Optimized Distributed Gradient Boosting with Explainable Artificial Intelligence for Real-Time Ransomware Detection in Networked Environments

Zeyad Al-Odat
Department of Computer and Communications Engineering, Faculty of Engineering,
Tafila Technical University, Tafila, Jordan
Email: zeyad.alodat@ttu.edu.jo (Z.A.-O.)

Manuscript received February 22, 2025; revised May 8, 2025; accepted May 21, 2025

***Abstract*—The severity of ransomware threats is increasing significantly, posing substantial risks to individuals and organizations. To address this problem, several techniques have been suggested; however, they are mostly signature-based, capable of identifying conventional ransomware but inadequate in detecting novel variants. This paper introduces a ransomware detection framework using distributed gradient boosting and explainable artificial intelligence. The suggested design utilizes the XGBoost algorithm for classification and explainable artificial intelligence for feature importance. The proposed design is deployed and tested using a dataset of 84 features. Weight-based feature selection is used to minimize the number of features. The selected features are used to train the XGBoost classification algorithm. Furthermore, the significance of the chosen features is assessed using SHapley Additive exPlantions (SHAP). The results show better performance in terms of accuracy, precision, recall, and F1-Score. The suggested approach outperforms the baseline machine learning algorithms and comparable results against state-of-the-art models across all performance criteria used.**

***Index Terms*—ransomware, XGBoost, explainable, machine learning, security**

I. Introduction

Recently, ransomware has emerged as one of the most intimidating challenges in cybersecurity. Unlike other threats (malware, spyware, etc.), ransomware is designed to pose an immediate threat, causing significant operational and financial losses [1]. The consecutive attacks on enterprises and individuals have triggered the urgent need for countermeasure mechanisms to maintain sustainability and secure assets. The traditional signature-based ransomware detection approach is effective against known ransomware threats, but there is still inadequacy in detecting modern ransomware. The newly generated ransomware employs sophisticated and complex techniques like polymorphism, encryption, and zero-day attacks [2].

Machine learning algorithms are considered a new approach in detecting ransomware, which relies on training and validation of common patterns. These algorithms provide the ability to detect complex anomaly patterns that are difficult to detect using traditional ransomware methods [3]. The selection of the machine learning method depends on several factors, including types of datasets, machine learning models, and the ability to interpret and visualize the model decisions. A closer look at the datasets used in cybersecurity applications, in particular ransomware, shows that these datasets contain imbalanced sample sizes where anomalous samples represent the minority. This imbalance causes the employed machine learning algorithms, e.g., KNN, LSTM, CNN, NN, decision tree, and random forest, to produce a biased output, resulting in low detection rates for ransomware [4–6].

Moreover, the decisions made by the machine learning algorithms are blind, which makes it difficult to understand the way the algorithms make these decisions. These difficulties represent a major challenge in cybersecurity, which in turn makes machine learning employment in cybersecurity applications limited [7].

The integration of Explainable Artificial Intelligence (XIA) has emerged as a substantial way to explain machine learning decisions. The most famous explanation tools are SHapley Additive exPlantions (SHAP) and Local Interpretable Model-agnostic Explanations (LIME). SHAP is considered a global explanation tool, while LIME is used for local explanations. SHAP is used to explain the model's decisions based on the contribution of each feature to the final decision, additively, and provide a global explanation. However, LIME is used to explain the model based on the prediction of each instance of the output and provides local explanations [8].

This work proposes an enhanced ransomware detection approach using optimized distributed gradient boosting and Explainable Artificial Intelligence (XAI). The XGBoost algorithm provides the ability to manage and train an imbalanced dataset efficiently with high performance. XGBoost creates a series of decision trees sequentially, addressing the errors made by each preceding

doi: 10.18178/ijeetc.14.4.233-242

tree. This sequential process yields a model capable of accurately capturing ransomware patterns within the data [9]. Additionally, the performance of the XGBoost model is improved through the application of the Synthetic Minority Oversampling Technique (SMOTE) to effectively manage imbalanced data [10].

Explainable Artificial Intelligence (XAI) serves to enhance the clarity and compatibility of the decisions generated by the XGBoost model. XIA is utilized to enhance the understanding of the decisions made by XGBoost. In this work, SHapley Additive exPlantions (SHAP) is utilized to interpret and explain the predictions made by XGBoost. In SHAP, each feature is assigned a value that quantifies its contribution to the final decision, referred to as SHAP feature importance. The provided explanations enhance stakeholder trust in the incubation of machine learning algorithms within the field of cybersecurity, enabling them to restructure their systems according to the newly defined models [11].

The contributions of this work are

1) Optimized Distributed Gradient Boosting: an optimized XGBoost model is utilized. The proposed work achieves high detection accuracy while minimizing overfitting. The model is fine-tuned using weight-based feature selection.
2) Explainable Artificial Intelligence: The proposed design integrates SHAP to provide interpretable explanations for the model's predictions.
3) Handling Class Imbalance: The proposed design addresses the class imbalance problem using SMOTE, ensuring that the model is effective in detecting ransomware out of a larger benign dataset.

By the combination of the durability of optimized XGBoost and the interpretability of SHAP, this work provides a robust and secure framework for ransomware detection. The proposed design enhances the detection accuracy of ransomware and improves the trustworthiness between stakeholders and ML algorithms in the field of cybersecurity.

The rest of the paper is organized as follows: Section II presents a literature review of the state of the art in the field of ransomware detection using machine learning. Section III elaborates on the proposed methodology. Section IV provides detailed discussions about the results we achieved from the proposed design. Section V concludes the paper and provides insights into potential future work.

II. Related Work

Many researchers take responsibility for working with ransomware detection, which is noticed obviously from the literature in this area. Generally, the ransomware analysis and detection methods are classified into two main categories: signature-based and behavior-based methods. On the one hand, the signature-based methods are good and effective in analyzing, detecting, and predicting the commonly known ransomware. However, the signature-based method struggles against new variants of ransomware. On the other hand, the behavior-based method focuses on the dynamic behaviors of the system activities. The dynamic behavior provides accurate detection outcomes for new variants of ransomware. However, the behavior-based method is time-consuming and represents a critical issue in preventing payload delivery in the case of a true positive [12].

The ransomware detection methods confront difficulties in analyzing new variants of ransomware. These difficulties include the utilized ransomware with code obfuscation and encryption. Moreover, some methods fail to distinguish between benign and malicious activities due to the false classification of the user and malware-triggered encryptions. Additionally, these approaches are prone to deception by code obfuscation, cryptography, and disassembler protection. In this literature, we provide reviews of the most used topics in this area and their corresponding methods.

A. Machine learning and deep learning

Khammas [13] proposed a ransomware detection approach using the Random Forest machine learning algorithm. The selection of the Random Forest algorithm comes from the fact that it is efficient in handling large datasets. The proposed design employed a feature selection mechanism called frequent pattern mining and gain ratio. This method is used to identify the best 1000 features as top candidates for analysis and detection. The authors claimed that their work outperformed other classifiers such as AdaBoost and Bagging. The reported results showed a high accuracy of 97.74% with low false positive and false negative rates.

Urooj *et al*. [14] have employed machine learning and deep learning algorithms to analyze ransomware behavior during system setup, achieving high detection rates. Similarly, UNVEIL in [15]. UNVEIL autonomously creates a simulated user environment and identifies when ransomware interacts with user data. Specifically, it monitors alterations to the system's desktop that signify ransomware-like activity. However, this work is limited by its inability to recover encrypted files before detection. Other approaches, like CryptoDrop, employed a series of behavioral clues. CryptoDrop may terminate a process that seems to be manipulating a substantial volume of the user's data. Moreover, with the merging of A collection of indications typical of ransomware allows the system to be configured for fast detection with few false positives. Experimental research of CryptoDrop prevents ransomware execution, resulting in a median loss of just 10 files out of around 5,100 accessible files. Results indicate that meticulous examination of ransomware behavior might provide an efficient detection method that substantially reduces victim data loss [16].

Despite these advancements, there remains a need for a more robust detection framework that can effectively identify both known and novel ransomware while minimizing false positives and enabling file recovery during attacks. The work in [17] addresses these gaps by proposing a multilayer detection model that combines grouped registry key operations, file entropy, and file signature monitoring, leveraging machine learning to achieve high detection accuracy and the ability to differentiate between user-triggered and ransomware-triggered encryption. Gulmez *et al*. [18] developed a

strategy to address the ransomware threats by proposing a two-layer convolutional neural network enhanced by Explainable Artificial Intelligence (XAI), termed XRAN. Their work integrates system call features such as API call sequences, DLLs, and mutual exclusions. The method utilizes a dual-layer Convolutional Neural Network (CNN) to identify correlations among these features, attaining a True Positive Rate (TPR) of 99.4%. XRan utilizes two XAI models, namely LIME and SHAP, to provide both local and global explanations for the model's choices. Experimental results show that XRan surpasses leading approaches, especially in differentiating ransomware from inoffensive software and other malware categories.

B. Hybrid and Enhanced Techniques

This category focuses on the methods that combine two or more detection techniques. The resulting hybrid approaches help in optimizing and enhancing efficiency and robustness. In hybrid approaches, static and dynamic analyses broaden detection scope. However, enhanced approaches cultivate the detection process through new algorithms.

Shaukat and Ribeiro [19] presents RansomWall, using both static and dynamic system analysis to provide a feature set that encapsulates the behavior of ransomware. RansomWall incorporates an additional abstract layer that serves as a robust trap for the early detection of ransomware. The suggested methodology is assessed with 574 ransomware samples, attaining a detection rate of 98.25% with almost zero false positives.

Palša *et al.* [20] introduce a malware-detecting antivirus tool using the XGBoost MLMD, which integrates static and dynamic analysis via the XGBoost algorithm. The instrument attains an accuracy of 91.92% on static analysis data and 96.48% on dynamic analysis data. This methodology facilitates the integration of these techniques into practical antimalware applications.

Abutu *et al*. [21] designed a deep learning framework to address the limitations of previous detection techniques by decreasing the false positive and false negative rates. This work concentrates on some important features, such as internet connection, file accessing, and process structures, to increase the precision of the detection process. The proposed work, called DeepCodeLock, improves the detection process accuracy by limiting the false positive and the false negative detection rates. Li *et al*. [22] suggested a new framework that combines the deep learning capabilities with the decision-making ability of Monte Carlo Tree (MCT); merging these together allows the detection of newer variants of ransomware. The proposed work is applicable in the business environment due to the computational efficiency that the new model achieves. The suggested framework enhances the detection process compared to traditional machine learning detection models.

Wasoye *et al*. [23] proposed work integrates machine learning models with the Binary Transformation and Lightweight Signature (BTLS) algorithm to improve the detection process in terms of efficiency and accuracy. The suggested approach can extract both static and dynamic features from the ransomware file, which improves the classification accuracy. The proposed algorithm enhances the detection accuracy by reducing the false positive detection rates.

Adaptive Progressive Feature Isolation (APFI) is proposed to enhance the detection of ransomware attacks; it applies the analysis in separate stages to identify the patterns, especially the high-risk patterns that should be isolated first. The suggested approach improves the accuracy of the detection and reduces the false positive rate. This approach employed deep learning to decrease computational demands. The experiments show that APFI is capable of differentiating ransomware at a highly accurate rate with low latency, which makes this technique an advanced automated solution for cybersecurity [24].

Brinkley *et al*. [25] studied the use of machine learning techniques in detecting zero-day ransomware attacks; this work develops and evaluates some machine learning techniques, such as Random Forst, support vector, and Neural Networks for identifying unseen ransomware behaviors. This study shows that machine learning can adapt to detect the ransomware without using static signatures. The experimental results illustrate the trade-offs between the detection capability and computational efficiency.

Other work presented a comprehensive approach to malware classification using the XGBoost-Gradient Boosted Decision Tree (GBDT) algorithm, leveraging a large, balanced dataset to build an efficient classifier with low-end computing resources [26]. The authors address the limitations of traditional signature-based malware detection methods, which struggle to keep up with the rapid evolution of polymorphic and metamorphic malware. They propose a machine learning-based solution that generalizes new malware variants, emphasizing the importance of feature selection and hyperparameter tuning to optimize model performance. In their work, the authors used a dataset named EMBER. The dataset is balanced and composed of ransomware and benign samples. In the reported results, they achieved 98.5% accuracy. The proposed work is considered robust and efficient due to its limited computational resources and scalability.

Android malware classification using XGBoost is proposed in [27]. The authors highlighted the growing threats of malware and classified them using image-based visualization. The executable binary files were converted to images; these images are in grayscale format, which helps in visualizing the malware and helps in detecting it. The authors compare the performance of XGBoost with other machine learning algorithms, such as k-Nearest Neighbors (k-NN) and Random Forest, and evaluate the effectiveness of using only the data section of the classes.dex file versus the full file. The results indicate that classification using the data section yields better accuracy (70.37%) compared to the full classes.dex file (68.06%), though XGBoost takes longer to train and test compared to other algorithms. The study concludes that while the proposed method shows promise, further improvements are needed to enhance accuracy and reduce computational time, particularly by fine-tuning XGBoost parameters and incorporating more diverse malware samples in future

work.

The work in [28] presented an enhanced Extreme Gradient Boosting (XGBoost) design combined with Chi-Squared Feature Selection. This work helped in finding a new way to detect malware instead of the conventional usage of signature-based methods. The signature-based approach is weak in detecting emerging malware, making it a clear challenge. This challenge has led to increased vulnerabilities in cybersecurity defenses, particularly against zero-day attacks.

C. Network-Based Detection

Network-based detection approaches may discover ransomware activities more promptly by examining communication patterns with Command and Control (C&C) servers. Almashhadani *et al*. [29] concentrate on Locky ransomware, a notable variant recognized for its use of Domain Generation Algorithms (DGA) and hybrid encryption methods. A thorough behavioral study of Locky's network traffic is performed, collecting 18 significant features from protocols including TCP, HTTP, DNS, and NBNS. These traits are categorized into behavioral and non-behavioral, as well as detectable and non-detectable classifications, establishing a good prototype for detection. The suggested multi-classifier intrusion detection system functions at both packet and flow levels, using machine learning methods to attain high detection accuracy (97.92% at the packet level and 97.08% at the flow level) with minimal false positive rates. This study enhances the domain of network-based ransomware detection by presenting a dual-level detection methodology, providing a more efficient solution for the prompt identification and alleviation of ransomware threats.

The work in [30] presented an early detection system for ransomware based on network behavior. The proposed design analyzed the network behavior pattern by employing Cerber ransomware as a case study. The authors addressed the limitations of static network analysis by designing a testbed that dynamically analyzes the system pattern. Their work was deployed on virtual machine environments (VMware and Wireshark). After processing, they extracted 20 features, which were fed to the KNN machine learning algorithm. The reported results show significance in accuracy, achieving 99.5%.

Achieving a balance between computing efficiency and accuracy, reducing false positives, facilitating real-time detection, and recovering encrypted information before detection solve these deficiencies but need more optimization for scalability and comprehensive threat coverage. Future research should emphasize adaptive frameworks that consolidate behavioral, static, and network information while including explainability to provide resilient protection against advancing ransomware threats.

III. Proposed Methodology

This work proposes a ransomware detection mechanism using XGBoost. The proposed design employs SHAP explainable artificial intelligence to decide which features are the most important in detecting ransomware behaviors. The proposed design architecture is depicted in Fig. 1. The design is trained on both benign and ransomware datasets. The datasets are combined to construct the input dataset for the XGBoost classifier. More about the datasets will be elaborated on in the results and discussion section IV.

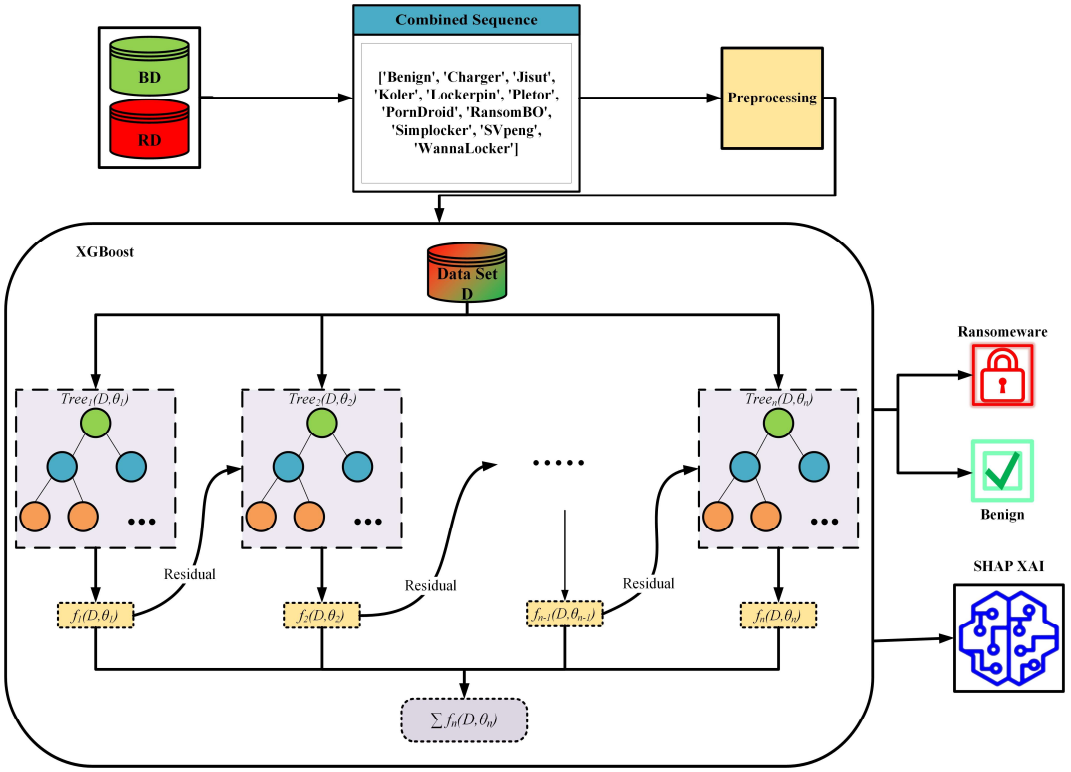

Fig. 1. The proposed design architecture and workflow.

A. XGBoost Classification

To classify the data into benign or ransomware, the XGBoost classifier is used as an extreme gradient boosting algorithm to classify the input data. The XGBoost is a powerful and scalable framework for supervised machine learning applications. The selection of the XGBoost comes from the fact that it provides high performance and speed. The XGBoost's main principle is gradient boosting, where the model is built sequentially to fix and correct the errors from the preceding step. Each step in the XGBoost is represented as a sequence of trees. The number of trees is determined in the hyperparameter configuration step for the classifier. Every step a new tree is added to minimize the residual error from the previous tree. The final decision of the XGBoost is the sum of all predictions from all trees, as depicted in Eq. (1).

$$\widehat{y_i} = \sum_{t=1}^{T} f_t(x_i) \tag{1}$$

where f_t: the prediction of tree i, x_i: the input data points, and T: total number of trees.

Each tree belongs to space $\mathbb{F}$ represented as

$$\mathbb{F} = f(x) = w\_\{q(x)\} \mid q: R^m \to N, w \in R^T \tag{2}$$

where $q(x)$: the function that maps the data into the leaf index, and w: represents the leaf weight.

For the proposed design, we used the *binary:logistic* objective function. The selection of this objective function is due to the binary nature of the classification task that we have (Benign, Ransomware). For a single sample, the logistic error is calculated using Eq. (3):

$$L(y_i, \overline{y_i}) = -[y_i \log(\overline{y_i}) + (1 - y_i) \log(1 - \overline{y_i})] \tag{3}$$

The total loss for N samples is calculated by

$$L = -\frac{1}{N} \sum_{i=1}^{N} [y_i \log(\overline{y_i}) + (1 - y_i) \log(1 - \overline{y_i})] \tag{4}$$

B. Explainable Artificial Intelligence

Explainable Artificial Intelligence (XAI) is used to interpret the machine learning prediction algorithm. This interpretation is accomplished by analyzing the input dataset and the features that were used by the machine learning algorithm. In most cases, the machine learning prediction approach is considered a black box, where there is no knowledge about the way the algorithm has chosen certain output or the most influential features in the dataset. XAI is used to provide the user with the most important features that affect the system selection.

We used SHapley Additive exPlanation (SHAP) XIA explainable model in our design. SHAP is employed to accomplish the task of categorizing the features according to their importance and then selecting the most important features for training and classification tasks. The SHAP equation is formulated as

$$\text{SHAP}_i = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!} [v(S \cup \{i\}) - v(S)] \tag{5}$$

where N represents a variable for all features, S is a subset of N without i, $v(S)$ represents the model value for the S features, and $|S|!$ the factorial of the size of S.

C. Performance Metrics

Since the proposed design deals with a binary classification task, we used the following performance metrics to assess our design:

1) Accuracy: which represents the ratio of the correctly classified (Benign or Ransomware) to the total number of samples. The accuracy is represented by

$$\text{Accuracy} = \frac{T^+ + T^-}{T^+ + T^- + F^+ + F^-} \tag{6}$$

2) Precision, which represents the positive predictive values as depicted in Eq. (7):

$$\text{Precision} = \frac{T^+}{T^+ + F^+} \tag{7}$$

3) Recall that represents the correctly predicted positive to all positive samples, as depicted in Eq. (8):

$$\text{Recall} = \frac{T^+}{T^+ + F^-} \tag{8}$$

4) F1-score that makes the harmonic balance between the precision and recall, as depicted in Eq. (9):

$$F1\text{-}Score = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{9}$$

D. Feature Importance

To select the most important features in our work, we employed weight-based feature importance. This selection reduces the number of features used in the training and reduces the training time accordingly.

For XGBoost, the weight of the feature is the number of times that the data across all XGBoost trees were split; features that are used frequently for splitting are considered the most important. This calculation is calculated by Eq. (10).

$$\text{Importance}(f) = S_f \tag{10}$$

where F is the number of features and S_f is the number of splits in trees for feature $f \in F$.

The higher the value of S_f implies that the feature f is used more frequently.

IV. Results and Discussion

A. Experimental Setup

The experiments were conducted on a PC with an Intel Core i7 processor, 32GB of Random Access Memory (RAM), and an NVIDIA RTX 1070ti Graphical Processing Unit (GPU). These configurations are set to make sure that the machine learning computations will be executed efficiently. The deployment of the software was built on Python 3 using scikit-learn for the machine learning algorithm.

In this work, ransomware and benign datasets were used. These datasets were adopted from Canadian Institute for Cybersecurity Android Malware 2017 (CICAndMal2017) [31]. The properties of this dataset are as follows:

1) The dataset contains APK files of malicious and benign applications collected from various sources. The malicious application samples are categorized

into the Adware, Spyware, Ransomware, and Banking Trojan families.
2) The ransomware family includes 10 types: LockerPin, Simplocker, Pletor, PornDroid, RansomBO, SvpengCharger, Jisut, Koler, and WannaLocker.
3) The total number of samples from the Ransomware family is 309,084 samples.
4) The total number of samples from the Benign family is 1,048,575.
5) Each type of dataset contains 84 features. Table I shows all features included in the dataset and their corresponding descriptions.
6) For the full details of the dataset, the reader is advised to follow up with the dataset's author through the work presented in [32].

TABLE I: FEATURE DESCRIPTION

Feature Name	Description	Feature Name	Description
Flow ID	A unique ID assigned to each network flow	Bwd IAT Std	The standard deviation of time intervals between backward packets
Source IP	The IP address where the flow originates	Bwd IAT Max	The longest time interval between backward packets
Source Port	The port number from which the flow starts	Bwd IAT Min	The shortest time interval between backward packets
Destination IP	The IP address where the flow is directed	Fwd PSH Flags	The number of PSH flags in the forward direction
Destination Port	The port number to which the flow is sent	Bwd PSH Flags	The number of PSH flags in the backward direction
Protocol	The protocol used in the flow (e.g., TCP, UDP)	Fwd URG Flags	The number of URG flags in the forward direction
Timestamp	The time at which the flow was recorded	Bwd URG Flags	The number of URG flags in the backward direction
Flow Duration	The total time duration of the flow	Fwd Header Length	The size of the header in forward packets
Total Pwd Packets	The total number of packets sent forward	Bwd Header Length	The size of the header in backward packets
Total Backward Packets	The total number of packets sent backward	Fwd Packets/s	The rate of forward packets per second
Total Length of Fwd Packets	The combined size of all forward packets	Bwd Packets/s	The rate of backward packets per second
Total Length of Bwd Packets	The combined size of all backward packets	Min Packet Length	The smallest size among all packets
Fwd Packet Length Max	The largest size of a forward packet	Max Packet Length	The largest size among all packets
Fwd Packet Length Min	The smallest size of a forward packet	Packet Length Mean	The average size of all packets
Fwd Packet Length Mean	The average size of forward packets	Packet Length Std	The standard deviation of packet sizes
Fwd Packet Length Std	The standard deviation of forward packet sizes	Packet Length Variance	The variance in packet sizes
Bwd Packet Length Max	The largest size of a backward packet	FIN Flag Count	The total number of FIN flags observed
Bwd Packet Length Min	The smallest size of a backward packet	SYN Flag Count	The total number of SYN flags observed
Bwd Packet Length Mean	The average size of backward packets	RST Flag Count	The total number of RST flags observed
Bwd Packet Length Std	The standard deviation of backward packet sizes	PSH Flag Count	The total number of PSH flags observed
Flow Bytes/s	The rate of bytes transferred per second	ACK Flag Count	The total number of ACK flags observed
Flow Packets/s	The rate of packets transferred per second	URG Flag Count	The total number of URG flags observed
Flow IAT Mean	The average time between packets in the flow	CWE Flag Count	The total number of CWE flags observed
Flow IAT Std	The standard deviation of time intervals between packets	ECE Flag Count	The total number of ECE flags observed
Flow IAT Max	The longest time interval between packets	DownUp Ratio	The ratio of downtime traffic to upload traffic
Flow IAT Min	The shortest time interval between packets	Average Packet Size	The mean size of all packets
Fwd IAT Total	The total time intervals between forward packets	Avg Fwd Segment Size	The average size of forward segments
Fwd IAT Mean	The average time interval between forward packets	Avg Bwd Segment Size	The average size of backward segments
Fwd IAT Std	The standard deviation of time intervals between forward packets	Fwd Header Length_1	An alternative measure of forward header size
Fwd IAT Max	The longest time interval between forward packets	Fwd Avg Bytes/Bulk	The average bytes per bulk in the forward direction
Fwd IAT Min	The shortest time interval between forward packets	Fwd Avg Packets/Bulk	The average packets per bulk in the forward direction
Bwd IAT Total	The total time intervals between backward packets	Fwd Avg Bulk Rate	The average bulk rate in the forward direction
Bwd IAT Mean	The average time interval between backward packets	Bwd Avg Bytes/Bulk	The average bytes per bulk in the backward direction

In our work, we selected the Benign and Ransomware datasets to conduct the experiments. The proposed design was trained on a combined dataset generated by combining the Benign and Ransomware datasets. The dataset comprises 84 features, which increase the complexity and computation time to train and validate the dataset. This work uses weight-based feature selection to train the network on the top 20 features based on their weight. This weight represents the frequency of the features in splitting the XGBoost trees. The top 20 features after this step are depicted in Table II.

TABLE II: TOP 20 FEATURES

Destination Port	Flow Duration	Source Port
Fwd Packet Length Max	Flow IAT Mean	Flow IAT Std
Flow Packets/s	Bwd Packet Length Min	Bwd Packet Length Mean
Total Fwd Packets	Flow Bytes/s	Total Length of Fwd Packets
Total Backward Packets	Fwd Packet Length Mean	Bwd Packet Length Max
Bwd Packet Length Std	Total Length of Bwd Packets	Fwd Packet Length Std
Fwd Packet Length Min	Protocol	

The features 'Flow ID', 'Timestamp', 'Source IP', and 'Destination IP' are dropped from training because they lack predictive power, affect overfitting to the system, and are non-informative.

B. Performance Evaluation

The proposed design is evaluated using the XGBoost training on the combined dataset. After training, the top twenty features of the dataset were selected. These features were fed to the XGBoost for the second round of training. The training and validation loss performance is depicted in Fig. 2. The training loss started high and then decreased rapidly, implying that it is learning effectively from the training dataset. Moreover, as the number of boosting rounds increases, the training loss decreases, approaching zero, deducing the training data well. On the other hand, the validation loss started high and decreased at the early stages of the boosting rounds, which implies that the model generalizes well to the unseen data. However, after rounds 30 to 40, the validation loss begins to stabilize, which shows an indication of data overfitting. To solve this issue, we employed the Synthetic Minority Oversampling Technique (SMOTE), which addresses the class imbalance.

To mitigate overfitting, we incorporated multiple regularization techniques into the XGBoost model. Table III summarizes the regularization techniques that are used in the XGBoost model. The regularization techniques improve generalization by constraining the model complexity. The L1 and L2 regularization techniques were used with reg_alpha (0.5–1.0) and reg_lambda (2.0–3.0), respectively. L1 helps in shrinking the less important feature values to zero, while L2 is used to discourage the large weights. The tree complexity controls the maximum depth that child trees will split, limiting the depth of the tree.

TABLE III: REGULARIZATION TECHNIQUES USED IN XGBOOST MODEL

Technique	Parameter	Value	Role
L2 Regularization	reg_lambda	2.0–3.0	Penalizes large weights.
L1 Regularization	reg_alpha	0.5–1.0	shrinking less important features to zero.
Tree Complexity	max_depth	-	Limits tree depth

The proposed design is compared with the baseline machine learning algorithms. The results of this comparison are depicted in Table IV.

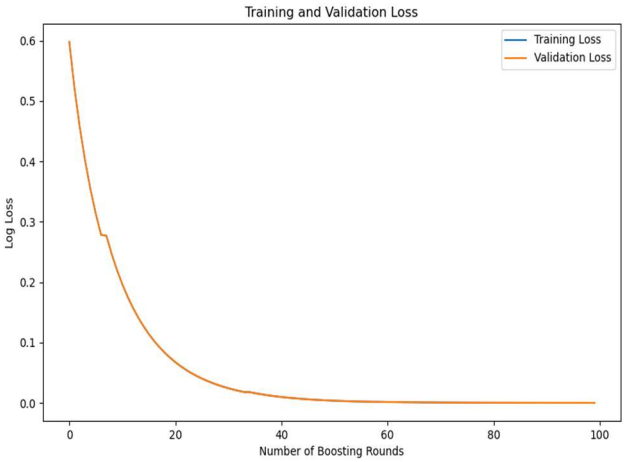

Fig. 2. Training and validation loss.

TABLE IV: COMPARISON OF BASELINE METHODS WITH THE PROPOSED DESIGN

Detection Method	**Precision**	**Accuracy**	**Recall (TPR)**	**F1-Score**	**FPR**	**TNR**	**FNR**	**TPR**
Decision Tree	0.901	0.887	0.909	0.893	0.123	0.877	0.091	0.909
Random Forest	0.937	0.928	0.944	0.931	0.081	0.919	0.056	0.944
Naive Bayes	0.575	0.611	0.556	0.596	0.358	0.642	0.444	0.556
KNN (k=5)	0.849	0.786	0.89	0.811	0.255	0.745	0.11	0.89
LSTM	0.964	0.952	0.981	0.955	0.07	0.93	0.019	0.981
CNN	0.962	0.955	0.977	0.958	0.061	0.939	0.023	0.977
Proposed Design	0.993	0.983	0.999	0.999	0.00615	0.99385	0.001	0.999

TABLE V: COMPARISON BETWEEN THE PROPOSED DESIGN AND THE STATE OF THE ART WORKS

Work	**Accuracy**	**Recall**	**F1-Score**	**Precision**	**TPR**	**FPR**	**TNR**	**FNR**
[13]	0.9774	0.998	0.978	N.A	0.998	0.043	0.957	0.002
[18]	0.99	0.994	0.992	0.992	0.994	0.01	0.99	0.006
[17]	0.987	0.995	0.996	0.999	0.991	0.036	0.0039	0.009
[19]	N.A	0.99	0.99	0.987	0.9825	0.056	0.944	0.017
[20]	0.964	N.A	0.977	N.A	N.A	N.A	N.A	N.A
Proposed Design	0.993	0.983	0.999	0.999	0.99385	0.00615	0.001	0.999

Table IV shows that the proposed design outperformed the baseline machine learning algorithms regarding accuracy, precision, F1-score, and recall. The results show that the detection rate of our proposal achieved 99.3\% with higher accuracy. The metrics of the deep learning algorithms (CNN, LSTM) show better results than the traditional machine learning algorithms. This difference comes from the conventional ML algorithms that individually evaluate systems' features.

A comparison between the proposed design and the state-of-the-art methods is depicted in Table V. The comparison shows that the proposed method achieves better results compared with the same techniques. Some works reported only a portion of the metrics we used; therefore, we used the equations mentioned earlier in Section III to calculate the corresponding values.

The proposed design achieves 99.3% accuracy, which reflects alongside precision (99.9%) the effectiveness of the proposed design in detecting ransomware behaviors. Some works didn't report all the values used in the performance metrics; therefore, we measured their corresponding values using the scientific equations concerning each value. However, the work in [20] reported only the accuracy value, which is inadequate in measuring the remaining performance value. Moreover, the work in [18] measured the metrics values concerning different scenarios; the corresponding result for this work is calculated by averaging them.

However, the low false positive rate (FPR) can have great consequences in real-world deployments. This will dramatically affect different aspects including alert fatigue, operational distribution, user trust and economic cost. The real impact of FPR basically depends on traffic volume. For instance, a daily traffic of million requests achieves 6150 false positive reading per day. In contrast, 6150 false alerts overwhelm the system troubleshooting and maintenance tasks.

C. Feature Selection Sensitivity Analysis

A sensitivity analysis for feature selection was performed to identify the most significant features. Table VI illustrates the outcomes of the sensitivity analysis. The research extended through features, starting with the top 5 and extending to the top 50. Metrics of Accuracy, Precision, Recall, and F1-Score were employed to evaluate the performance of the features. The measured findings indicate that the performance metrics stabilized at 20 features. In our proposal, we selected the top 20 elements as a key benchmark in design.

TABLE VI: FEATURE SELECTION SENSITIVITY ANALYSIS

No. Features	Accuracy	Precision	Recall	F1-Score
5	0.94	0.94	0.94	0.94
10	0.96	0.96	0.96	0.96
15	0.97	0.97	0.96	0.96
20	0.99	0.99	0.98	0.99
25	0.99	0.98	0.98	0.98
30	0.99	0.99	0.98	0.98
40	0.99	0.99	0.99	0.99
50	0.99	0.99	0.98	0.99

Fig 3 illustrates the correlation between the chosen performance indicators and the quantity of top features. Commencing with the top 20 features, the performance measures start to stabilize. The selection of the top 20 features in our methodology is predicated on the observation that augmenting the number of features requires more resources and computing time. Consequently, the proposed architecture preserves the minimum number of features to guarantee optimal performance and minimal resource requirements.

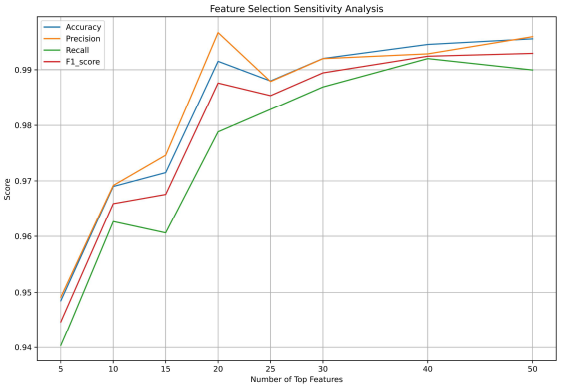

Fig. 3. Feature selection sensitivity analysis.

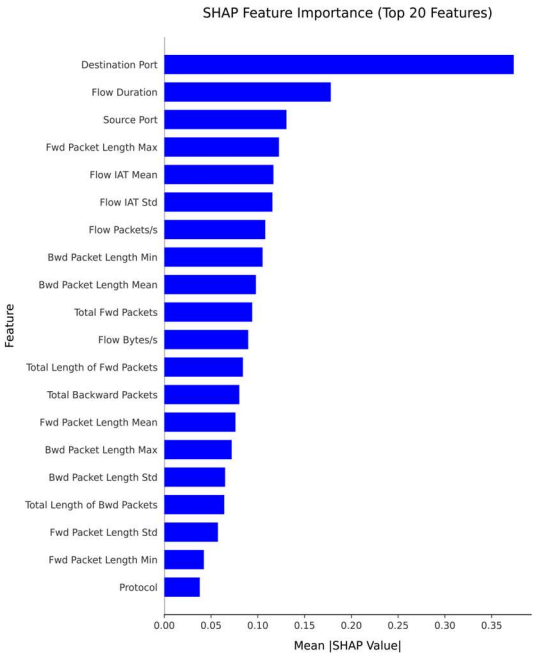

Fig. 4. SHAP features importance for the top 20 features.

D. Feature Explanation

In the list of selected twenty features, SHAP was used to distinguish between them according to importance. Fig. 4 shows the importance of the feature in detecting ransomware. The importance of feature analysis is summarized as follows:

- Flow Duration, Flow IAT Mean, and Flow Packets\/s describe the temporal behavior of the traffic.
- Total Fwd Packets, Total Length of Fwd Packets, and Flow Bytes/s quantify the amount of data being transferred.
- Fwd Packet Length Max, Bwd Packet Length Mean, and Fwd Packet Length Std describe the size and

variability of packets.

- Protocol, Destination Port, and Source Port provide context about the type of traffic and services being used.

The SHAP summary plot, as depicted in Fig. 5, shows the SHAP importance values for the top 20 features. These features represent the most influential network traffic feature in detecting potential threats. To better understand the figure, follow the following points:

1) The features are ranked from top to bottom according to their importance.
2) The *x*-axis represents the SHAP value. Positive values push the prediction toward malicious classification. Negative values push the prediction toward benign classification.
3) The color of the dots represents the actual value of the feature in the data. Red dots represent high values, and blue dots represent low values.
4) Features with horizontal spread imply greater impact on prediction. While the vertical spread for features with overlapping dots have less impact on predictions.

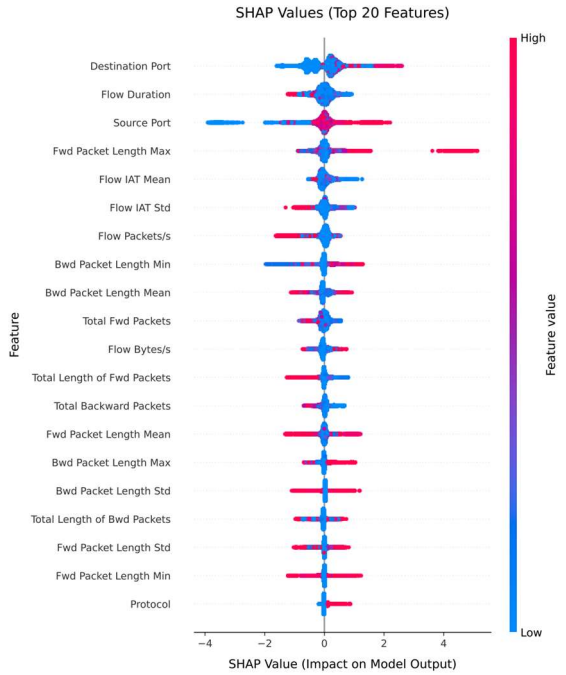

Fig. 5. SHAP summary plot for the top 20 features.

This summary is used to elucidate the influence of each attribute on the model's predictions. The y-axis illustrates the top 20 characteristics ranked in decreasing order of significance, with each feature shown by a horizontal line. The x-axis denotes the SHAP value, indicating the influence of the characteristics on the model's output. SHAP values may be either positive or negative; positive values augment the model's output, while negative values diminish it. Each characteristic's value is shown by red and blue dots, with red indicating high feature value and blue indicating low feature value. The chart indicates that the Destination Port exerts the most influence, whilst the Protocol feature demonstrates the least effect among the top 20 in the prediction model.

The data shown by Fig. 5 is essential for comprehending the model's behavior and facilitating educated decision-making based on its predictions. For example, Destination Port, Flow Duration, and Source Port exhibit a broad spectrum of SHAP values (ranging from negative to positive), substantially influencing the forecasts. Nonetheless, Protocol exhibits a reduced range of SHAP values, indicating a diminished impact on the predictions. Furthermore, the Fwd Packet Length Max (red dots) indicates that greater packet sizes correlate with a certain class (Ransomware/Benign). Furthermore, elevated SHAP values for certain destination ports suggest that traffic to these ports is associated with malicious behavior, whilst the substantial packet size of the 'Fwd Packet Length Max' characteristic indicates that the traffic is malicious.

The proposed design can be integrated into security infrastructure as a supplemental detection layer. The deployment comprises the following applications:

1) The deployment as a plugin to analyze network logs. This is achieved by flagging suspicious traffic patterns.
2) Embed as a decision support model in firewalls, intrusion detection systems and intrusion prevention systems. Consequently, blocking traffic is classified as malicious.
3) Can be used as a trigger for automated workflows in platforms as an API endpoint. Consequently, automatic quarantine of exposed devices.
4) Used in Endpoint Detection and Response (EDR) solutions. Sharing threat scores with software defenders to achieve correlation between them. This will help in identifying Advance Persistent Threats (APT).

V. Conclusion

This work presented an explainable and optimized ransomware detection approach using extreme gradient boosting algorithm (XGBoost) and Explainable Artificial Intelligence (XIA). The proposed design is optimized using a Synthetic Minority Oversampling Technique (SMOTE) and weight-based feature selection. SMOTE is used to address the data imbalance of the dataset, while weight-based is used to select the most important feature to train the XGBoost classifier. According to the result, the proposed design outperformed other state-of-the-art in terms of accuracy, precision, F1-Score, and recall, achieving 99.3%, 99.9%, 99.9%, and 98.3%, respectively. The selected features were explained according to their importance in making the system decision. This explanation is performed using the global explanation algorithm SHAP, which helped in understanding the system decisions and which features affected the most in the decisions. The key limitations that confront the proposed design are comprised as technical constraints, operational challenges and generalization gaps. The technical constraints represent a focal point regarding the labeled data dependency which is expensive to acquire attack patterns, consequently, inference computational overhead and resource managements. The operational

challenges are represented by the integration of the system with legacy systems, that require middleware infrastructure. The generalization gaps limit the ability to interpret and analyze modern networks that use TLS/SSL encryption.

In the future, more experiments will be conducted on different datasets of different sizes. Moreover, applying the same algorithm to identify the type of ransomware, which represents a challenging task. An adaptive learning framework is a potential which can be deployed online to add new traffic patterns. Furthermore, build the explainable model with trust frameworks by extending the SHAP interpretation with natural language explanations.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

REFERENCES

[1] M. Conti, S. Khandhar, and P. Vinod, "A few-shot malware classification approach for unknown family recognition using malware feature visualization," *Comput. \& Secur.*, vol. 122, 102887, 2022.

[2] A. Kapoor, A. Gupta, R. Gupta, *et al.*, "Ransomware detection, avoidance, and mitigation scheme: A review and future directions," *Sustainability*, vol. 14, no. 1, p. 8, 2021.

[3] J. F. Abukhait and M. S. Saleh, "An adaptive confidentiality security service enhancement protocol using image-based key generator for multi-agent ethernet packet switched networks," *Int. J. Electr. Electron. Eng. \& Telecommun.*, vol. 12, no. 2, pp. 112–123, 2023.

[4] R. Brewer, "Ransomware attacks: detection, prevention and cure," *Netw. Secur.*, vol. 2016, no. 9, pp. 5–9, 2016.

[5] P. O'Kane, S. Sezer, and D. Carlin, "Evolution of ransomware," *Iet Networks*, vol. 7, no. 5, pp. 321–327, 2018.

[6] H. Oz, A. Aris, A. Levi *et al.*, "A Survey on Ransomware: Evolution, Taxonomy, and Defense Solutions," *arXiv e-prints*, p. arXiv--2102, 2021.

[7] H. Oz, A. Aris, A. Levi *et al.*, "A survey on ransomware: Evolution, taxonomy, and defense solutions," *ACM Comput. Surv.*, vol. 54, no. 11s, pp. 1–37, 2022.

[8] D. Minh, H. X. Wang, Y. F. Li *et al.*, "Explainable artificial intelligence: a comprehensive review," *Artif. Intell. Rev.*, pp. 1–66, 2022.

[9] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA: Association for Computing Machinery, 2016, pp. 785–794. doi: 10.1145/2939672.2939785

[10] L. Camacho, G. Douzas, and F. Bacao, "Geometric SMOTE for regression," *Expert Syst. Appl.*, vol. 193, 116387, 2022.

[11] V. Vimbi, N. Shaffi, and M. Mahmud, "Interpreting artificial intelligence models: a systematic review on the application of LIME and SHAP in Alzheimer's disease detection," *Brain Informatics*, vol. 11, no. 1, p. 10, 2024.

[12] C. C. Moreira, D. C. Moreira, and C. de S. de Sales Jr, "Improving ransomware detection based on portable executable header using xception convolutional neural network," *Comput. \& Secur.*, vol. 130, 103265, 2023.

[13] B. M. Khammas, "Ransomware detection using random forest technique," *ICT Express*, vol. 6, no. 4, pp. 325–331, 2020.

[14] U. Urooj, B. A. S. Al-rimy, A. Zainal *et al.*, "Ransomware detection using the dynamic analysis and machine learning: A survey and research directions," *Appl. Sci.*, vol. 12, no. 1, p. 172, 2021.

[15] A. Kharaz, S. Arshad, C. Mulliner *et al.*, "UNVEIL: A large-scale, automated approach to detecting ransomware," in *Proc. 25th USENIX security symposium (USENIX Security 16)*, 2016, pp. 757–772.

[16] N. Scaife, H. Carter, P. Traynor *et al.*, "CryptoLock (and drop it): Stopping ransomware attacks on user data," in *Proc. 2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*, 2016, pp. 303–312. doi: 10.1109/ICDCS.2016.46

[17] B. Jethva, I. Traoré, A. Ghaleb *et al.*, "Multilayer ransomware detection using grouped registry key operations, file entropy and file signature monitoring," *J. Comput. Secur.*, vol. 28, no. 3, pp. 337–373, 2020.

[18] S. Gulmez, A. G. Kakisim, and I. Sogukpinar, "XRan: Explainable deep learning-based ransomware detection using dynamic analysis," *Comput. \& Secur.*, vol. 139, 103703, 2024.

[19] S. K. Shaukat and V. J. Ribeiro, "RansomWall: A layered defense system against cryptographic ransomware attacks using machine learning," in *Proc. 2018 10th International Conference on Communication Systems \& Networks (COMSNETS)*, 2018, pp. 356–363. doi: 10.1109/COMSNETS.2018.8328219

[20] J. Palša, N. Ádám, J. Hurtuk *et al.*, "MLMD—A malware-detecting antivirus tool based on the XGBoost machine learning algorithm," *Appl. Sci.*, vol. 12, no. 13, 2022. doi: 10.3390/app12136672

[21] G. Abutu, D. Weissman, P. Hoffmann *et al.*, "Deepcodelock: A novel deep learning-based approach for automated ransomware detection using behavioral signatures," *Authorea Prepr.*, 2024.

[22] G. Li, S. Wang, Y. Chen *et al.*, *A Hybrid Framework for Ransomware Detection Using Deep Learning and Monte Carlo Tree Search*, 2024. doi: 10.31219/osf.io/cjyvb

[23] S. Wasoye, M. Stevens, C. Morgan *et al.*, *Ransomware Classification Using Btls Algorithm and Machine Learning Approaches*, 2024.

[24] P. Sladkova, S. Berger, Z. Skoglund *et al.*, *Adaptive Deep Learning-Based Framework for Ransomware Detection through Progressive Feature Isolation*, 2024. doi: 10.31219/osf.io/7bxg2

[25] Y. Brinkley, D. Thompson, and N. Simmons, *Machine Learning-Based Intrusion Detection for Zero-Day Ransomware in Unseen Data*, 2024. doi: 10.22541/au.172685266.62026194/v1

[26] R. Kumar and G. S, "Malware classification using XGboost-Gradient Boosted Decision Tree," *Adv. Sci. Technol. Eng. Syst. J.*, vol. 5, pp. 536–549, 2020.

[27] F. M. Darus, N. A. Ahmad, and A. F. M. Ariffin, "Android malware classification using XGBoost on data image pattern," in *Proc. 2019 IEEE International Conference on Internet of Things and Intelligence System (IoTaIS)*, 2019, pp. 118–122.

[28] S. Rosyada, F. A. Rafrastara, A. Ramadhani *et al.*, "Enhancing XGBoost performance in malware detection through chi-squared feature selection," *J. Sisfokom (Sistem Inf. dan Komputer)*, vol. 13, no. 3, pp. 396–402, 2024.

[29] A. O. Almashhadani, M. Kaiiali, S. Sezer, *et al.*, "A multi-classifier network-based crypto ransomware detection system: A case study of locky ransomware," *IEEE access*, vol. 7, pp. 47053–47067, 2019.

[30] H. Abu-Helo and H. Ashqar, "Early Ransomware Detection System Based on Network Behavior," in *Proc. International Conference on Advanced Information Networking and Applications*, 2024, pp. 447–458.

[31] Android Adware 2017 Datasets Research Canadian Institute for Cybersecurity UNB. (Feb. 2024). [Online]. Available: https://www.unb.ca/cic/datasets/android-adware.html

[32] A. H. Lashkari, A. F. A. Kadir, H. Gonzalez, *et al.*, "Towards a Network-Based Framework for Android Malware Detection and Characterization," in Proc. *2017 15th Annual Conference on Privacy, Security and Trust (PST)*, 2017, pp. 233–23309. doi: 10.1109/PST.2017.00035

Zeyad A. Al-Odat was born in 1986 in Jordan. He received his B.Sc. degree in computer engineering from Mutah University, Karak, Jordan in 2009, the M.Sc. degree in computer engineering from Yarmouth University, Irbid, Jordan in 2013, and the Ph.D. degree in computer engineering from North Dakota State University, Fargo, USA in 2020. His work experience includes cryptography, cybersecurity, IoT, big data, and high-performance computing. Currently he works as an assistant professor in the Department of Computer and Communications Engineering at Tafila Technical University, Tafila, Jordan.