4.6 Discussion

To select an optimal feature subset from a given dataset, a filter-based supervised feature selection method based on rough set theory is proposed. This method identifies the best feature subset by evaluating the relevance and significance of each feature. By leveraging rough set theory and introducing a novel criterion, the approach effectively pinpoints the most relevant features. The method is tested on ten real-world datasets to evaluate its performance. The accuracy of the feature selection method (FSR) is compared to seven other existing methods, consistently showing superior results. Additionally, the method is specifically evaluated on the TUANROMD and XSSD malware and malware attack datasets, where it also shows superior performance. This highlights the method's robustness and effectiveness in handling complex and diverse data, making it a valuable tool for feature selection in various applications. While the method has shown promising results, its applicability to other high-dimensional domains remains unexplored. As a potential extension, future work could involve applying the FSR method to ultra-high-dimensional transcriptomic data. This would involve addressing the unique challenges posed by the high dimensionality and complexity of such data.

Chapter 5

A Cost-Effective Method for Ransomware Detection

5.1 Introduction

With rapid advances in technology, more than one third of the world's population has now entered the digital world. The Internet provides the backbone to the digital world where people constantly make use of beneficial services and applications available on the Internet. The Internet is used for basic communication purposes as well as for numerous online transactions. The services available on the Internet can be exploited by people with destructive intentions. Malicious software or Malware is used to further increase the harmful intentions of such people. There are various types of malware available in the wild and each of them has been designed for specific purpose. In recent years, ransomware has emerged as a new malware epidemic that creates havoc on the Internet. The first known ransomware "AIDS Trojan[1]" appeared in 1989. It infiltrates a victim system or network and encrypts all personal files or the whole system using a variety of encryption techniques. Such techniques prevent users from ac- cessing files or the system until the required amount of ransom is paid.

Locker and *Crypto* are primarily the two categories of ransomware. Both kinds of ransomware utilize the same infection vectors like drive-by download, social engineering, phishing, spam emails, or removable media to get into the information devices and systems, including mobile and Internet of Things devices. However, the way of compromising the victim's system is different for both types

[1]https://spideroak.com/ransomware/timeline

of ransomware. The *locker* ransomware is designed to lock the target system and denies user access to the system without making any modification to the file system and then a certain amount of ransom is demanded from the victim. On the other hand, *crypto* ransomware, after getting into the victim's system, encrypts all or selected files in the system using various encryption techniques like AES or RSA [61]. After encryption, a message with all the payment instructions is displayed to the users. To gain access to the system, a required ransom need to be paid to the attackers and in return a decryption key is obtained. For the ransom, digital currencies such as Ukash, cryptocurrency are used which is very difficult to trace. The cyber-attacks carried out by ransomware is growing and becoming more sophisticated to defend against. With the development ransomware malware automated creation tools, the ransomware is uploaded heavily on the internet. The malware creation also helps the inexperienced users to create their own malware and to carried out an attack.

5.1.1 Motivation

In recent years, the proliferation of ransomware attacks has posed a significant threat to individuals, businesses, and organizations worldwide. The rapidly evolving nature of these malicious activities demands the development of advanced and cost-effective methods for timely detection and mitigation. The complexity of ransomware attacks necessitates a multi-faceted approach to detection. Employing ensemble techniques that combine the strengths of various learning algorithms, can enhance the overall efficacy of ransomware detection. However, the development and implementation of such ensemble methods pose challenges related to resource allocation, algorithm integration, and real-time adaptability.

5.1.2 Contribution

This chapter presents a fast, yet reliable ransomware defense solution, referred to as ERAND, powered by an optimal feature selection method to discriminate the ransomware class as a whole, as well as the eleven variants of the ransomware family from the goodware instances.

The proposed solution is significant, considering the following clauses.

1. It is able to identify an optimal subset of Indicator of Compromises (IoCs)

for the ransomware as a family as well as its individual variants to ensure better accuracy.

2. It is able to classify or discriminate instances of ransomware class (as a whole) from non-ransomware as well as instances corresponding to individual variants from one another with high accuracy.

5.2 Background

Ransomware is a type of malicious software designed to block access to a computer system or files, usually by encrypting them, until a sum of money is paid to the attacker. The term is a portmanteau of "ransom" and "software," reflecting the extortionate nature of the attack [62][63]. Ransomware attacks typically follow a series of well-defined phases, each contributing to the success of the overall attack. The attack phases of ransomware can be summarized as follows.

1. Reconnaissance: The first phase of a ransomware attack involves reconnaissance and target selection. This phase is crucial for attackers to identify vulnerabilities, weaknesses, and suitable entry points into a target's systems. Attackers may leverage various techniques, such as open-source intelligence gathering, social engineering, and scanning for publicly available information. The goal is to acquire a comprehensive understanding of the target's network architecture and personnel, allowing the attackers to craft tailored and effective delivery methods in subsequent stages.

2. Delivery: Once a target has been selected, the attackers will attempt to gain initial access to their network or systems. Malicious payloads are delivered to the target systems, often through phishing emails, malicious websites, or infected attachments. Social engineering plays a pivotal role as attackers craft convincing messages to trick unsuspecting users into opening or downloading the malware-laden content. Once the victim interacts with the malicious payload, the ransomware gains a foothold within the system.

3. Exploitation: Once the malicious payload is delivered and a user interacts with it, the attackers exploit weaknesses in the system's software, applications, or configurations. This exploitation allows them to circumvent security measures and infiltrate the target network.

4. Installation: In the installation phase of a ransomware attack, the malicious code delivered during the previous phases is executed on the compromised

system. This phase involves the ransomware establishing a presence on the victim's machine, often by copying its files, creating registry entries, or employing other techniques to ensure persistence.

5. Command and Control (C2): The ransomware establishes communication with a remote server controlled by the attackers. This phase is crucial for maintaining control over the infected systems and initiating actions such as data encryption, exfiltration, or further exploitation.

6. Encryption: Encryption is a pivotal phase in a ransomware attack where the malware encrypts files on the compromised system and, in some cases, connected network drives. The encryption process renders the victim's data inaccessible without the corresponding decryption key, which is held by the attackers.

7. Extortion: Finally, after successfully encrypting files, attackers present victims with a ransom note that outlines the terms for the release of the decryption key. The note typically demands payment, often in cryptocurrency, in exchange for the key that can decrypt the victim's files.

Some ransomware families exhibit worm-like behavior, a tactic that involves actively seeking out and infecting additional victims within the same network. This propagation method allows the malware to rapidly spread and escalate the impact of the attack. For instance, the WannaCry ransomware, which emerged in 2017, demonstrated worm-like characteristics by exploiting a Windows vulnerability to automatically replicate itself across interconnected systems. Another example is the NotPetya ransomware, which, in 2017, utilized a similar technique to quickly infect numerous computers within large organizations. By autonomously moving through network-connected devices, these ransomware variants can efficiently compromise multiple systems, creating widespread disruption and increasing the potential for a higher ransom payout as the scale of the attack expands.

Ransomware can be classified into different categories based on various criteria. Ransomware can be classified based on its target victim, distinguishing between Consumer Ransomware, which focuses on individual users and personal devices, often employing tactics like phishing and malicious websites; Enterprise Ransomware, which specifically targets businesses, organizations, and institutions, seeking financial gain or the disruption of operations through more sophisticated and potentially targeted attacks; and Critical Infrastructure Ransomware, which poses a serious threat by targeting essential services like energy, transportation,

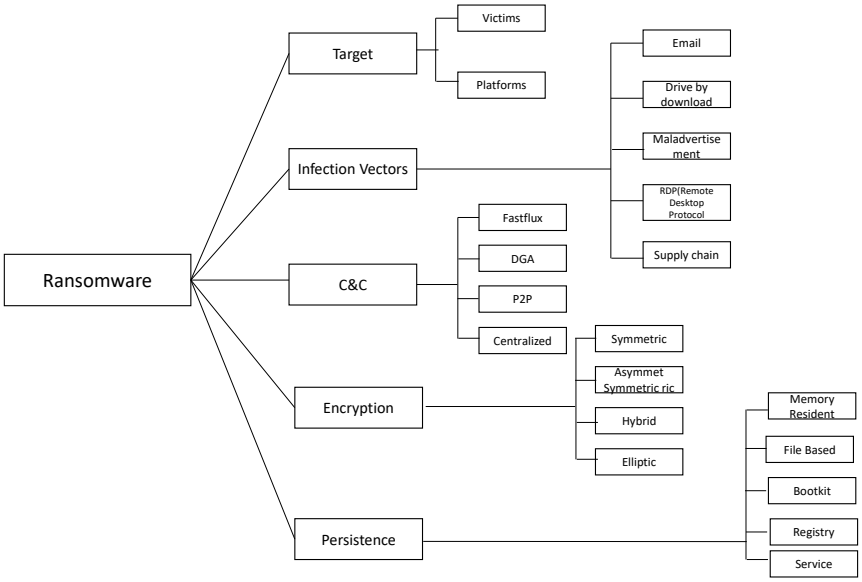

Figure 5-1: Taxonomy of Ransomware

healthcare, and utilities, aiming to disrupt critical systems with potentially severe consequences. Understanding these distinctions is crucial for designing effective defense mechanisms tailored to the specific characteristics and motivations of the attackers based on their chosen victim type.

Ransomware can also be classified based on the platforms it targets, encompassing various operating systems. Windows Ransomware is designed to infect systems running Microsoft Windows, often spreading through malicious attachments or phishing emails. Android Ransomware targets mobile devices operating on the Android OS, commonly distributed through malicious apps or links. Linux Ransomware affects systems running the Linux OS, posing a threat to servers and other Linux-based environments. macOS Ransomware is crafted to exploit vulnerabilities in devices running Apple's macOS. Additionally, Cross-Platform Ransomware exhibits an advanced level of threat sophistication, capable of targeting multiple operating systems, including Windows, Linux, and macOS environments. Recognizing these distinctions is vital for implementing effective cybersecurity measures tailored to the specific platforms at risk, ensuring comprehensive protection against ransomware threats.

Ransomware can be classified based on its infection vectors, revealing the diverse methods attackers use to infiltrate systems. Email-based ransomware is delivered through phishing emails containing malicious attachments or links, targeting individuals and organizations through deceptive communication. Drive-by download ransomware exploits vulnerabilities in browsers or plugins, infecting users who visit compromised websites. Watering hole ransomware compromises

websites frequented by specific target groups, relying on the trust users place in familiar online spaces. Malvertising ransomware leverages malicious advertisements on legitimate websites to deliver ransomware payloads. Remote Desktop Protocol (RDP) ransomware exploits vulnerabilities in RDP services, gaining unauthorized access to systems through compromised credentials. Fileless ransomware operates in memory, evading traditional detection methods. Supply chain ransomware targets software supply chains by infecting trusted vendors' software or updates, impacting users who unknowingly download compromised versions. Recognizing these infection vectors is crucial for organizations to bolster their defenses and adopt proactive measures against diverse ransomware threats.

Ransomware can be classified based on its Command and Control (C&C) communication methods, revealing distinct approaches attackers use to maintain control over infected systems. Centralized C&C ransomware relies on a single server for communication, making it vulnerable to detection and mitigation when the central server is identified. In contrast, Decentralized or Peer-to-Peer (P2P) C&C ransomware establishes communication between infected systems without relying on a central server, enhancing resilience against takedowns. Domain Generation Algorithm (DGA) C&C ransomware dynamically generates numerous domain names to establish communication, adding evasion capabilities by making it challenging to predict and block domains. Fast Flux C&C ransomware employs a rapidly changing network infrastructure, associating multiple IP addresses with a single domain name, complicating takedown efforts through continuous changes. Recognizing these C&C communication variations is vital for developing effective defense strategies against ransomware attacks.

Ransomware can be categorized based on the encryption techniques it employs, revealing variations in how attackers encrypt and subsequently hold files hostage. Symmetric encryption ransomware uses a single key for both encryption and decryption processes, making the encryption faster but necessitating secure key management. Asymmetric encryption ransomware utilizes a pair of keys—a public key for encryption and a private key for decryption—providing a more secure method for key management. Hybrid encryption ransomware combines aspects of both symmetric and asymmetric encryption, generating a unique symmetric key for each victim. Elliptic Curve Cryptography (ECC) ransomware utilizes elliptic curve-based encryption algorithms for enhanced security with shorter key lengths. Recognizing these encryption methods is essential for devising effective defense strategies against ransomware and developing appropriate mitigation measures.

Ransomware can be categorized based on its persistence methods, reflecting how it maintains a presence on infected systems. Memory resident ransomware operates in system memory without leaving traditional traces on the disk, evading detection by conventional security solutions. File-based ransomware writes executable files to disk, ensuring persistence through system reboots. Bootkit ransomware infects the Master Boot Record or boot sector, loading before the operating system for enhanced control. Registry persistence ransomware modifies the Windows Registry, making detection and removal more challenging. Service-based ransomware installs itself as a service in the Windows Service Manager, running discreetly in the background. Recognizing these persistence techniques is vital for cybersecurity professionals to develop effective defense strategies and mitigation measures against ransomware threats.

Cyber threats involving ransomware or ransom malware are growing at an exponential rate to extort money from individual users or organizations. The intent of such malware is to deny users access to their own systems by encrypting some files or even the whole system. In return, the attacker demands ransom that needs to be paid through virtual currency, like bitcoin in order for users to regain access to their system. If the ransom is not paid, they lose their valuable data indefinitely [64]. In the recent past, several defense solutions have been proposed to combat ransomware. To combat malware, it is necessary to analyze the malware binaries properly to extract the IoCs to distinguish malware from goodware. As shown in Figure 5-2, malware files are fed as input to the host machine of the analysis framework, which executes the binaries in an emulated environment and records their static and dynamic characteristics, and provides an output report to the user. These reports are finally used to extract the IoCs.

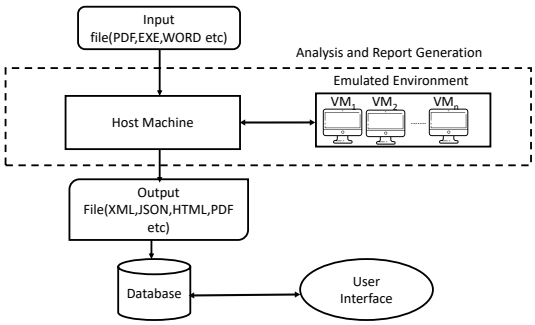

Figure 5-2: A generic architecture of malware analysis framework

5.3 Problem Statement

Let D represent the set of all data instances, where each instance d_i is a high-dimensional vector representing the features extracted from ransomware.

$$D = \{d_1, d_2, ..., d_n\}$$

Each data instance d_i is associated with a binary label y_i, indicating whether the instance is benign ($y_i = 0$) or malicious ($y_i = 1$). The challenge lies in devising a discriminative model $f : D \rightarrow \{0, 1\}$ that accurately predicts the ransomware label for new, unseen instances.

To enhance the detection accuracy, it is essential to identify relevant features and establish a mapping $g : D \rightarrow F$, where F is the feature space. The relationship between the feature space and the ransomware label can be expressed as:

$$y_i = f(g(d_i))$$

The proposed ransomware detection method aims to optimize the model parameters θ to minimize a suitable objective function $J(\theta)$. The optimization problem can be formulated as:

$$\theta^* = \arg\min_{\theta} J(\theta)$$

where $J(\theta)$ encapsulates a combination of factors such as false positive rate, false negative rate, and model complexity.

5.4 Proposed Framework

This section presents the proposed detection framework, referred to as ERAND, and discusses its components. Figure 7-3 illustrates the architecture of the proposed framework and its components.

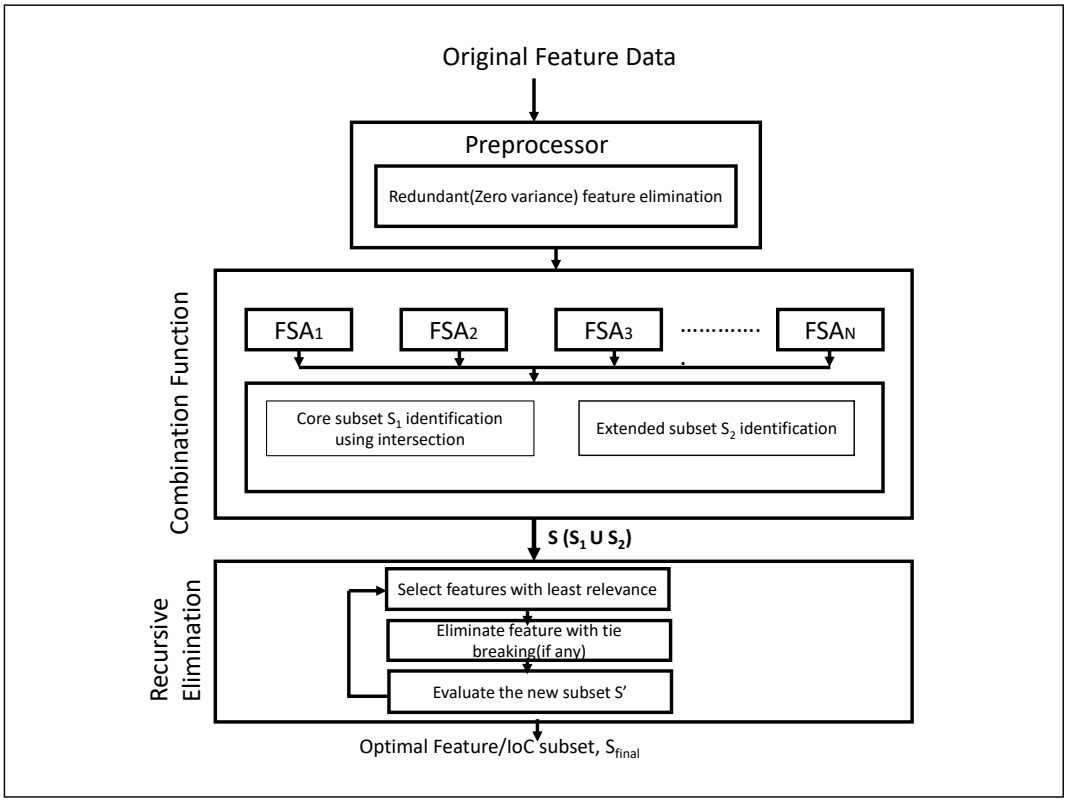

Figure 5-3: ERAND Detection framework for Ransomware and its variants

5.4.1 Data Collection

We evaluate our method using two datasets. The first dataset consists of 2,288 ransomware samples and 933 goodware samples. This dataset includes two categories of features: a) Permission-based features and b) API-based features. In total, there are 241 features, with 214 belonging to the permission-based category and the remaining 27 belonging to the API category. The entire process of generating this feature dataset is described in detail in Chapter 3. The final description of this dataset is provided in Table 5.1.

Table 5.1: Ransomware dataset with two classes

No. of Instances		No. of classes	No. of Features	
Ransomware	2288	2	Permission-based	214
Goodware	933		API-based	27
Total Instances:	3221		Total Features:	241

For our experiment, we also use another feature dataset from Sgandurra et al. [29]. This dataset contains 582 samples of ransomware, encompassing 11 different variants, and 942 samples of benign programs. The dataset includes a comprehensive set of 30,962 attributes representing all instances, both goodware

Table 5.2: Ransomware dataset with Normal and 11 ransomware subclasses

Sl no	Class	No of samples
1	Goodware	942
2	Critroni	50
3	CryptLocker	107
4	CryptoWall	46
5	KOLLAH	25
6	Kovter	64
7	Locker	97
8	MATSNU	59
9	PGPCODER	4
10	Reveton	90
11	TeslaCrypt	6
12	Trojan-Ransom	34
		Total samples: 1524
		Total features: 30962

and ransomware, present in the dataset. These attributes provide a wide range of features for in-depth analysis and evaluation of our method. A detailed description of this dataset is provided in Table 5.2.

5.4.2 Data Preprocessing

Data preprocessing helps convert the input data into an appropriate format to make it suitable for further analysis. In our work, we preprocessed our original dataset before performing subsequent analysis. In the preprocessing phase, we removed all those attributes whose variance is zero. In addition to that, we have discarded all those malware binaries that are failed to execute during analysis phase.

5.4.3 Ensemble Feature Selection

Data mining or machine learning algorithms may face the curse of dimensionality issues while dealing with high dimensional data. Additionally, the learning models may overfit in the presence of a large number of features which may lead to performance degradation, in addition to increased memory requirements and computational cost. Therefore, it is necessary to remove irrelevant features. Since both the datasets have large number of features (241 for dataset 1 and 30,962 for dataset 2), we perform feature selection to find an optimal subset of features. Each variant of ransomware exhibits different characteristics and as such feature selection techniques are used for each of them present in our dataset. More specifically, we use an ensemble feature selection approach, where the results of the base feature selection algorithms are combined with a consensus to generate an optimal

subset of features.

5.4.3.1 Selection of the base feature selection algorithms

Ensemble feature selection eliminates the biases of individual participating feature selection methods to yield the best possible output using an appropriate consensus function. Similar to other ensemble approaches, there are two steps in ensemble feature selection. First, we need a set of well-performing feature selectors, each of which provides a subset of features found relevant. Second, based on the output received from each ranker or feature selection algorithm, we apply an appropriate consensus function to yield the best possible subset of relevant features. Guided by our experimental results, we use the following three feature selectors, namely, CMIM[31], MIFS[32], and ReliefF[33].

1. **Conditional Mutual Information Maximization (CMIM)** [31]: For a given set of selected features, CMIM feature selector functions in an iterative manner by selecting features with maximum mutual information with the class labels. In other words, CMIM discards those features which are similar to the previously selected features even though its predictive power is strong concerning the class labels. For each unselected feature X_i, the feature score is calculated using Equation 1.

$$J_{CMIM}(X_i) = min_{X_j \epsilon S}[I(X_i; Y|X_j)] \tag{5.1}$$

2. **Mutual Information Feature Selection (MIFS)** [32]: A good feature should be highly correlated with the class label, but it should not have high correlation with other features. Both feature relevance and feature redundancy are taken into consideration by MIFS during feature selection phase. The feature score is calculated for each unselected feature X_k using Equation 2.

$$J_{MIFS}(X_K) = I(X_k; Y) - \beta \sum_{X_j \epsilon S} I(X_j; X_k) \tag{5.2}$$

3. **ReliefF** [33]: ReliefF selects features to efficiently separate instances from different classes. Assume that l data instances are randomly selected among all n instances, then the feature score of any feature f_i in ReliefF is estimated using Equation 3

$$ReliefFscore(f_i) = 1/c \sum_{j=1}^{l} (-1/m_j) \sum_{x_r \epsilon NH(j)} d(X(j,i) - X(r,i)) + \sum_{y \neq y\_j} 1/h_{jy} p(y)/(1-p(y)) \sum_{x_r \epsilon NM(jy)} d(X(j,i)X(r,i)) \quad (5.3)$$

where $NH(j)$ and $NM(j,y)$ are the nearest instances of x_j in the same class and in class y, respectively. Their sizes are m_j and h_{jy}, respectively. $p(y)$ is the ratio of instances with class label y.

5.4.3.2 Combination function to generate initial feature subset

In this work, we introduce a 3-step consensus building process to identify an intial optimal subset of features for the subsequent recursive optimality test.

C1 Consider the intersection of selected feature subsets given by each base feature selection algorithm to obtain a core subset of features, which we denote as S_1.

C2 Consider the scores of all features given by all individual algorithms and select those features (other than those included in S_1) for each algorithm with scores higher than a user defined threshold (say, α). We consider all those features whose score value is greater than 0. Finally, this step outputs a feature set S_2 based on contributions from all the feature selection algorithms.

C3 Obtain the initial optimal feature subset S by taking union of S_1 and S_2 for consideration of the recursive optimality test, described next, towards generation of the final optimal feature subset, i.e., S_{final}.

5.4.3.3 Generation of final optimal feature subset using recursive optimality test

In this section, a recursive elimination method is applied to get the final optimal feature subset S_{final} based on S. There are three steps in this process, which are stated below.

O1 Consider the feature subset, i.e., S as input and identifies one or more features with least relevance score.

O2 Eliminate the feature(s) to obtain a new subset, S'. In case of tie, based on relevance score estimated for an idetified pair of candidate features, we choose feature for elimination given by an inconsistent performing ranker algorithm.

O3 Evaluate S' in terms of classification accuracy. It terminates the elimination process if and only if a significant performance degradation is observed in terms of accuracy due to the elimination of a feature, and considers the recent S_i as S_{final}.

5.4.3.4 Optimal Feature Subset for each class and for the whole family

Initially, the three feature selection algorithms namely: CMIM, MIFS and ReliefF generate 3 feature subsets for each variant of ransomware. The naming convention for each subset is Ran_sub_{ij}, where i goes from *1* to n and j goes from 1 to 3. Next, we apply a consensus function (described in Section 5.4.3.2) on these subset of features to generate a common subset for each class of ransomware. This generates n feature subsets for each variant of ransomware present in the dataset. Now, for each of the n feature subsets, an optimal feature subset is identified using the process described in the Section 5.4.3.3. The number of optimal features identified for each dataset with each variant of ransomware is given in the Table 5.3. Table 5.4 lists all the optimal features for ransomware dataset 1 and Table 5.5 lists top ranked feature categories for dataset 2.

Table 5.3: Number of optimal features for each ransomware dataset

Dataset	No. of optimal features
Dataset 1	15
Dataset 2 (class 1)	23
Dataset 2 (class 2)	36
Dataset 2 (class 3)	19
Dataset 2 (class 4)	42
Dataset 2 (class 5)	20
Dataset 2 (class 6)	36
Dataset 2 (class 7)	31
Dataset 2 (class 8)	4
Dataset 2 (class 9)	32
Dataset 2 (class 10)	8
Dataset 2 (class 11)	25

Table 5.4: List of selected features for dataset 1

Feature Rank	Feature Name
1	$SEND\_SMS$
2	$RECEIVE\_BOOT\_COMPLETED$
3	GET_TASKS
4	$Ljava/net/URL; -> openConnection$
5	$VIBRATE$
6	$WAKE\_LOCK$
7	$KILL\_BACKGROUND\_PROCESSES$
8	$SYSTEM\_ALERT\_WINDOW$
9	$ACCESS\_WIFI\_STATE$
10	$DISABLE\_KEYGUARD$
11	$Landroid/location/LocationManager;$ $-> getLastKnownLocation$
12	$READ\_PHONE\_STATE$
13	$RECEIVE\_SMS$
14	$CHANGE\_WIFI\_STATE$
15	$WRITE\_EXTERNAL\_STORAGE$

Table 5.5: List of top ranked feature categories for dataset 2

Feature Rank	Feature Category
1	$RegistryKeysOperations$
2	$APICalls$
3	$Strings$
4	$Filesoperations$
5	$Fileextensions$
6	$Directoryoperations$
7	$Droppedfiles$

5.4.3.5 Correctness of feature selection results

To establish the correctness of the feature selection results, the following two lemmas are proposed.

Lemma 5.4.1. *For discrimination of each ransomware variant, i.e., R_i from benign instances, selected feature subset, i.e. Ran_sub_i is optimal.*

Proof: Suppose for the sake of contradiction, we assume that for a ransomware variant R_i, Ran_sub_i is non-optimal and $|Ran\_sub_i| = k$. Also, assume that $|Ran\_sub_i| + 1$ gives us the best possible accuracy. However, as shown in Figure 5-4 and also stated in Section 5.4.3, the highest possible accuracy to discriminate R_i from normal or goodware after recursive elimination is for the subset of features Ran_sub_i. Any addition or deletion of features from Ran_sub_i does not improve accuracy. Hence, the assumption is false and it contradicts the non-optimality assumption. Hence, Ran_sub_i is optimal.

Lemma 5.4.2. *The accuracy given by Ran_sub_{all} cannot be greater than the overall highest accuracy given by the individual optimal subset of features i.e., Ran_sub_i.*

Proof: A feature subset Ran_sub_i for each ransomware variant R_i is identified and its optimality as stated in the Lemma 5.4.1 is established. Ran_sub_{all} includes Ran_sub_i and some additional features. In other words, for discrimination of R_i, Ran_sub_{all} includes some additional redundant features with reference to Ran_sub_i. Such additional features cannot improve classifier performance for R_i (as substantiated by Lemma 5.4.1). Hence, the accuracy given by $Ran\_sub_{all} \not> Ran\_sub_i$ (where i varies from 1 to 11).

5.4.4 Classification of Ransomware family and Its Variants

After obtaining the dataset using an optimal feature subset, it is necessary to establish the performance of the features to distinguish the ransomware from the goodware and a supervised approach can be used to achieve the same. To avoid biases of classifiers, we consider an ensemble approach for unbiased evaluation of the optimality of the subset of IoCs given by the previous step. However, ensemble methods are also not totally free from drawbacks due to the limitations of the inherent combination/consensus function used. Therefore, in our study, we consider a number of consistently performing ensemble classifiers for optimal classification. The five ensemble classifiers used in our work are Random forest[65], Extra Tree [66], Adaboost[67], Gradient boosting [68], and XGBoost[69].

To avoid individual biases of these ensemble methods, we combine their outputs to yield the best possible classification performance by minimizing false alarms.

5.4.4.1 Balancing of classes

Data balancing is a technique to make an approximately balanced number of instances in each class. An imbalanced data distribution may lead to unusual model performance. Our dataset is highly imbalanced as shown in Table 5.2 of class distribution and hence, it needs to be balanced. For data balancing, we need additional class specific instances which are difficult to collect because ransomware data are not readily available due to various other reasons in addition to security reasons. Therefore, we use a sampling technique to handle the class imbalance problem. In general, there are three sampling techniques: undersampling, oversampling, and hybrid sampling. In our case, we use an oversampling technique called SMOTE [70], where the minority class is oversampled by creating "synthetic" examples rather than by over-sampling with replacement. Finally, in the dataset all the class instance distributions become 50:50. In addition to that, we also validate our method's performance in other data distributions also like 60:40, 70:30 and 80:20.

5.4.4.2 Classification of Ransomware Variants

The above mentioned classifiers, i.e., Random forest, Extra tree, Adaboost, Gradient boosting, and XGboost are used to build a predictive model for each Ran_sub_i

dataset, where i goes from 1 to 11 and each dataset includes the instances of goodware and ransomware variants.

5.4.4.3 Classification of Ransomware family

The classifier models are also built using the Ran_sub_{all} dataset, where all the instances of ransomware variants are inlcuded as a single ransomware class along with the goodware instances.

5.5 Results

The experimental analysis is performed in a Python environment. The experiments are carried out on a workstation with 64 GB main memory, 2.26 Intel(R) Xeon processor and Ubuntu operating system.

To build consensus based on decisions given by the individual classifiers, we use a weighted majority approach. The approach uses individual weights of the classifiers given by a multiobjective optimization technique for unbiased combinations of the individual decisions to achieve best possible accuracy.

5.5.1 Computation of weights for classifiers using NSGA-II

We use multiobjective evolutionary method to compute best possible set of weighting factors for the participating classifiers based on their classification performances on each of the 11 variants of ransomware. Since none of the classifiers has been found to give winning performance consistently for all the variants of ransomware, we decided to exploit weighted majority based combination function to achieve best possible classification accuracy. A good number of multiobjective evolutionary algorithms are available in the literature and some of their comparisons are available at [71] [72]. NSGA-II has already been established as a promising optimization method to handle multiobjective problem due to its elitist approach. An arithmetic crossover and Gaussian mutation operators generates offspring population from parent population. Offspring populations are then added to the current population. The NSGA-II algorithm uses (i) non-dominated sorting method for ranking individuals into different non-domination levels and (ii) crowding distance method to sort individuals within the same level. An individual

dominates another individual if it is strictly better in at least one objective and no worse in all the other objectives.

In this work, we exploit the NSGA-II to compute an optimal set of weighting factor for the five classifiers (c_1 to c_5) based on their individual performances for 11 variants. We use their 5x11=55 performance values as input to the NSGA-II for computation of the optimal set of weights. The optimal weights obtained are given in Table 5.6.

Table 5.6: Weightage of the classifiers given by NSGA-II

Classifier, c_i	Weightage
ExtraTree (c_1)	0.35
Gradient Boosting (c_2)	0.28
AdaBoost (c_3)	0.15
XGBoost (c_4)	0.12
Random Forest (c_5)	0.10

5.5.2 Weighted Majority Based Combination Function

To generate an unbiased classification output, ERAND uses 'weighted majority voting' to build consensus among the outputs given by the individual classifiers. We intially carry out an exhaustive experimentation with these classifiers using the datasets described in Section 3.1 and consider their performance as the basis for subsequent processing to decide their weights while building the consensus. Our experimental study based on weighted majority voting uses equation 4 to decide the class label of a test instance. It computes anomaly score, S_i for each test instance given by equation 4 to recognise either as 'goodware' or 'malware' with respect to a user defined threshold, β.

$$S_i = w_1c_1 + w_2c_2 + w_3c_3 + w_4c_4 + w_5c_5 \tag{5.4}$$

If the value of $S_i \geq \beta$ (a user-defined threshold), the instance is considered anomalous or belonging to a malware class. In our experimentation, we consider β =0.63. Because if any two best performers (like (c_1,c_2) or (c_1,c_3) agree, then their total weights are used as threshold. The value of c_i for a given class can be either 1 (if belongs to the class) or 0 (if not). In case of tie, we give priority to

the decision of the best performers. For example, if c_1 and c_4 are on one side and c_2, c_3 and c_5 are on the other side, we prefer the decison of (c_1, c_4).

5.5.3 Classification of Ransomware Variants

We evaluate the performance of the selected subset of features to distinguish the instances of the ransomware family from the goodware, using five classifiers individually. The classification results of dataset 1 and dataset 2 are given in Tables 5.7 and 5.8. It can be observed from Tables 5.7 and 5.8 that ERAND consistently performs well like ExtraTree and Gradient Boosting in classifying each variant of the ransomware malware.

Table 5.7: Classification accuracies of dataset 1

Classifiers	Accuracy
Gradient Boosting	97.5
Adaboost	96.47
Random Forest	98
Extra Tree	98.27
XGBoost	97.2
ERAND	98.2

Table 5.8: Classification accuracies of dataset 2 for each variant

Ransomware Family	Classifiers					
	Gradient Boosting	Adaboost	Random Forest	Extra Tree	XGBoost	ERAND
Critroni	98.7	98.7	98	98.1	98.7	98.8
CryptLocker	96.8	96.8	96.7	96.8	96.7	96.8
CryptoWall	96.3	96.17	96.23	96.44	96.33	96.42
KOHLER	98.6	98.56	98.61	98.8	98.7	98.83
Kovter	98.76	97.88	97.8	98.1	98.23	98.78
Locker	96.39	96.34	96.50	96.92	96.39	96.55
MATSNU	97.61	97.70	97.77	97.70	97.88	97.88
PGPCODER	98.12	98.12	98.43	98.7	98.43	98.7
Reveton	98.40	98.72	98.84	97.7	98.72	98.79
TeslaCrypt	98.87	98.62	98.77	98.6	97.88	98.86
Trojan-Ransom	98.43	97.78	98.32	98.61	97.86	98.67

5.5.4 Classification of Ransomware Family

To calculate the performance of the selected features to distinguish whole ransomware family from the goodware, we used five classifiers namely, Random forest, Gradient boosting, Adaboost, Extratree and XGboost. The classification results of whole ransomware family with respect to goodware are enlisted in Table 5.9.

5.5.5 Result analysis and Performance comparison

For effective evaluation of our method, four machine learning performance metrics are used namely: Accuracy, Recall, Precision, and F-Measure. The precision,

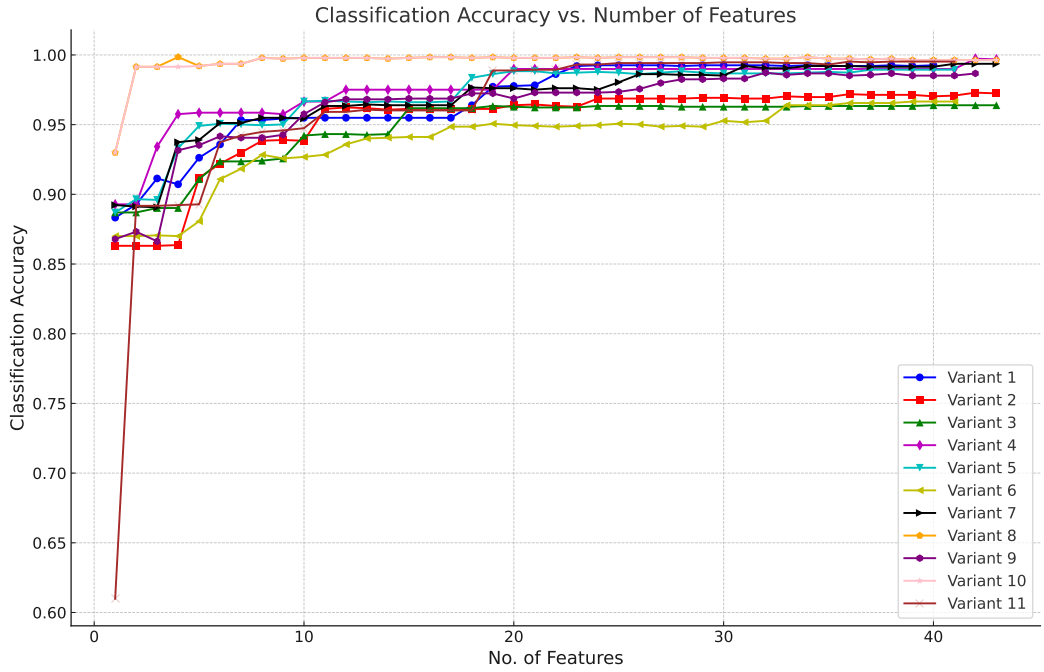

Figure 5-4: Number of features vs Classification Accuracy

Table 5.9: Classification accuracies of whole ransomware family

Goodware	Ransomware	Classifiers					ERAND
		Random Forest	Extra Tree	AdaBoost	XGBoost	Gradient Boosting	
942	618	97.8	98.7	97.9	98.1	98.6	98.6

recall, and F1 Score values of each classifier are reported in Table 5.10. On the other hand, the classification accuracies of each classifier are reported in Table 5.8 for 50:50 class distribution.

- *Accuracy*: The no of instances correctly detected by a classifier divided by the total of goodware and ransomware instances gives the accuracy.

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

- *Precision*: It defines what proportion of predicted ransomware are actually correct. Thus, Precision of a model is calculated as follows:

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

- *Recall*: It defines what proportion of all ransomware samples are correctly predicted. The recall of a model is calculated as follows:

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

- *F1-score* is calculated by taking the weighted average of precision and recall. F1-score is defined as follows:

$$F1\ \ Score = \frac{2 * (Recall * Precision)}{(Recall + Precision)}$$

Where TP: True Positive, TN: True Negative, 1.65cmFP: False Positive, FN: False Negative. In order to evaluate the performance of the proposed method, we used the K-Fold cross-validation where we set the value of K=10. Table 5.10 presents the Precision, Recall, and F1 scores of all the classifiers. The proposed method is evaluated in comparison with existing ransomware detection methods, highlighting similarities and differences. Like the approaches in [73], [74], and [75], the proposed method employs various supervised approaches for classifying ransomware instances into their respective families. While these studies focused on multi-class classification of ransomware instances against 8, 9, and 7 families of ransomware respectively, the proposed method uses 11 families for validation.

Similar to the dataset in [73], the dataset used by the proposed method is not balanced in terms of goodware and ransomware samples. However, unlike [73], the proposed method applies data balancing techniques to validate the method with different class distributions. Additionally, the proposed method diverges from [73] and [74] by using platform-specific ransomware and extracting both dynamic and static behaviors as features to distinguish ransomware from goodware.

In [75], stable performance was obtained using 131 features, which is significantly higher compared to the proposed method, where the maximum feature dimension used is 45 and the minimum is 4, achieving stable performance among ransomware families. Similarly, [73] reported stable performance with 123 features, while the proposed method achieves this with significantly fewer features. [74] achieved the highest accuracy with 8 features at 97.10%, whereas the proposed method reaches a higher accuracy of 98.7% with only 4 features for Dataset 2 (class 8).

Furthermore, unlike [73] and [74], the proposed method achieves better accuracy in terms of classification. The best accuracy obtained by [73] was 91.43%, while the proposed method achieved 98.7%. Table 5.11 presents a comparative study of the proposed method with some of the existing methods.

Table 5.10: Precision, Recall, and F1 score of all the classifiers

Datasets	Precision					Recall					F1-Score				
	Gradient Boosting	Adaboost	Random Forest	Extra Tree	XGBoost	Gradient Boosting	Adaboost	Random Forest	Extra Tree	XGBoost	Gradient Boosting	Adaboost	Random Forest	Extra Tree	XGBoost
Dataset 1	98.1	97	98.9	98.7	98.4	97.9	97.5	98.1	98	97.6	98	97.22	98.5	98.6	97.8
Dataset 2 (Class 1)	98.9	98.89	98.89	98.88	98.89	98	98.3	98.3	98	98.4	98	98.1	97.9	98.7	98.4
Dataset 2 (Class 2)	95.7	95.9	96.8	96.2	95.8	98.8	98.8	97.5	98.7	98.3	97.2	97.4	96.7	97.2	97.1
Dataset 2 (Class 3)	94.3	93.2	93.4	93.3	93.1	98.7	98.6	98.5	98.8	98.7	96.5	96.2	96.4	96.5	96.3
Dataset 2 (Class 4)	98.7	98.7	98.7	98.8	98.8	98.6	98.6	98.6	98.6	98.6	98.7	98.7	98.7	98.7	98.7
Dataset 2 (Class 5)	98.5	98.7	98.2	98.6	98.6	98.3	98.6	98	98.5	98	98.9	98.6	98.7	98	98.8
Dataset 2 (Class 6)	95.4	94.5	94.7	94.6	94.3	98.1	98.3	98.6	98.2	98.7	96.7	96.5	96.6	96.8	96.4
Dataset 2 (Class 7)	98.6	98.4	98.7	98.5	98.6	98	98.8	98.2	98.7	98.9	98.3	98.1	98.2	98.1	98.3
Dataset 2 (Class 8)	97.7	98.8	98.76	98.8	98.88	97.8	97.8	97.7	98.8	97.8	97.8	97.9	97.8	98.7	97.9
Dataset 2 (Class 9)	98.1	98.6	98.3	98.4	98.2	98.2	98.5	98.6	98.7	98.2	98.6	98	98.89	98.89	98.7
Dataset 2 (Class 10)	98.4	98.7	98.8	98.8	97.3	98.7	98.6	98.5	98.88	97.5	98.6	98.7	98.8	97.8	98.4
Dataset 2 (Class 11)	98.5	98.4	98.8	98.6	98.4	98.4	98.3	98.3	98.5	98.6	98.5	98.4	98.5	98.6	98.5