MACHINE LEARNING FOR ANOMALY DETECTION IN ENCRYPTED NETWORK TRAFFIC

MITACS GLOBALINK RESEARCH INTERNSHIP REPORT

Submitted By:

Manigandan Ramadasan

Supervisor:

Dr. Hakima Ould-Slimane

Université du Québec à Trois-Rivières

August 2023

ABSTRACT

Anomaly detection in network traffic is crucial for identifying abnormal patterns and behaviours that indicates potential cybersecurity threats. The increase in cyber-attacks and the complexity of modern network infrastructures have led many organizations to outsource their anomaly detection tasks to third parties. However, this can expose sensitive information, leading to potential privacy breaches. In response to these challenges, Machine Learning models such as Logistic Regression, Support Vector Machine, Decision Tree and Neural Networks such as Artificial Neural Network and Convolution Neural Network is developed to detect anomalies on homomorphically encrypted data. By this approach, third parties are allowed to perform anomaly detection by carrying privacy preserving computations on homomorphically encrypted data without any privacy concerns. This research highlights the efficacy of these models for anomaly detection in an outsourced environment, while also comparing their relative performances.

Keywords: Anomaly detection, Outsourcing, Privacy Breaches, Machine Learning, Privacy preserving computation, Homomorphic encryption

CONTENTS

1. Introduction:

The increasing reliance on network communication and the growing complexity of cyber threats have made anomaly detection a critical aspect of network security which helps to keep the network secure from cyber-attacks. Usage of Machine Learning and Deep Learning to detect anomalous patterns (which has not been seen before) in network traffic plays a very crucial role in identifying potential security breaches, preventing cyber-attacks, and ensuring the confidentiality, integrity, and availability of network resources. However, as the need for robust anomaly detection mechanisms grows, so does the concern for preserving the privacy of sensitive network data. This is the case when anomaly detection tasks are outsourced to third-party service providers where protecting the privacy of network traffic becomes even more critical.

Homomorphic Encryption is a cryptographic technique which had many developments in last decade and it is a promising solution for preserving the privacy of sensitive data by allowing computations to be performed on ciphertexts in a way that decrypting the result of computation performed on the ciphertexts will be the same as the result if the computation were applied on plaintexts. By leveraging the power of homomorphic encryption, organizations can outsource anomaly detection tasks to third-party service providers without revealing the underlying network traffic. This approach enables privacy-preserving anomaly detection, allowing organizations to enhance their security and safeguarding sensitive information about their network.

In this project, we present the results of extensive experimentation designed to assess the feasibility and effectiveness of using homomorphic encryption in detecting anomalies in network traffic flow using Machine Learning. Utilizing a range of Machine Learning techniques, the potential of homomorphic encryption in creating a more secure, privacy-preserving approach to network anomaly detection is demonstrated, thus opening new avenues for future research and application.

The organization of rest of the report is as follows: The Literature Survey is summarized in Section 2. In Section 3, the preliminaries are described which is required to understand rest of the report. The architecture adopted in the project is described in Section 4 and Section 5 discusses about the algorithms employed and its training results. Experimentation details and encrypted evaluation results are discussed in detail in Section 6 and Section 7 brings the report to a close with a comprehensive conclusion, encapsulating key findings and implications of the research and by discussing the future works.

2. Literature Review:

Anomaly detection on encrypted traffic using Machine Learning and Deep Learning has been a major domain of research for past years and variety of methods using NN, CNN, RNN, Autoencoders, GAN, Graph CNN etc., has emerged. [15, 22, 24]. A 2D CNN algorithm which performs malware classification on encrypted data which uses USTC-TRC2016 dataset is proposed in [6] where first 784 bytes of the flow or session and perform binary classification, if it is found to be malware, they further classify it. This was further improved in [8] by a novel 1D CNN method to classify encrypted traffic based on application which will help to detect

anomalies. In [20], a method for DPI using Random Forest for VPN traffic and Naïve Bayes for SSL traffic is proposed. Time series-based classification using Machine Learning techniques that employed SVM, Decision Tree and KNN was proposed on encrypted network traffic [21] and Deep learning technique which employed LSTM [14] to classify encrypted network traffic also gave high accuracy.

When it comes to Machine Learning and Deep Learning techniques in homomorphically encrypted data, CryptoNets [5] was he first Neural Network developed which used somewhat homomorphic encryption which was trained on MNIST data and used square activation function along with scaled-mean pool function for pooling operation and this was further improved in [7] through batch normalization technique. In [2], a method to perform comparison operation, argmax, sign determination and dot product in bit-level was developed and Naïve Bayes, Hyperplane decision classifier and decision trees was developed using the proposed operations. An efficient FHE scheme with efficient noise controlling techniques was proposed along with comparison protocol efficient than [2] and the proposed FHE was used to develop Hyperplane decision classifier, Naïve Bayes and Decision Tree algorithm that has less interaction between user and server [11].

The potential application of FHE in the domain of Machine Learning and Deep Learning which included Logistic Regression, Naïve Bayes, Decision Tree, Clustering and Neural Networks in genome sequence prediction and comparison is discussed in [19] and along with that, a method to perform Secure Genome-Wide Association study was also proposed using homomorphic encryption. Other applications and open areas in homomorphic encryption were discussed in [23] which includes MLaaS. [18] proposed a novel homomorphic encryption framework based on Conjugacy Search Problem over non-abelian and used it to develop Machine Learning models. Privacy preserving anomaly detection technique on encrypted traffic was proposed in [28] known as SlimBox that uses searchable encryption which works based on OXT scheme which consumes less bandwidth. The architecture in that work is somewhat like the architecture adopted in our work. Similarly, searchable encryption was used to detect anomalies based on rules in IoT in [29]. A novel FHE scheme which is specifically developed for Neural Networks was proposed in [12] and they also implement a protocol for private information retrieval. [4] proposes a deep learning model which uses back propagation by using BGV scheme for encryption for big data feature learning.

An efficient anomaly detection system for industrial control systems using SVM that uses pallier cryptosystem which is additively homomorphic was proposed in [26] which follows client-server architecture. Similarly, anomaly detection in smart cities using BGV scheme which is one of the homomorphic encryption schemes was developed which uses MapReduce to distribute the tasks and parallelize computations to overcome the overhead caused by homomorphic encryption [16]. A method which incorporates federated learning and homomorphic encryption where each organization trains the data locally using Isolation Forest Algorithm to detect outlier (i.e., anomalies) and shares partial information which is encrypted homomorphically based on which a collective decision is taken to detect anomalies by the central server [27]. In [30], CKKS scheme is used to encrypt the data and anomaly detection was performed on the data by Logistic Regression which was trained using NSL KDD dataset.

This review signifies the importance of homomorphic encryption and its open areas along with its application in anomaly detection which has a large potential.

3. Preliminaries:

3.1 Homomorphic Encryption and CKKS Scheme

Homomorphic Encryption is a form of encryption that allows computations to be performed on encrypted data without decrypting it. This kind of encryption is critical for privacy-preserving computations and secure data outsourcing. A noise is applied during the encryption process to ensure the security of the data. After every operation performed on the ciphertext, the amount of noise present in the ciphertext increases which when crosses a threshold, gives incorrect results after decryption. Fully Homomorphic Encryption is a type of homomorphic encryption where any number of operations can be performed. There is another type of homomorphic encryption which is Levelled Homomorphic Encryption which is a slightly restricted version of Fully Homomorphic Encryption where the number of operations that can be performed on the encrypted data is limited and predetermined by a parameter set during the encryption process. This is the type of encryption used in this project as the number of operations will be known beforehand.

Cheon-Kim-Kim-Song (CKKS) scheme [9] is a type of homomorphic encryption scheme that is designed for performing operations on real and complex numbers. CKKS operates in the complex plane and produces approximate results rather than exact results unlike BFV [1] or BGV [3]. CKKS scheme is designed to reduce noise growth and it supports batch processing of data, enabling significant improvements in computational speed and efficiency. Since the dataset contains floating point numbers, CKKS scheme is chosen to perform homomorphic encryption.

3.2 TenSEAL

TenSEAL [25] is a python library which is used for performing homomorphic operations on tensors. It provides high-level APIs for performing operations like addition, multiplication, convolution, etc., on encrypted data which is necessary for developing privacy preserving Machine Learning models to make predictions on encrypted data. TenSEAL is built on top of Microsoft's SEAL library, a leading C++ library for homomorphic encryption. TenSEAL supports both BFV and CKKS scheme. Since the dataset includes floating point numbers, CKKS scheme was chosen and choosing parameters in TenSEAL plays a crucial role to maintain correctness and compactness of the encryption. The hyperparameters include N (Polynomial Modulus), q (Coefficient Modulus) and scaling factor. The value of N represents the lattice dimension (security level) and size of ciphertexts produced and it is always a value which it powers of 2. The value of q is a list of numbers which indicates the multiplicative depth of the scheme. It is a list of numbers which represents the bit length of the prime numbers generated which is used for rescaling after every multiplication operation and the sum of all numbers in the list should have an upper bound as per the values mentioned in Table 1 [10]. Number of rescaling operations that can be performed is equal to the number of elements present in the list except the first and last element. The last element in the list is used for modulus switching which is not used in CKKS scheme. The difference between the first and

second element present in the list represents the bit precision allocated to the integer part of the floating-point number. The scaling factor represents the precision of the floating-point numbers, and its value will be same as the central values present in q to maintain the same precision after every rescaling operation. If there is a need to perform dot product, Galois key needs to be generated in addition to the public and secret key.

N	Bit length of default q (To maintain 128-bit security)
1024	29
2048	59
4096	110
8192	219
16382	441
32768	885

Table 1: Upper bound values of q with respect to N

3.3 Dataset

The dataset used in this research is CIC-IDS2017 [13] which contains 78 unique statistical features which defines the network flow. The reason to choose this dataset because it was designed to be representative of real-world data and scenarios and includes different types of attacks, such as Brute Force, Heartbleed, Botnet, DoS, DDoS, Web attacks, Infiltration etc., which makes it useful for training machine learning models for anomaly detection. The dataset was pre-processed through various techniques. The null and infinity values were dropped as it gave better results than filling the null and infinite values by the mean or median of column belonging to the particular attack class in order to preserve the characteristics of the attack or by KNN Imputation. All the duplicate instances were removed, and all the malicious samples were put into a single class making it a binary classification. The features with zero variance were dropped and highly correlated features were dropped through agglomerative clustering method which gave 30 highly important features at the end and the dataset was under sampled to avoid biases because the benign and malicious class was 83.11% and 16.89% respectively.

4. Architecture:

The proposed architecture for privacy-preserving anomaly detection in network traffic is depicted in Figure 1 where the network traffic from Internet, bounded to the Enterprise is routed through the gateway where feature extraction and encryption of the features is done. Upon completion of the encryption process, the encrypted features are sent to cloud server or third-party server for anomaly detection which is outsourced. The cloud server used the model weights and biases and performs computations on the encrypted data and send back the results. Then the results are decrypted and if any anomalous behaviour is found, necessary actions will be taken.

The architecture proposed operates under the assumption of an honest-but-curious threat model. Under this threat model, it is assumed that the third-party service provider or the cloud server will perform the tasks assigned to it as per the protocol correctly, but it might also attempt

to gain unauthorized knowledge from the sensitive data it processes. Despite this threat model, the privacy of the sensitive network data is still maintained due to the use of homomorphic encryption. Even though the server can perform computations on the data for anomaly detection, it does not have the ability to see the actual content of the data due to the encryption. Therefore, even if the server attempts to derive sensitive information from the data, it would only be analysing encrypted data, and any information extracted would be meaningless without the secret key.

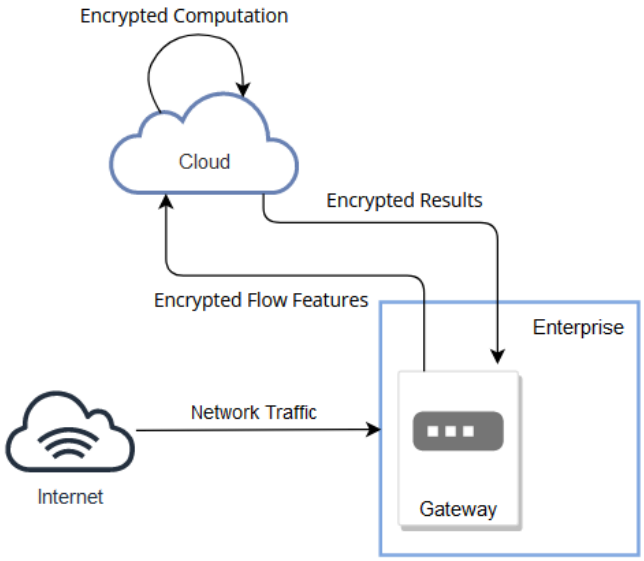

Figure 1: Architecture of the proposed model

5. Private Classification Algorithms:

Logistic Regression, Support Vector Machine, Artificial neural Network, Convolutional Neural Network and Decision Tree was chosen for anomaly detection on encrypted traffic as these algorithms are compatible with homomorphic encryption and these algorithms has given better results in previous research which was discussed in Section 2. Naïve Bayes algorithm was avoided as it considers each feature independently which gives negative results in anomaly detection. The training of the above-mentioned algorithms is done on plaintext data and their weights and biases are extracted (which can also be encrypted) which is used for prediction with homomorphically encrypted data. The details about hyperparameters and training results are discussed in further sub sections.

5.1 Logistic Regression

Logistic Regression is a supervised learning method which works by fitting a sigmoid function (Eqn. 1), to a set of data. The function estimates the probability of a particular outcome, given an input vector, $x = \{x_1, x_2, \ldots, x_n\}$ and a weight vector $w = \{w_1, w_2, \ldots, w_n\}$, where $x_i, w_i \in \mathbb{R}$. The output is a value between 0 and 1 and has a threshold at 0.5 to determine the binary class label (0 or 1).

$$Pr(y = 1|X = x) = \frac{1}{1+e^{-w^T x}} \quad \text{(Eqn. 1)}$$

Since homomorphic operation supports only linear operations, there is no direct way to implement sigmoid function. This can be solved if the sigmoid operation is performed at the client side or gateway whereas the server will be responsible for multiplying the weight and input vector and sending back the encrypted results. Standard Scaling was applied on the data and the model was trained using LBFGS optimizer (which converged faster than other optimizers such as SGD, Adam etc.) along with Binary Cross Entropy Loss function. The developed model has one linear dense layer which requires one multiplication operation and a sigmoid operation at the client side. The model was trained with a learning rate of 0.001 in 300 epochs and the training results are tabulated in Table 2 below.

Evaluation Metrics	Obtained Values (in %)
Accuracy	92.21
Recall	96.59
Precision	88.81
F1 – Score	92.54

Table 2: Evaluation of Logistic Regression

5.2 Support Vector Machine

Support Vector Machine or Hyperplane Decision based classifier is another supervised learning method which works by finding a hyperplane that best separates the classes in a high-dimensional feature space. The optimal hyperplane is the one that maximizes the margin, i.e., the distance between the hyperplane and the nearest data points from each class, which are known as support vectors and the data points are classified as per Eqn. 2 where $x = \{x_1, x_2, \ldots, x_n\}$ is the input vector, $w = \{w_1, w_2, \ldots, w_n\}$ is the weight vector and b is the bias, where $x_i, w_i, b \in \mathbb{R}$.

$$y = w^T.x + b = \begin{cases} -1, & y < 0 \\ +1, & y \geq 0 \end{cases} \quad \text{(Eqn. 2)}$$

Linear SVM with soft margin was developed as homomorphic encryption supports only linear function which restricted the use of kernels. Due to this restriction, linear kernel was used which is bad for anomaly detection. The model was developed with one layer along with Hinge loss function and Stochastic Gradient Descent optimizer which gave the better results compared to other optimizers. The model was trained on the data scaled by Standard Scaler with the learning rate of 0.0007, C value of 0.001 in 10 epochs and the training results are tabulated in Table 3 below.

Evaluation Metrics	Obtained Values (in %)
Accuracy	80.18
Recall	78.81
Precision	82.56

F1 – Score	80.64

Table 3: Evaluation of Support Vector Machine

5.3 Artificial Neural Network

Artificial Neural Network (ANN) is made of input layer, one or more hidden layers and an output layer. The process of training an ANN involves tuning the weights and biases of the network to iteratively improve the accuracy of the predictions made by the network which is achieved through backpropagation and optimization algorithm. The developed ANN has 3 layers – an input layers with 30 input features and 16 output features, one hidden layer with 16 input features and 8 output features and an output layer with 8 input features and 1 output feature with square activation function for the input layer and hidden layer as homomorphic encryption supports only linear function and it gave better results than approximation of other activation functions. The sigmoid activation function for the output layer will be performed at the client side after decrypting the results sent by the server. Adam optimizer along with Binary Cross Entropy with Logit Loss Function was used. The model was trained on the data scaled by Standard Scaler with a learning rate of 0.013 in 1000 epochs and the training results are tabulated in Table 4 below.

Evaluation Metrics	**Obtained Values (in %)**
Accuracy	96.86
Recall	98.75
Precision	95.15
F1 – Score	96.92

Table 4: Evaluation of Artificial Neural Network

5.4 Convolutional Neural Network

Convolutional Neural Network (CNN), traditionally successful in computer vision tasks, have emerged as effective tools for anomaly detection [6]. CNNs leverage spatial features in numerical data, learning to differentiate normal patterns from malicious patterns. The model has one Convolutional layer and two dense linear layers, with square activation function after every layer and sigmoid for the last layer which is performed at client side. The convolutional layer had to be limited to one because of the im2col (image to column) operation in TenSEAL. The im2col operation addresses this issue by transforming the convolution operation into a matrix multiplication operation, which is highly optimized in TenSEAL because it will become a simple dot product and bias addition which produces the result in vector format due to which pooling, or convolution operation can't be performed further, and only linear dense layers can be added to the network. Adam optimizer along with Binary Cross Entropy with Logit Loss Function was used. The model was trained on the data scaled by Standard Scaler with a learning rate of 0.001 in 30 epochs with a batch size of 64 and the training results are tabulated in Table 5 below.

Evaluation Metrics	Obtained Values (in %)
Accuracy	94.95
Recall	95.47
Precision	94.49
F1 – Score	94.98

Table 5: Evaluation of Convolution Neural Network

5.5 Homomorphic Comparison Operation

Comparison operation is one of the most important operations in most of the machine learning algorithms and since homomorphic encryption supports only linear functions, there is no direct way of comparing two homomorphically encrypted data. Cheon et al. (2019) in [17] proposed a new method for comparison for two word-wise homomorphically encrypted data which is has been adopted in this project. The main idea followed in their work is that the comparison of two numbers *x* and *y* can be evaluated a *f(x - y)* where *f* is the step function defined in Eqn. 3.

$$f(x) = \begin{cases} 1, & x > 0 \\ 0, & otherwise \end{cases} \quad \text{(Eqn. 3)}$$

Since step function is a discontinuous function which can be performed on homomorphically encrypted data, it can be approximated by scaling the sigmoid function with some factor *k*. Even after scaling the sigmoid function, we will be left with the calculation of exponential value which can be avoided by using logarithm since it preserves order i.e., if $x > y$ then $\log x > \log y$. So, the final equation (Eqn. 4) will be left with taking power and performing inverse operation.

$$comp(x, y) \approx \sigma_k(\log x - \log y) = \frac{e^{k \log x}}{e^{k \log x} + e^{k \log y}} = \frac{x^k}{x^k + y^k} \quad \text{(Eqn. 4)}$$

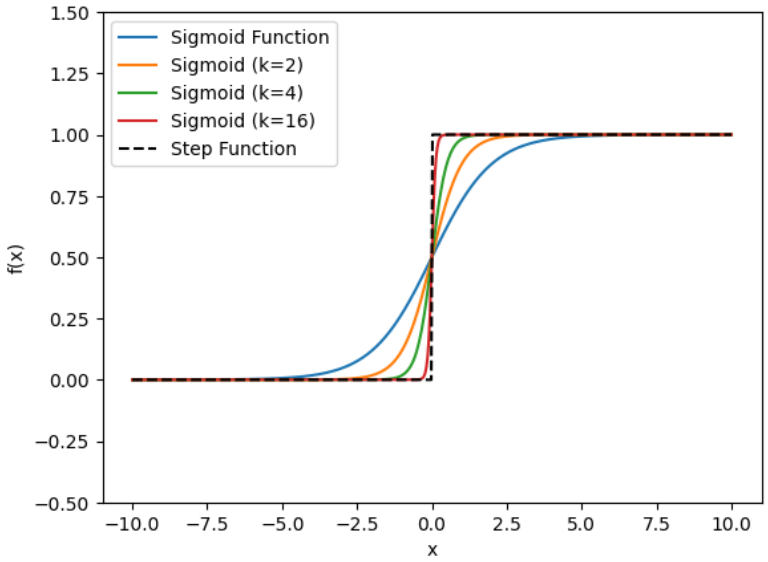

Figure 2: Step function approximation using scaled sigmoid functions

An efficient way to perform inverse operation is also proposed in [17] based on Goldschmidt's division algorithm (Eqn. 5) which can be approximated for sufficiently large value of $d > 0$.

$$\frac{1}{x} = \frac{1}{1-(1-x)} = \prod_{i=0}^{\infty}\left(1 + (1-x)^{2^i}\right) \approx \prod_{i=0}^{d}\left(1 + (1-x)^{2^i}\right) \qquad \text{(Eqn. 5)}$$

5.6 Decision Tree

Decision Tree is a supervised learning algorithm which is known for its high accuracy in anomaly detection, but the main obstacle is that decision tree requires comparison operation to traverse through the tree to arrive at a decision. Since homomorphic encryption doesn't provide direct way to evaluate comparison operation, comparison operation through approximation discussed in Section 5.5 is used to perform anomaly detection on homomorphically encrypted data. The decision tree was trained using scikit-learn library on the data scaled between the range $\left[\frac{1}{2}, \frac{3}{2}\right)$ to get best result from the comparison operation through approximation as suggested in [17]. The decision tree was trained with K-Fold Cross Validation of K = 3 to avoid overfitting of data as decision tree generally overfits the data along with log loss criterion which is used to find the best split and has a depth of 17. The training results for Decision Tree are tabulated in Table 6 below.

Evaluation Metrics	Obtained Values (in %)
Accuracy	99.63
Recall	99.66
Precision	99.60
F1 – Score	100.00

Table 6: Evaluation of Decision Tree

6. Experimentation Results:

All the experimental evaluations and model developments were conducted with Python and its libraries on 64-bit Windows operating system with Intel® Core™ i9-9880H at 2.3GHz with 32.0 GB RAM and the client server architecture was tested in 64-bit Linux operating system. The models were developed using PyTorch except for Decision Tree which was developed using scikit-learn.

The models are initially trained on unencrypted data. The weights and biases of all the layers are extracted and they are stored (which can be encrypted). These weights and biases are then multiplied with the homomorphically encrypted features then this encrypted result is sent back to the client or gateway where it is decrypted and sigmoid operation is applied to get the final prediction result. This is the case except for Decision Tree which is discussed in sub section 6.5.

Each algorithm was tested with different set of parameters to obtain the best parameters and the important evaluation metrics to be considered are accuracy and F1 – Score because F1 – Score implies about both recall and precision which is important for anomaly detection where

we don't want false positives or false negatives. The parameters were chosen in such a way that 128-bit security is maintained. The experimental results along with the time taken to evaluate one instance corresponding to each algorithm is discussed in further sub sections.

6.1 Logistic Regression

The developed Logistic Regression model has a one multiplication operation which implies that the multiplicative depth is one. So, the length of the list is kept to 3 since we need to rescale the ciphertext only once. The evaluation results of Logistic Regression with different parameters are tabulated in Table 7 below.

S. No.	Parameters			Evaluation Metrics (%)				Time (ms)
	N	q	Scaling Factor	Accuracy	Recall	Precision	F1-Score	
1	8192	31, 26, 31	26	91.31	96.42	87.25	91.60	13.27
2	8192	40, 31, 40	31	92.21	96.59	88.81	92.54	14.52
3	8192	60, 40, 60	40	92.21	96.59	88.81	92.54	16.97
4	16384	60, 40, 60	40	92.21	96.59	88.81	92.54	38.28
5	32768	60, 40, 60	40	92.21	96.59	88.81	92.54	75.40

Table 7: Evaluation of Logistic Regression on Encrypted Data with different parameters

In table 7, it can be noticed that (2) – (5) gives the same results and it is equal to the results obtained while testing with unencrypted data in table 2 and since there is only one multiplication operation, the noise growth is less, and it is handled well. While (1) has degraded output which means that when q is [31, 26, 31], precision is lost due to which incorrect results occurs after decrypting. From (2), it can be inferred that we need a minimum precision of 31 bits for fractional part and 9 bits (31 – 26) for integer part of the floating points number but since there can be a lot of unseen values in statistical value of real time network traffic, it is better to keep precision of 40 bits for fractional part and 20 bits (60 – 40) for integer part of the floating points number. As we increase the value of N while keeping the q same in (3) – (5), the results remain unchanged but the time for evaluating one instance doubles that is unnecessary which implies that the optimal value of N is 8192. So, the optimal parameters for logistic regression are N = 8192, q = [60, 40, 60] with a scaling factor of 40. From the confusion matrix in figure 3, it can be inferred that malicious flows are predicted as benign flow. This is because of less precision exhibited by the model.

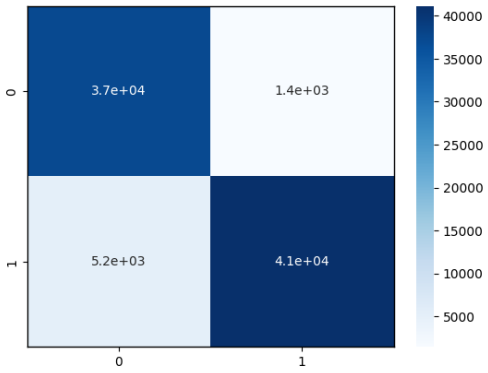

Figure 3: Confusion Matrix for Logistic Regression

6.2 Support Vector Machine

The developed Support Vector Machine model also has a one multiplication operation which implies that the multiplicative depth is one. So, the length of the list is kept to 3 since we need to rescale the ciphertext only once. The evaluation results of Support Vector Machine with different parameters are tabulated in Table 8 below.

S. No.	Parameters			Evaluation Metrics (in %)				Time (ms)
	N	q	Scaling Factor	Accuracy	Recall	Precision	F1-Score	
1	8192	31, 26, 31	26	79.98	78.11	81.53	79.78	15.84
2	8192	40, 31, 40	31	80.12	78.60	82.29	80.40	17.03
3	8192	60, 40, 60	40	80.18	78.81	82.56	80.64	17.61
4	16384	60, 40, 60	40	80.18	78.81	82.56	80.64	36.32
5	32768	60, 40, 60	40	80.18	78.81	82.56	80.64	71.98

Table 8: Evaluation of Support Vector Machine on Encrypted Data with different parameters

In table 8, it can be noticed that (3) – (5) gives the same results and it is equal to the results obtained while testing with unencrypted data in table 3 and since there is only one multiplication operation, the noise growth is less, and it is handled well. While (1) and (2) has degraded output which means that when q is [31, 26, 31] or [31, 26, 31], precision is lost due to which incorrect results occurs after decrypting. As we increase the value of N while keeping the q same in (3) – (5), the results remain unchanged but the time for evaluating one instance increases which suggests that the optimal value of N is 8192 and we need minimum precision of 40 bits for fractional part and 20 bits (60 – 40) for integer part of the floating points number

and the scaling factor of 40 is more than enough to maintain the precision. So, the optimal parameters for Support Vector Machine are N = 8192, q = [60, 40, 60] with a scaling factor of 40.

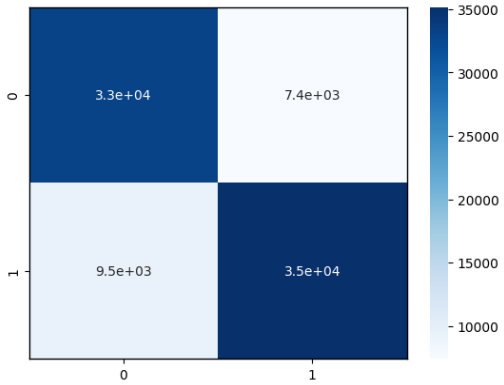

Figure 4: Confusion Matrix for Support Vector Machine

6.3 Artificial Neural Network

The developed Artificial Neural Network model has a multiplicative depth of 5. So, the length of the list is kept to 7 since we need to rescale the ciphertext five times after every multiplication operation. The evaluation results of Artificial Neural Network with different parameters are tabulated in Table 8 below.

S. No.	Parameters			Evaluation Metrics (in %)				Time (sec)
	N	q	Scaling Factor	Accuracy	Recall	Precision	F1-Score	
1	8192	40, 21, 21, 21, 21, 21, 40	21	94.03	92.65	95.19	93.90	0.191
2	8192	31, 26, 26, 26, 26, 26, 31	26	94.85	98.38	91.80	94.98	0.236
3	16384	60, 46, 46, 46, 46, 46, 60	46	93.79	98.09	90.13	93.94	0.490
4	16384	60, 40, 40, 40, 40, 40, 60	40	94.72	98.39	91.65	94.90	0.413

5	32768	60, 46, 46, 46, 46, 46, 60	45	93.77	98.09	90.13	93.94	1.317
6	32768	60, 40, 40, 40, 40, 40, 60	40	95.13	98.77	92.08	95.31	1.162

Table 9: Evaluation of Artificial Neural Network on Encrypted Data with different parameters

From table 9, it can be inferred that when N = 32768, q = [60, 40, 40, 40, 40, 40, 60] with a scaling factor of 40 gives the best results compared to other set of parameters. The results are not same as the results when tested on unencrypted data (table 4) because there are five multiplication operation due to which the noise grows after every multiplication, and it is rescaled due to which some precision bits are lost. Since it is done for five times, the final metrics on encrypted data are lesser than the metrics obtained when tested on unencrypted data. When N = 8192, we have a decent result, but it is lesser than the best parameters because of the bits allocated to the fractional and integer part of the floating-point number. In (1), bits allocated for fractional part is 21 and integer part is 19 (40 – 21) where the bits allocated for fractional part is less and in (2) bits allocated for fractional part is 26 and integer part is 5 (31 – 26) where the bits allocated for integer part is less which makes it to give results lesser than the one obtained using best parameters. This implies that there is always a trade-off of precision bits of integer and fractional part of the floating-point number. In (4), we have same value of q but N = 16382, we get better results when N = 32768 because we are increasing the space for encoding plaintexts, and thus potentially reducing the noise introduced by the encryption process. This is not necessary (higher value of N to accommodate the noise) in Logistic Regression and Support Vector Machine because of its low multiplicative depth.

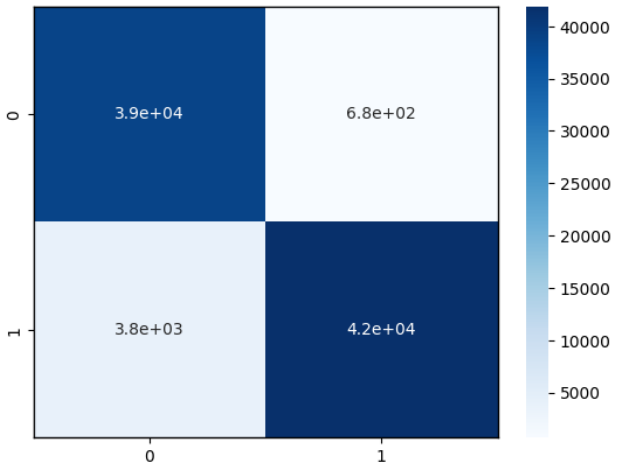

Figure 5: Confusion Matrix for Artificial Neural Network

6.4 Convolutional Neural Network

The developed Convolutional Neural Network model has a multiplicative depth of 6. So, the length of the list is kept to 8 since the ciphertext must be rescaled six times after every

multiplication operation. In the developed model, it can be noticed that there are five multiplication operation. The extra multiplication operation comes because of the im2col encoding which we perform using TenSEAL later which the ciphertext is packed together. The evaluation results of Convolutional Neural Network with different parameters are tabulated in Table 8 below.

S. No.	Parameters			Evaluation Metrics (in %)				Time (sec)
	N	q	Scaling Factor	Accuracy	Recall	Precision	F1-Score	
1	8192	40, 21, 21, 21, 21, 21, 21, 40	21	94.13	90.21	97.67	93.79	0.337
2	8192	31, 26, 26, 26, 26, 26, 26, 31	26	90.25	94.51	86.84	90.51	0.341
3	16384	40, 31, 31, 31, 31, 31, 31, 40	31	94.86	95.46	94.33	94.89	0.622
4	16384	60, 46, 46, 46, 46, 46, 46, 60	46	94.86	95.46	94.33	94.89	0.702
5	32768	60, 46, 46, 46, 46, 46, 46, 60	45	94.86	95.46	94.33	94.89	1.536
6	32768	60, 40, 40, 40, 40, 40, 40, 60	40	94.86	95.46	94.33	94.89	1.489

Table 10: Evaluation of Convolutional Neural Network on Encrypted Data with different parameters

From table 10, it can be inferred that in (1) and (2) when N = 8192, the evaluation on encrypted data is underperforming. This is because of low values present in q due to which precision is not maintained properly and this can't be increased when N = 8192 because 128-bit security must be maintained (table 1). So, we increase the value of N, and it can be noticed that (3) to (6) have the same result except the time taken to evaluate one instance. This is because the precision is preserved when rescaling operation is performed after every multiplication. So, the best results are obtained when N = 16384, q = 60, 46, 46, 46, 46, 46, 46, 60 with a scaling factor of 46 with a reasonable amount of time taken to evaluate one instance. This set of parameters is better than other parameters because in (2), only 9 bits (40 – 31) are allocated to

integer part of the floating-point number which can be a problem in real time network traffic and in (5) and (6), the time taken to evaluate one instance is too long.

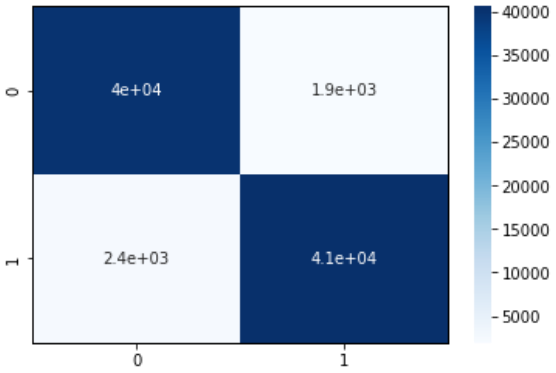

Figure 6: Confusion Matrix for Convolutional Neural Network

6.5 Decision Tree

In decision tree, the model is trained after which the internal structure of the decision tree is extracted. This includes the decision thresholds at each node and the features used for each decision. This is achieved through a recursive function that traverses through the decision tree where for each node in the tree, it checks if the node is a leaf node. If it's not a leaf node, it prints an if-else statement that corresponds to the decision at that node, and then recursively explores the left (true) and right (false) subtrees. If it is a leaf node, it prints a return statement with the value of the node. This generates a python function which consists of if-else statements with thresholds. This threshold is encoded into a list and the corresponding feature value is also encoded into another list and it is sent to the server to perform comparison operation. The server performs the comparison operation on homomorphically encrypted data, and the results are sent back to the client where it is decrypted and converted to Boolean values which is passed as an argument to the function which contains if-else statements of decision tree rules, and the final prediction result is obtained. The evaluation of if-else function is done at client side to reduce the client and server interaction.

S. No.	Parameters			Evaluation Metrics (in %)				Time (*min*)
	N	q	Scaling Factor	Accuracy	Recall	Precision	F1-Score	
1	32768	40, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 40	31	81.56	83.48	80.79	82.11	3.12

2	32768	40, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 40	31	78.92	80.95	78.11	79.50	3.27

Table 11: Evaluation of Decision on Encrypted Data with different parameters

From table 11, it can be inferred that the performance of Decision Tree is not impressive. This is because the comparison operation is generating more noise and the multiplicative depth of comparison operation is 21 due to which the noise accumulated, and time taken to evaluate one instance is too high. To perform 21 multiplication operation, the length of q must be increased to 23. To maintain 128-bit security, N had to be chosen as 32768 and the floating-point numbers had to be converted to 32-bit floating-point number to reduce computation overhead. (1) gives better results because 9 bits are allocated to integers (40 – 31) and 31-bit precision is maintained for fractional part with scaling factor of 31 which is the same for 32-bit floating-point numbers where 9 bits are allocated to integer part (including sign bit) and 23 bits for fractional part. When compared to other algorithms, Decision Tree is not a feasible option as the evaluation results are not up to the mark and the time taken to evaluate one instance is high.

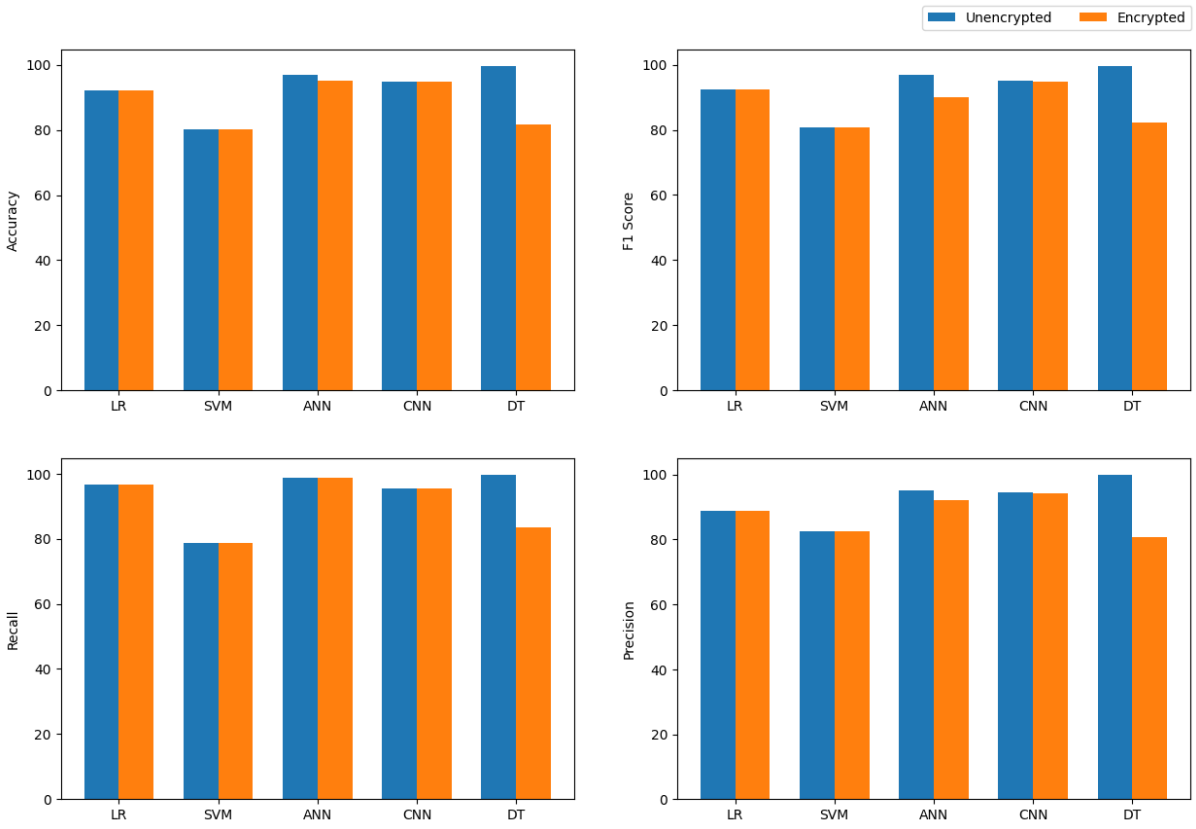

Figure 7: Evaluation metrics for all models in unencrypted Vs encrypted data

The models were also tested in client server environment where the client captures the packet using tcpdump tool and extracts features using CICFlowMeter. Then it cleans the data, encrypt it, and make the TenSEAL context public by dropping the secret key. Then the public context and the encrypted data is serialized and sent to the sever. The server is a Flask application which performs encrypted computation on the encrypted data after deserializing the client's data after loading the required model weight and biases. After performing the computation, the server sends back the encrypted output and then the client decrypt the result and applies sigmoid operation to obtain the final prediction result.

7. Conclusion

After extensive testing and evaluation on both encrypted and unencrypted data, it is found that the Artificial Neural Network (ANN) delivered the most accurate results on encrypted data relatively. However, the time taken to evaluate one instance was approximately 1.162s, given with its best parameters. This duration may be considered excessive for real-time scenario. On the other hand, the Convolutional Neural Network (CNN) presented a more balanced trade-off between metrics and evaluation time, and it had very less difference in metrics of encrypted and unencrypted data. With an evaluation time of 0.702s per instance with its best parameters, CNN offered a more efficient alternative to ANN while still maintaining satisfactory performance metrics.

In contrast, the performance of Logistic Regression and Support Vector Machine models fell short in comparison to ANN and CNN. They were unable to achieve comparable results, making them less preferable for anomaly detection. The Decision Tree model demonstrated promising results on unencrypted data. However, its performance drastically reduced when applied to encrypted data as the evaluation time went up to 3.12 minutes per instance with less accuracy. This extended evaluation period and low accuracy is due to the inherent inefficiencies related to comparison operations and noise growth in the encrypted data, making it infeasible for real-time scenario.

For future work, investigating ways to enhance the efficiency of comparison operations within encrypted data can be done in Python. By improving this aspect, we could potentially unlock more efficient and accurate ways of processing encrypted data, thus broadening the practical applications of encrypted machine learning algorithms especially for Decision Tree. This will enable organizations to harness the power of data while maintaining preserving the privacy.

8. References:

[1] Fan, J., & Vercauteren, F. (2012). Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive.

[2] Bost, R., Popa, R. A., Tu, S., & Goldwasser, S. (2014). Machine learning classification over encrypted data. Cryptology ePrint Archive.

[3] Yagisawa, M. (2015). Fully homomorphic encryption without bootstrapping. Cryptology ePrint Archive.

[4] Zhang, Q., Yang, L. T., & Chen, Z. (2015). Privacy preserving deep computation model on cloud for big data feature learning. IEEE Transactions on Computers, 65(5), 1351-1362.

[5] Gilad-Bachrach, R., Dowlin, N., Laine, K., Lauter, K., Naehrig, M., & Wernsing, J. (2016, June). Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In International conference on machine learning (pp. 201-210). PMLR.

[6] Wang, W., Zhu, M., Zeng, X., Ye, X., & Sheng, Y. (2017, January). Malware traffic classification using convolutional neural network for representation learning. In 2017 International conference on information networking (ICOIN) (pp. 712-717). IEEE.

[7] Chabanne, H., De Wargny, A., Milgram, J., Morel, C., & Prouff, E. (2017). Privacy-preserving classification on deep neural network. Cryptology ePrint Archive.

[8] Wang, W., Zhu, M., Wang, J., Zeng, X., & Yang, Z. (2017, July). End-to-end encrypted traffic classification with one-dimensional convolution neural networks. In 2017 IEEE international conference on intelligence and security informatics (ISI) (pp. 43-48). IEEE.

[9] Cheon, J. H., Kim, A., Kim, M., & Song, Y. (2017). Homomorphic encryption for arithmetic of approximate numbers. In Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I 23 (pp. 409-437). Springer International Publishing.

[10] Chen, H., Han, K., Huang, Z., Jalali, A., & Laine, K. (2017). Simple encrypted arithmetic library v2. 3.0. Microsoft Research, December, 13.

[11] Sun, X., Zhang, P., Liu, J. K., Yu, J., & Xie, W. (2018). Private machine learning classification based on fully homomorphic encryption. IEEE Transactions on Emerging Topics in Computing, 8(2), 352-364.

[12] Minelli, M. (2018). Fully homomorphic encryption for machine learning (Doctoral dissertation, Université Paris sciences et lettres).

[13] Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. ICISSp, 1, 108-116.

[14] Vu, L., Thuy, H. V., Nguyen, Q. U., Ngoc, T. N., Nguyen, D. N., Hoang, D. T., & Dutkiewicz, E. (2018, September). Time series analysis for encrypted traffic classification: A deep learning approach. In 2018 18th International Symposium on Communications and Information Technologies (ISCIT) (pp. 121-126). IEEE.

[15] Rezaei, S., & Liu, X. (2019). Deep learning for encrypted traffic classification: An overview. IEEE communications magazine, 57(5), 76-81.

[16] Alabdulatif, A., Khalil, I., Kumarage, H., Zomaya, A. Y., & Yi, X. (2019). Privacy-preserving anomaly detection in the cloud for quality assured decision-making in smart cities. Journal of Parallel and Distributed Computing, 127, 209-223.

[17] Cheon, J. H., Kim, D., Kim, D., Lee, H. H., & Lee, K. (2019, November). Numerical method for comparison on homomorphically encrypted numbers. In International

Conference on the Theory and Application of Cryptology and Information Security (pp. 415-445). Cham: Springer International Publishing.

[18] Li, J., Kuang, X., Lin, S., Ma, X., & Tang, Y. (2020). Privacy preservation for machine learning training and classification based on homomorphic encryption schemes. Information Sciences, 526, 166-179.

[19] Wood, A., Najarian, K., & Kahrobaei, D. (2020). Homomorphic encryption for machine learning in medicine and bioinformatics. ACM Computing Surveys (CSUR), 53(4), 1-35.

[20] Muliukha, V. A., Laboshin, L. U., Lukashin, A. A., & Nashivochnikov, N. V. (2020, May). Analysis and classification of encrypted network traffic using machine learning. In 2020 XXIII International Conference on Soft Computing and Measurements (SCM) (pp. 194-197). IEEE.

[21] Baldini, G. (2020, June). Analysis of encrypted traffic with time-based features and time frequency analysis. In 2020 Global Internet of Things Summit (GIoTS) (pp. 1-5). IEEE.

[22] Sun, B., Yang, W., Yan, M., Wu, D., Zhu, Y., & Bai, Z. (2020, November). An encrypted traffic classification method combining graph convolutional network and autoencoder. In 2020 IEEE 39th International Performance Computing and Communications Conference (IPCCC) (pp. 1-8). IEEE.

[23] Pulido-Gaytan, L. B., Tchernykh, A., Cortés-Mendoza, J. M., Babenko, M., & Radchenko, G. (2021). A survey on privacy-preserving machine learning with fully homomorphic encryption. In High Performance Computing: 7th Latin American Conference, CARLA 2020, Cuenca, Ecuador, September 2–4, 2020, Revised Selected Papers 7 (pp. 115-129). Springer International Publishing.

[24] Bakhshi, T., & Ghita, B. (2021). Anomaly detection in encrypted internet traffic using hybrid deep learning. Security and Communication Networks, 2021, 1-16.

[25] Benaissa, A., Retiat, B., Cebere, B., & Belfedhal, A. E. (2021). TenSEAL: A library for encrypted tensor operations using homomorphic encryption. arXiv preprint arXiv:2104.03152.

[26] Gao, H., Yuan, L., Yin, F., & Shen, G. (2021, April). Epcad: efficient and privacy-preserving data anomaly detection scheme for industrial control system networks. In Journal of Physics: Conference Series (Vol. 1856, No. 1, p. 012028). IOP Publishing.

[27] Itokazu, K., Wang, L., & Ozawa, S. (2021, July). Outlier Detection by Privacy-Preserving Ensemble Decision Tree U sing Homomorphic Encryption. In 2021 International Joint Conference on Neural Networks (IJCNN) (pp. 1-7). IEEE.

[28] Liu, Q., Peng, Y., Jiang, H., Wu, J., Wang, T., Peng, T., & Wang, G. (2022). SlimBox: Lightweight Packet Inspection Over Encrypted Traffic. IEEE Transactions on Dependable and Secure Computing.

[29] Chen, D., Wang, H., Zhang, N., Nie, X., Dai, H. N., Zhang, K., & Choo, K. K. R. (2022). Privacy-preserving encrypted traffic inspection with symmetric cryptographic techniques in IoT. IEEE Internet of Things Journal, 9(18), 17265-17279.

[30] Adiyodi Madhavan, R., & Sajan, A. Z. (2022). Malicious Activity Detection in Encrypted Network Traffic using A Fully Homomorphic Encryption Method.