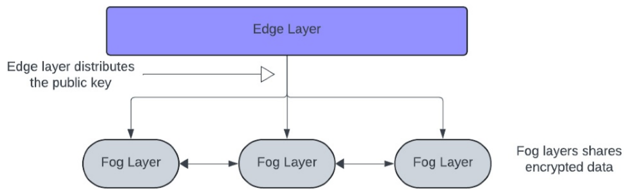
Edge Layer
Edge layer distributes the public key
Fog Layer
Fog Layer
Fog Layer
Fog layers shares encrypted data

FIGURE 4 | Homomorphic encryption use case 1.

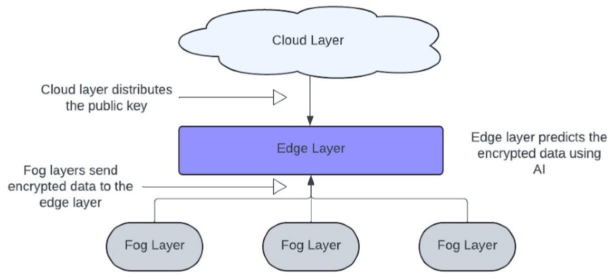
Cloud Layer
Cloud layer distributes the public key
Edge Layer
Edge layer predicts the encrypted data using AI
Fog layers send encrypted data to the edge layer
Fog Layer
Fog Layer
Fog Layer

FIGURE 5 | Homomorphic encryption use case 2.

communicate with each other and perform arithmetic operations on the data without knowing the actual data. If one of the fog layers is compromised, only the data from that particular layer will be leaked. This adds a level of security for the fog layers to communicate.

In use case 2, as shown in Figure 5, the cloud layer can distribute the public key to other layers. Each layer cannot read the data from other layers except the cloud. The edge layer collects the encrypted data from fog layers and predicts machine conditions, then returns the predicted encrypted data to the cloud layer.

4.3 | Intrusion Detection System (IDS)

Deep learning is a method in machine learning that teaches machines to learn from patterns of the data inspired by how human brains process. It uses multiple layers to extract features from the data by enlarging the number of channels. Hence, deep learning is highly effective in identifying patterns that humans may not recognise easily or use differently, such as patterns of attacks in network packets. For example, certain attacks have high duration, large packet size, or use different transmission protocols. Identifying these abnormal patterns in network packets can effectively classify the attacks.

4.3.1 | Models

This paper compares three models for IDS viz. MLP, CNN, and TabNet [17]. Figure 6 shows the flowchart of the implemented MLP model. The input layer takes the 2D data from the dataset and sends it to the linear layer. The linear layer learns the data details and is passed to the batch normalisation layer, where the data is normalised for training stability. This is then followed by a ReLU activation layer to introduce non-linearity to the data. The linear, batch normalisation and activation layers are repeated three times. Finally, an activation function is used to turn the output into a probability mass function (PMF), sigmoid is for binary classification, and softmax is for multi-class classification.

The CNN model shown in Figure 7 is similar to the MLP model. They have nearly the same architecture, with the hidden layers repeated three times. Instead of the linear layers, 1D convolutional layers are used, which perform convolution operations over the one-dimensional sequence of the network packet. It uses sliding kernels over the data to capture and learn the patterns, for example, packet length or duration. This process

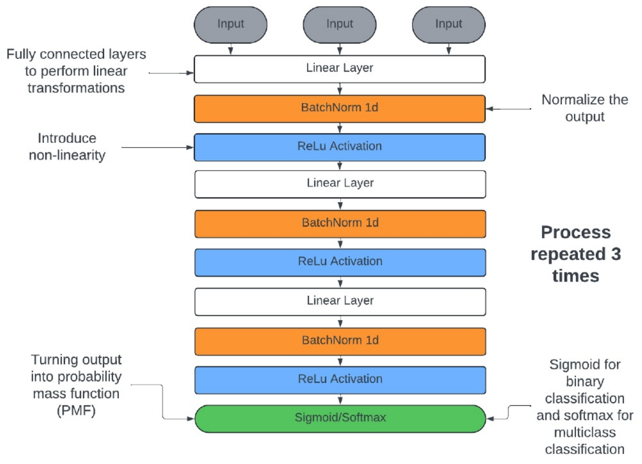

FIGURE 6 | MLP model architecture.

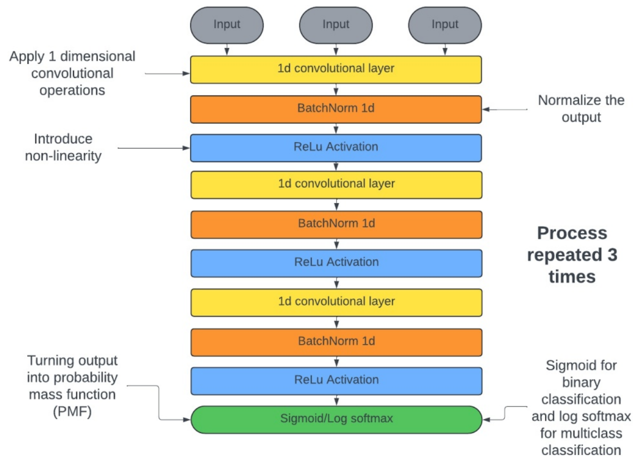

FIGURE 7 | CNN model architecture.

also reduces the sequence length and increases the feature dimension, which can help the model to learn more efficiently.

TabNet [17] is a deep neural network architecture that specifies tabular data and utilises sequential attention to extract and select the most important features in tabular data. The architecture, as detailed in ref. [17], consists of feature transformer blocks using denser layers, normalisation, feature selection, and ghost batch normalisation to process and interpret the input data; additionally, the attentive transformer blocks use learned masks to hide less relevant features and select the more important features. TabNet can be used for regression and classification tasks, with the final activation layers adjusted to sigmoid or softmax functions for binary or multi-class classification.

4.3.2 | Dataset

To compare the performances of the deep learning models, the ToN_IoT dataset [20] is used which consists of heterogeneous data collected from a simulation of IIoT devices in a large-scale testbed network. The data set contains three main types of files: network packets, operating system log files, and IoT device data. Each file has normal data and simulated attack scenarios:

Scanning attack, DoS attack, DDoS attack, ransomware attack, backdoor attack, injection attack, XSS attack, password cracking attack, and man-in-the-middle (MITM) attack. The network packet data used for experimentation by the proposed models comprises connection, statistical, DNS, SSL, HTTP, violation, and labeling features. The connection features contain information about the identifiers; the statistical features contain the sizes; the DNS attributes show the interactions and high-level services; the HTTP and SSL features present the information of the SSL and HTTP used; violation features include any unusual activity occurred during transmission; finally, each row of data is labeled with 0 for normal record, or 1 for attack, and the type of attack categories.

5 | Experimentation and Analysis

This section presents the experimental results and compares the proposed scheme with state-of-the-art. In the proposed scheme, ECC (specifically ECDH) is used for key exchange authentication, lattice-based operations are used for homomorphic data transmission encryption and deep learning models are used as IDS to classify attacks on network packets. The source code is available at github[1] for reproducibility.

5.1 | Key Exchange Authentication Analysis

To validate ECDH's efficacy, performance is compared with the Rivest–Shamir–Adleman (RSA) cryptosystem, a widely used method for public key encryption that has long been a standard in the National Institute of Standards and Technology (NIST). The comparison was made by comparing the public key generation time.

Based on the NIST recommended key size [25], shown in Table 2, the performance is evaluated relative to the security bits of symmetric key sizes.

To ensure security is maintained during experimentation, the curve parameters are chosen based on well-established, proven secure by experts and validated in NIST standards. While the performance may be slightly dependent on these parameters, those variants are not expected to affect the result's robustness significantly. The following presents the curves to be tested on [26–28]:

- SECP160r1

TABLE 2 | NIST recommended key size.

Security bits	RSA key sizes (bits)	ECDH key sizes (bits)
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15,360	512

$$y^2 = x^3 - 3x + 1632357913061681105466049194032715795305483454 13$$

- NIST224p

$$y^2 = x^3 - 3x + 18958286285566608000408668544493926415504680968679321075787234672564$$

- SECP256k1

$$y^2 = x^3 + 0x + 7$$

- NIST384p

$$y^2 = x^3 - 3x + 27580193559959705877849011840389048093056905856361568521428707301988689241309860865136260764883745107765439761230575$$

- BRAINPOOLP512r1

$$y^2 = x^3 + 6294860557973063227666421306476379324074715770622746227136910445450301914281276098027990968407983962691151853678563877834221834027439718238065725844264138x + 3245789008328967059274849584342077916531909009637501918328323668736179176583263496463525128488282611559800773506\ 9737717977648114988349952343415308622866 27$$

By plotting public key size against security bits, the key sizes of RSA are notably larger than ECC's key sizes as shown in Figure 8. This suggests that even if the processing time of both ECC and RSA key generation is the same, ECC will require less key generation time for the same security level.

Figure 9 shows the result of key generation time between RSA and ECC. RSA takes significantly longer to generate the key with security bits similar to ECC's. For 80 bits of security, the ECC key was only generated in 0.009 s, while the RSA key was

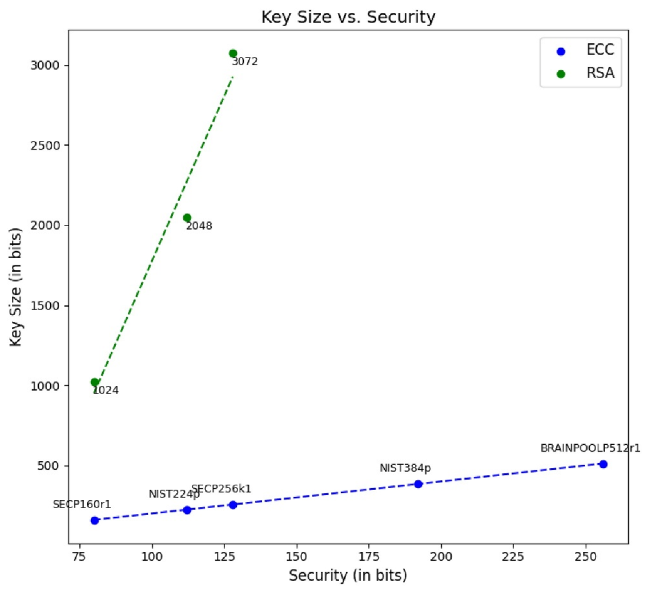

FIGURE 8 | Key sizes between RSA and ECC.

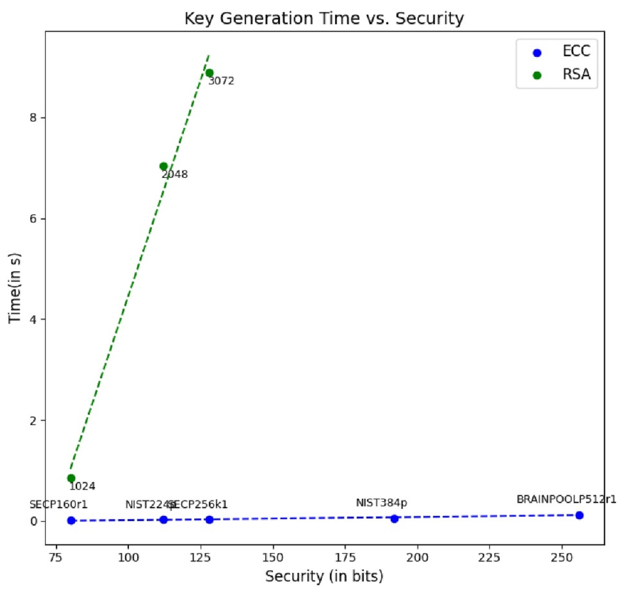

FIGURE 9 | Comparison between RSA and ECC with key generation time.

generated in 0.4 s. As the security bit increases, the difference between the key generation time of RSA increases exponentially. At the largest security value of 256 bits, the ECC key only took 0.12 s, with a difference of only 0.111 s; the RSA key took hours to generate and was not included in the graph to maintain scale consistency.

5.2 | Homomorphic Encryption Analysis

Lattice-based encryption cannot be directly compared to traditional encryption schemes such as RSA or AES as they operate on different security assumptions and classes of computationally hard problems. The difference in security assumptions makes lattice-based encryption considered to be quantum secure. Hence, comparisons are made between standard LWE and RLWE in terms of encryption time and ciphertext size. To ensure fair results, the encryption will be performed on the plaintext integer 2^{20}. As LWE cannot directly encrypt an integer, the plaintext is first turned into a binary array, and encryption is applied to each bit. For RLWE, the plaintext modulo is 2^{22}; a larger modulo may decrease the computational efficiency, while decryption will fail if the modulo is smaller than the plaintext.

LWE and RLWE parameters must be carefully chosen to ensure secure encryption. Hence, the parameters of popular schemes are used as they have been studied and verified by experts in the field. The following presents the parameters of the schemes [29–32]:

- Kyber512:
  $n = 2 \times 256,$
  $q = 3329, Xs = CenteredBinomial(3, n), Xe = CenteredBinomial(3, n), m = 2 \times 256$
- Kyber768:
  $n = 3 \times 256, q = 3329, Xs = CenteredBinomial(2, n), Xe = CenteredBinomial(2, n), m = 3 \times 256$
- Kyber1024:
  $n = 4 \times 256, q = 3329, Xs = CenteredBinomial(2, n), Xe = CenteredBinomial(2, n), m = 4 \times 256$
- LightSaber:
  $n = 2 \times 256, q = 8192, Xs = CenteredBinomial(5, n), Xe = UniformMod(8, n), m = 2 \times 256$
- Saber:
  $n = 3 \times 256, q = 8192, Xs = CenteredBinomial(4, n), Xe = UniformMod(8, n), m = 3 \times 256$
- FireSaber:
  $n = 4 \times 256, q = 8192, Xs = CenteredBinomial(3, n), Xe = UniformMod(8, n), m = 4 \times 256$
- Frodo640:
  $n = 640, q = 2^15, m = 640 + 16, Xs = GaussianDistribution(2.8, n), Xe = GaussianDistribution(2.8, m)$
- Frodo976:
  $n = 976, q = 2^16, m = 976 + 16, Xs = GaussianDistribution(2.3, n), Xe = GaussianDistribution(2.3, m)$
- Frodo1344:
  $n = 1344, q = 2^16, m = 1344 + 16, Xs = GaussianDistribution(1.4, n), Xe = GaussianDistribution(1.4, m)$
- NewHope512:
  $n = 512, q = 12289, Xs = CenteredBinomial(8, n), Xe = CenteredBinomial(8, n)$
- NewHope1024:
  $n = 1024, q = 12289, Xs = CenteredBinomial(8, n), Xe = CenteredBinomial(8, n)$

The above schemes use different distributions for the error and secret key. The centered binomial distribution is the subtraction between two binomial distributions, leading to the mean being closer to 0. The uniform mod distribution samples from a uniform distribution over integers $\left[-\frac{q}{2}, \frac{q}{2} - 1\right]$. Finally, the Gaussian distribution function samples are from the discrete normal distribution.

Most schemes are designed for LWE's variant module learning with error (MLWE); for demonstration purposes, they'll be used for pure LWE, as both are believed to offer similar security against classical and quantum attacks. RLWE has an additional property of the ring structure to enhance computation efficiency, which could have led to vulnerabilities that could be exploited. However, there has yet to be a known attack that takes advantage of the structure and breaks RLWE more efficiently. Hence, the security of RLWE is generally similar to that of LWE, given that the parameters are carefully chosen.

In some implementations of the schemes, the decryption failed during experimentation. This is due to the value of the modulo q being too small, causing the noise of the error to be too large and exceeding the threshold for the algorithm to interpret the encrypted message correctly, leading to the incorrect decrypted value. Those schemes will not be shown in the results for RLWE. To validate the security of the schemes, a lattice estimator [33] is used to estimate the concrete securities. The lattice estimator uses different known attack algorithms and estimates the time needed to break those parameters. For each scheme, the security level is considered the minimum time for an attack to break.

Figure 10 shows the encryption time of LWE and RLWE on the plaintext relative to the security. Overall, the encryption time taken for RLWE is less than that for LWE for the same security. For 160 bits of security, LWE took around 0.009 s to encrypt, and RLWE took around 0.003 s, with a difference of 0.006 s. At 287 bits, LWE took around 0.047 s while RLWE only took 0.009 s; the difference has increased to 0.038 s. The regression line of LWE has a much steeper gradient, which shows a higher increase in encryption time than RLWE as the security increases.

For the ciphertext size, shown in Figure 11, the results are similar to encryption time, as LWE generally has a larger size than RLWE. Regardless, the difference was less, and the regression lines were closer than the encryption time. At 140 bits of security, the ciphertext size for LWE was about 7500 bytes, and the ciphertext for RLWE was about 6000 bytes, with a difference of 1500 bytes. At 275 security bits, the ciphertext size for LWE was about 20,000 bytes, while RLWE only needed around 13,400 bytes. The ciphertext produced by some schemes have a large residual with respect to the regression line, such as Frodo640, Frodo976, and Frodo1344; this shows that the choice of parameter selection can have a huge impact on the ciphertext sizes even if the schemes offer equivalent security levels.

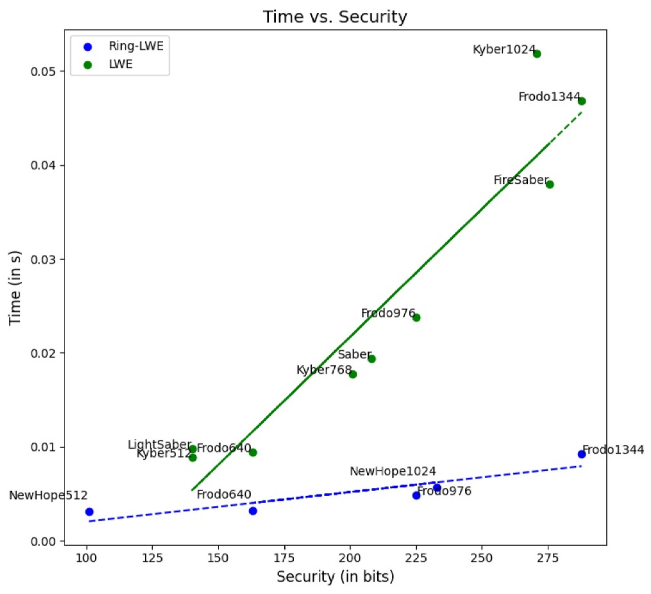

FIGURE 10 | Encryption time between LWE and RLWE.

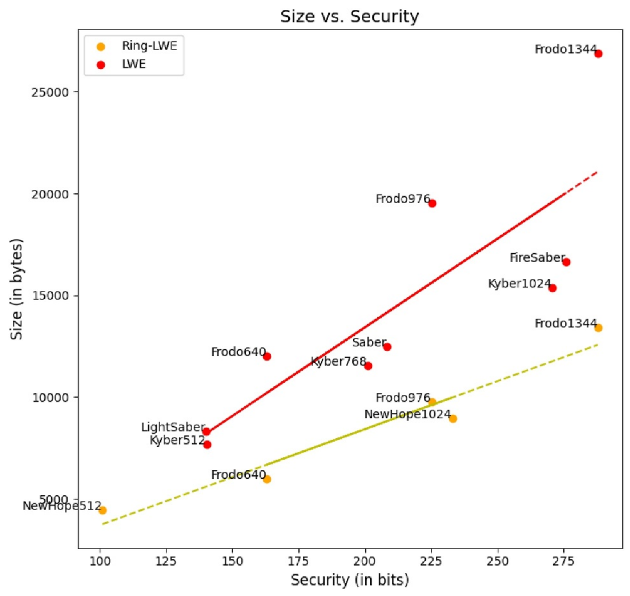

FIGURE 11 | Ciphertext sizes between LWE and RLWE.

5.3 | Intrusion Detection System Analysis

Three deep-learning models, as discussed in Section IV-C, were built and trained using Pytorch [34]. As direct training on the dataset is inefficient, data processing was done before feeding the models. The full process of the data processing is shown in Figure 12 and discussed as follows.

5.3.1 | Data Preprocessing

First, the dataset is imported; features such as source and destination IP addresses and ports are removed to ensure that the model will learn from the network packet patterns instead of overfitting in the source and destination information. Furthermore, a wrapper feature selection technique-based random forest [20] was used to estimate the influence of each feature and the features with the least importance are also removed. Figure 13 shows the importance of each feature, and features with near 0 scaled importance are removed. Duplicated data are dropped, and the dataset is shuffled to randomise the order. Label encoding is used on DNS queries, containing many unique values. Data with no record is filled with -1 to represent missing data. Dummy encoding is used for the rest of the

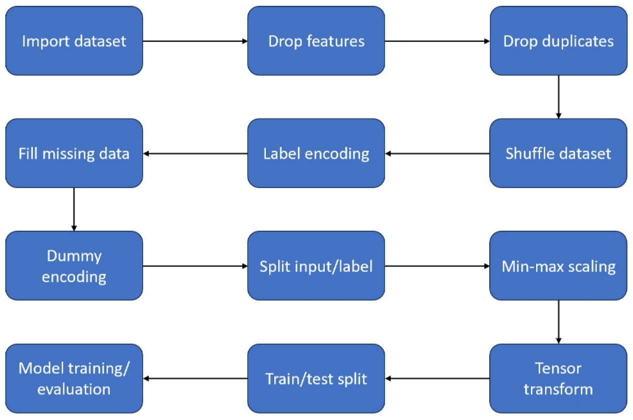

FIGURE 12 | Flowchart of preprocessing steps for network packet dataset.

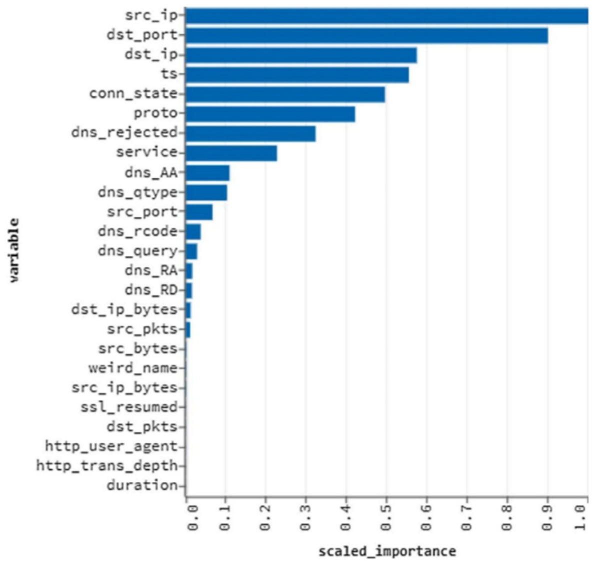

FIGURE 13 | Importance of each feature in the TON_IoT dataset using wrapper feature selection technique-based RF.

non-integer columns. After the data processing, the dataset is split into input and label, where the label can be binary (0 for normal and 1 for attack) and multiclass (representing different attack types). The input is normalised using min-max scaling. The dataset is then transformed into Pytorch tensors and is split into train/test sets with a ratio of 70 **:** 30. Finally, the input data is fed into the models for training and evaluation.

5.3.2 | Performance Evaluation

Three evaluation metrics are used to evaluate the models' performance: Accuracy, Precision, and Recall. Accuracy measures the overall correctness of the true results, both true positives and true negatives; Precision measures the reliability by assessing the true positives out of all predicted positives; Recall measures the proportion of the true instances that were identified correctly, and this means it measures the ability to capture all the true positives. The formula for the three metrics is as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{6}$$

$$Precision = \frac{TP}{TP + FP} \tag{7}$$

$$Recall = \frac{TP}{TP + FN} \tag{8}$$

where TP, TN, FP, and FN are true positive, true negative, false positive, and false negative. An IDS's most important factor is its ability to identify any attacks performed; hence, the paper proposes Recall as the most important factor in measuring the models.

The results for binary classification are shown in Table 3. The parameters for the models were chosen using a hyperparameter tuner, which decided the parameters randomly and returned the ones that output the best results. All the models are trained for

TABLE 3 | Binary classification results on the three model trained.

Models	MLP	CNN	TabNet
Parameters	9850	18,585	15,320
Highest test accuracy %	94.83	94.83	95.28
Final test accuracy %	94.07	93.47	94.38
Highest test precision %	83.58	92.15	87.14
Final test precision %	75.27	75.90	75.79
Highest test recall %	86.04	89.53	91.86
Final test recall %	81.40	73.26	83.72

TABLE 4 | Multiclass classification results on the three models trained.

Models	MLP	CNN	TabNet
Parameters	9850	18,585	15,320
Highest test accuracy %	92.85	93.16	92.71
Final test accuracy %	92.86	92.55	92.40
Highest test precision %	93.59	92.96	94.34
Final test precision %	93.21	91.89	92.36
Highest test recall %	92.85	93.16	92.71
Final test recall %	92.86	92.55	92.40

1000 steps, as most of the models converged before 1000 steps during experimentation.

TabNet achieved the highest accuracy of all three models, with the highest test accuracy of 95.28% and the highest final test accuracy of 95.387%. CNN achieved the highest precision score with 92.15% and 75.90% for the highest and final test accuracy. TabNet has the highest recall, with the highest of 91.86% test recall and 83.72% as the final test recall. Overall, the difference between the three models was small, with some differences being less than 1%; it was also observed that some parameter tweaks or a different test/train split produced a different result in the experimentation. The number of parameters needed was relatively low, with at most 20,000; this shows that only a few model parameters are needed to achieve high results in the network packet classification task.

The parameters were kept the same for multiclass classification, and the activation layer was changed from sigmoid to softmax/log softmax. There's an overall decrease in accuracy and an increase in precision and recall. The results are shown in Table 4. CNN has the highest test accuracy and recall with 93.16% and 93.16%, and TabNet has the highest test precision with 94.34%. However, there's a drop as the training went on, as MLP had the highest final test accuracy, precision, and recall with 92.86%, 93.21%, and 92.86% respectively, suggesting MLP had the most stable training out of all three models.

5.3.3 | Attack Types Analysis

To evaluate how well the models performed against different attacks, each attack type was turned into a subset and was fed into the model. The accuracy of identifying the attacks was measured, and the results for MLP, CNN, and TabNet were shown in Figures 14–16 respectively.

In the results, all models were excellent at identifying normal packets, with each model having around 99% accuracy for normal packets. However, they were not able to identify all attacks. Some attacks had not been identified correctly by any models; MITM, scanning, and ransomware attacks had 0% accuracy out of all models. This is followed by password attacks where it only had 3.23% accuracy on the CNN model and 0% on the others. The attack with the highest accuracy was the XSS attack, with around 80% accuracy across all models. The second highest accuracy was DDoS attacks, with around 30% accuracy across all. CNN performed the best as it had 37.5% accuracy at backdoor attacks, 35.2% accuracy at DoS attacks, and more than

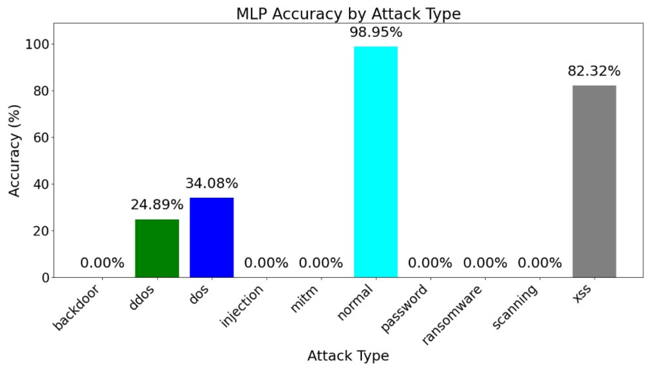

FIGURE 14 | MLP model accuracy against all attack types.

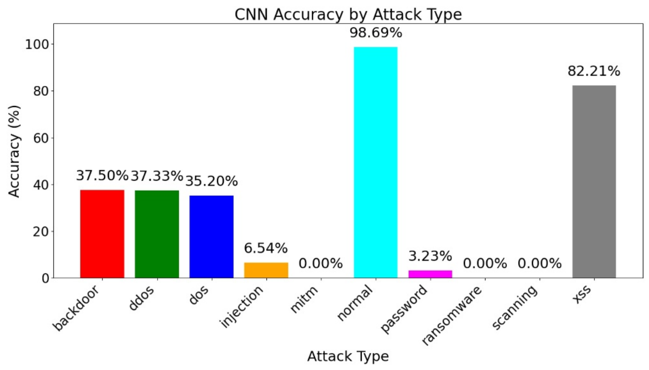

FIGURE 15 | CNN accuracy against all attack types.

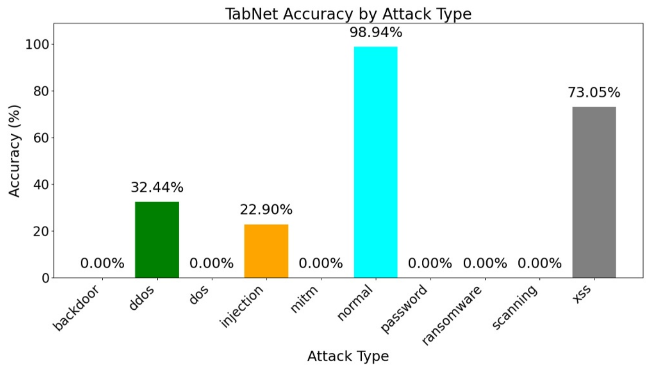

FIGURE 16 | TabNet model accuracy against all attack types.

0% accuracy at password attacks. Overall, the models achieved a high score from the metrics by identifying the patterns of normal packet data instead of learning from the attacks.

The dataset was analysed by plotting scatter graphs with attacks against each feature value to understand what might have affected the results. It was noted that normal packets behave differently from malicious packets; for example, some of the normal packets have a higher duration of packet connections than other attacks as shown in Figure 17. Normal packets have a higher number of original packets, as estimated from the source/destination systems and source/destination length of the IP header field as depicted in Figure 18. These differences allowed the models to capture the normal network packet patterns efficiently.

On the other hand, most attacks do not have a distinct pattern that can be identified as illustrated in Figure 19. The most noticeable attack is the DDoS attack, with the highest source/destination bytes originating from TCP sequence numbers payload bytes. Other attacks have similar patterns in the features across all types of packets and, therefore, cannot be easily identified from the graphs.

5.4 | Disucssion

Overall, the paper presented the security limitations in the current IIoT industry and suggested a general architecture for improvement. The ECDH key authentication system has shown

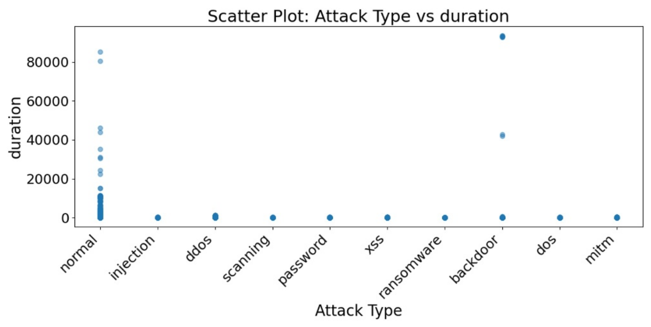

FIGURE 17 | The time of the packet connections against all attacks, normal packets have a higher duration than others.

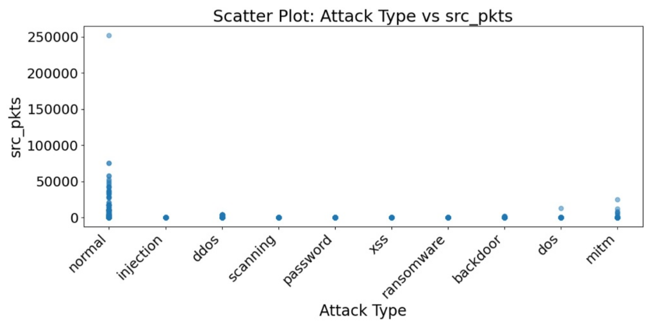

FIGURE 18 | Number of original packets estimated from source systems against all attacks, normal packets have a higher number of packets than other attacks.

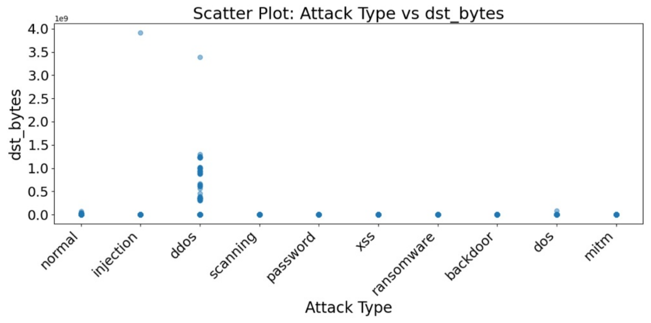

FIGURE 19 | Destination bytes which are responded payload bytes from TCP sequence numbers against all attacks, DDoS attacks have a higher destination bytes than the rest of the attacks.

improvement in efficiency over the current standard of using RSA. This allows for quicker authentication and less bandwidth usage. Homomorphic encryption gives the IIoT industry a flexible approach to data operations without sharing the content. However, as lattice-based encryption cannot be directly compared with symmetric key encryption, no evidence was obtained to suggest how much more computational resources must be implemented in the edge and fog layers. The use of deep learning models has shown good results in the classification of the attacks.

This paper assumed basic security measurements such as access control were implemented beforehand, as there's a lack of practical industry material to test; the actual implementation acted as a black box, such as data processing in different layers. Nevertheless, the three-layer architecture has suggested improvements that can be implemented in real-life applications.

6 | Conclusion

This paper proposes security mechanisms for a three-layer architecture for industrial IoT using lightweight key exchange authentication, homomorphic encryption, and deep learning-based intrusion detection systems. The ECDH key exchange has a more efficient key generation and smaller key size than the standard RSA cryptography. The homomorphic encryption system uses lattice-based encryption, and RLWE, which takes less encryption time and produces a smaller ciphertext size than traditional LWE. These properties make the proposed schemes a more suitable candidate for IIoT, as these devices have small computation power and communications are constantly occurring between different layers. The deep learning IDS models have achieved high performance in detecting intrusion in network packets, with TabNet achieving the highest recall score of 91.86% for binary classification and CNN with 93.16% for multiclass classification. However, analysis has shown that models effectively captured the pattern of normal packets but could not identify some specific attack types correctly.

In the future, ECDH can be integrated with digital certificates to provide a more secure authentication between two layers. Moreover, new deep-learning model architectures can be developed with better performance to detect all the attack types with high precision.

Author Contributions

Siu Ting Tsoi: data curation, formal analysis, investigation, methodology, resources, software, validation, visualization, writing – original draft. **Anish Jindal:** conceptualization, funding acquisition, investigation, methodology, project administration, supervision, validation, writing – original draft, writing – review and editing.

Conflicts of Interest

The authors declare no conflicts of interest.

Data Availability Statement

The data that support the findings of this study are openly available in TON_IoT Datasets at https://research.unsw.edu.au/projects/toniot-datasets.

Endnotes

[1] https://github.com/KRDecadeZero/csproject.

References

1. D. Evans, "The Internet of Things. How the Next Evolution of the Internet Is Changing Everything," Cisco Internet Business Solutions Group (IBSG), 2011, https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf.

2. S. Tweneboah-Koduah, K. E. Skouby, and R. Tadayoni, "Cyber Security Threats to Iot Applications and Service Domains," *Wireless Personal Communications* 95, no. 1 (2017): 169–185, https://doi.org/10.1007/s11277-017-4434-6.

3. K. Kimani, V. Oduol, and K. Langat, "Cyber Security Challenges for Iot-Based Smart Grid Networks," *International journal of critical infrastructure protection* 25 (2019): 36–49, https://doi.org/10.1016/j.ijcip.2019.01.001.

4. B. Guembe, A. Azeta, S. Misra, V. C. Osamor, L. Fernandez-Sanz, and V. Pospelova, "The Emerging Threat of Ai-Driven Cyber Attacks: A Review," *Applied Artificial Intelligence* 36, no. 1 (2022): 2037254, https://doi.org/10.1080/08839514.2022.2037254.

5. Y. Lu and L. D. Xu, "Internet of Things (Iot) Cybersecurity Research: A Review of Current Research Topics," *IEEE Internet of Things Journal* 6, no. 2 (2019): 2103–2115, https://doi.org/10.1109/jiot.2018.2869847.

6. H. Landaluce, L. Arjona, A. Perallos, F. Falcone, I. Angulo, and F. Muralter, "A Review of Iot Sensing Applications and Challenges Using Rfid and Wireless Sensor Networks," *Sensors* 20, no. 9 (2020): 2495, https://doi.org/10.3390/s20092495.

7. L. Bellehumeur, "The 5 Layers of Iot Architecture that Give it Super Power," Novote, accessed February 27, 2024, https://novotech.com/learn/m2m-blog/blog/2023/01/03/the-5-layers-of-iot-architecture-that-give-it-super-power/.

8. N. Kumar, A. Jindal, M. Villari, and S. N. Srirama, "Resource Management of Iot Edge Devices: Challenges, Techniques, and Solutions," *Software: Practice and Experience* 51, no. 12 (2021): 2357–2359, https://doi.org/10.1002/spe.3006.

9. Y. Lu and L. Da Xu, "Internet of Things (Iot) Cybersecurity Research: A Review of Current Research Topics," *IEEE Internet of Things Journal* 6, no. 2 (2018): 2103–2115, https://doi.org/10.1109/jiot.2018.2869847.

10. I. Lee, “Internet of Things (Iot) Cybersecurity: Literature Review and Iot Cyber Risk Management,” *Future Internet* 12, no. 9 (2020): 157, https://doi.org/10.3390/fi12090157.

11. Keyfactor, “Iot Device Security,” accessed, February 27, 2024, https://www.keyfactor.com/education-center/iot-device-security/.

12. K. A. Abuhasel and M. A. Khan, “A Secure Industrial Internet of Things (Iiot) Framework for Resource Management in Smart Manufacturing,” *IEEE Access* 8 (2020): 117.354–117.364, https://doi.org/10.1109/access.2020.3004711.

13. V. S. Miller, “Use of Elliptic Curves in Cryptography,” in *Conference on the Theory and Application of Cryptographic Techniques*, Lecture Notes in Computer Science, 218, (Springer, 1985), 417–426, https://doi.org/10.1007/3-540-39799-X_31.

14. V. Lyubashevsky, C. Peikert, and O. Regev, “On Ideal Lattices and Learning With Errors over Rings,” *Cryptology ePrint Archive, Paper 2012/230* (2012), https://eprint.iacr.org/2012/230.

15. A. Dua, R. Chaudhary, G. S. Aujla, A. Jindal, N. Kumar, and J. J. Rodrigues, “Lease: Lattice and Ecc-Based Authentication and Integrity Verification Scheme in E-Healthcare,” in *2018 IEEE Global Communications Conference (GLOBECOM)* (IEEE, 2018), 1–6.

16. P. Novotney, *Weak Curves in Elliptic Curve Cryptography*, (2010), https://www.wstein.org/edu/2010/414/projects/novotney.pdf.

17. S. Ö. Arik and T. Pfister, “Tabnet: Attentive Interpretable Tabular Learning,” *Proceedings of the AAAI Conference on Artificial Intelligence* 35, no. 8 (2021): 6679–6687, https://doi.org/10.1609/aaai.v35i8.16826.

18. P. Wu, N. Moustafa, S. Yang, and H. Guo, “Densely Connected Residual Network for Attack Recognition,” in *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)* (IEEE, 2020), 233–242.

19. I. Tareq, B. M. Elbagoury, S. El-Regaily, and E.-S. M. El-Horbaty, “Analysis of Ton-Iot, Unw-Nb15, and Edge-Iiot Datasets Using Dl in Cybersecurity for Iot,” *Applied Sciences* 12, no. 19 (2022): 9572, https://doi.org/10.3390/app12199572.

20. N. Moustafa, “A New Distributed Architecture for Evaluating Ai-Based Security Systems at the Edge: Network Ton_iot Datasets,” *Sustainable Cities and Society* 72 (2021): 102994, https://doi.org/10.1016/j.scs.2021.102994.

21. I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, “Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization,” *ICISSp* 1 (2018): 108–116.

22. N. Moustafa and J. Slay, “Unsw-nb15: A Comprehensive Data Set for Network Intrusion Detection Systems (Unsw-nb15 Network Data Set),” in *2015 Military Communications and Information Systems Conference (MilCIS)* (IEEE, 2015), 1–6.

23. Z. Liang and Y. Zhao, “Number Theoretic Transform and its Applications in Lattice-Based Cryptosystems: A Survey,” *arXiv preprint arXiv:2211.13546* (2022).

24. G. Peralta, R. G. Cid-Fuentes, J. Bilbao, and P. M. Crespo, “Homomorphic Encryption and Network Coding in Iot Architectures: Advantages and Future Challenges,” *Electronics* 8, no. 8 (2019): 827, https://doi.org/10.3390/electronics8080827.

25. W. T. Polk, D. F. Dodson, W. E. Burr, H. Ferraiolo, and D. Cooper, “Cryptographic Algorithms and Key Sizes for Personal Identity Verification,” *NIST Special Publication* 800, no. 78–4 (2015): 1–19, http://dx.doi.org/10.6028/NIST.SP.800-78-4.

26. M. Qu, “Sec 2: Recommended Elliptic Curve Domain Parameters,” *Certicom Res.* (1999).

27. FIPS PUB 186-4, *Digital Signature Standard (DSS)*, (National Institute of Standards and Technology, Tech. Rep., 2013), https://doi.org/10.6028/NIST.FIPS.186-4.

28. M. Lochter and J. Merkle, “Elliptic Curve Cryptography (Ecc) Brainpool Standard Curves and Curve Generation,” *Tech. Rep.* (2010).

29. J. Bos, L. Ducas, E. Kiltz, et al., “Crystals-kyber: A Cca-Secure Module-Lattice-Based Kem,” in *2018 IEEE European Symposium on Security and Privacy (EuroS&P)* (IEEE, 2018), 353–367.

30. J.-P. D’Anvers, A. Karmakar, S. Sinha Roy, and F. Vercauteren, “Saber: Module-Lwr Based Key Exchange, Cpa-Secure Encryption and Cca-Secure Kem,” in *Progress in Cryptology–AFRICACRYPT 2018: 10th International Conference on Cryptology in Africa, Marrakesh, Morocco*, (Springer, 2018), 282–305.

31. J. W. Bos, C. Costello, M. Naehrig, and D. Stebila, “Post-Quantum Key Exchange for the Tls Protocol From the Ring Learning With Errors Problem,” in *2015 IEEE Symposium on Security and Privacy* (IEEE, 2015), 553–570.

32. J. Bos, C. Costello, L. Ducas, et al., “Frodo: Take off the Ring! Practical, Quantum-Secure Key Exchange From Lwe,” in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (2016), 1006–1018.

33. M. R. Albrecht, R. Player, and S. Scott, “On the Concrete Hardness of Learning With Errors,” *Journal of Mathematical Cryptology* 9, no. 3 (2015): 169–203, https://doi.org/10.1515/jmc-2015-0016.

34. A. Paszke, S. Gross, F. Massa, et al., “Pytorch: An Imperative Style, High-Performance Deep Learning Library,” *Advances in Neural Information Processing Systems* 32 (2019).