Applied Soft Computing 121 (2022) 108744

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Behavior-based ransomware classification: A particle swarm optimization wrapper-based approach for feature selection

Muhammad Shabbir Abbasi [a,b,*], Harith Al-Sahaf [a], Masood Mansoori [a], Ian Welch [a]

[a] School of Engineering and Computer Science, Victoria University of Wellington, P.O. Box 600, Wellington 6140, New Zealand
[b] Department of Computer Science, University of Agriculture Faisalabad, Punjab, Pakistan

ARTICLE INFO

Article history:
Received 11 October 2021
Received in revised form 11 February 2022
Accepted 10 March 2022
Available online 23 March 2022

Keywords:
Evolutionary computation
Ransomware detection
Feature selection
Particle swarm optimization

ABSTRACT

Ransomware is malware that encrypts the victim's data and demands a ransom for a decryption key. The increasing number of ransomware families and their variants renders the existing signature-based anti-ransomware techniques useless; thus, behavior-based detection techniques have gained popularity. A difficulty in behavior-based ransomware detection is that hundreds of thousands of system calls are obtained as analysis output, making the manual investigation and selection of ransomware-specific features infeasible. Moreover, manual investigation of the analysis output requires domain experts, who are expensive to hire and unavailable in some cases. Machine learning methods have shown success in a wide range of scientific domains to automate and address the problem of feature selection and extraction from noisy and high-dimensional data. However, automated feature selection is under-explored in malware detection. This study proposes an automated feature selection method that utilizes particle swarm optimization for behavior-based ransomware detection and classification. The proposed method considers the significance of various feature groups of the data in ransomware detection and classification and performs feature selection based on groups' significance. The experimental results show that, in most cases, the proposed method achieves comparable or significantly better performance than other state-of-the-art methods used in this study for benchmarking. In addition, this article presents an in-depth analysis of the significance of various features groups and the features selected by the proposed method in ransomware detection and classification.

1. Introduction

Ransomware is malware that extorts money (ransom) by holding hostage a victim's data, usually through encryption [1–6]. Cybercriminals collect the ransom as crypto-currency, mainly Bitcoins, to hide their identity [7]. Ransomware falls into two main classes: *lockers* and *crypto-ransomware*, with the latter being more commonplace and posing a severe threat compared to the former [8].

Ransomware evolved from a low impact *PC-Cyborg* (also known as *AIDS*) attack [3,6,9–12], to high impact attacks such as *WannaCry*, *Cryptolocker*, *Cryptowall*, and *Locky*. The number of variants of ransomware has been on the rise since 2012. Variants of ransomware grew from 1 to 193 between 2012 and 2016 [13]. Over this time, ransomware came to prominence as a rapidly growing threat in cybersecurity. Ransomware-as-a-Service (RaaS) families appeared in 2017, causing high financial losses globally, for example, Cryptolocker, CryptoWall, Locky, and TeslaCrypt [14].

Ransomware attacks have escalated as they allow attackers to earn easy money amounting to millions per month [3]. An estimated number of ransomware attacks in 2017 was about 184 million, where most of the victims of these ransomware attacks comprised of general users [9]. The year 2018 witnessed around 850 million ransomware attacks [15]. According to Verizon[1] ransomware accounted for 39% share of overall malware attacks [16]. The estimated global damage by ransomware in 2019 and 2020 was around 11.5 and 20 billion US dollars, respectively [17].

Detection systems utilize various static and dynamic attributes of the malware, such as signatures and system call patterns, identified through rigorous malware analysis [18]. Static detection techniques, such as signature matching, can be easily evaded by techniques, e.g., polymorphism and metamorphism, employed by many ransomware families [19]. In order to overcome the

* Corresponding author at: School of Engineering and Computer Science, Victoria University of Wellington, P.O. Box 600, Wellington 6140, New Zealand.
E-mail addresses: Shabbir.Abbasi@ecs.vuw.ac.nz, Shabbir.Abbasi@uaf.edu.pk (M.S. Abbasi), Harith.Al-Sahaf@ecs.vuw.ac.nz (H. Al-Sahaf), Masood.Mansoori@ecs.vuw.ac.nz (M. Mansoori), Ian.Welch@ecs.vuw.ac.nz (I. Welch).

[1] An American multinational communication conglomerate.

https://doi.org/10.1016/j.asoc.2022.108744
1568-4946/© 2022 Elsevier B.V. All rights reserved.

problems of static detection techniques, behavior-based detection techniques (methods based on dynamic analysis of malware) have gained popularity as these techniques do not rely on matching static patterns [20].

Researchers have adopted various ways of behavior-modeling of ransomware using the execution traces of a process identified through dynamic analysis, including, but not limited to, monitoring file system activity [21,22], network traffic [23,24], and Application Programming Interface (API) calls [25,26]. The most widely adopted technique of behavior modeling uses system-call traces of a process (goodware or malware). System calls pattern provide information on the similarities and differences of execution patterns between malware and goodware [27]. Machine learning techniques that use system calls as features face scalability problems because of high dimensionality [28]. Classification methods based on behavior are more robust against evasion techniques than methods based on syntax [29].

The process of identifying distinct and relevant behavioral features for accurate classification of malware and goodware from a myriad of noisy and redundant features collected through dynamic analysis of malware is a challenge. Manual investigation of analysis logs is time-consuming, tedious, and requires domain expertise, which cannot keep pace with increasing ransomware variants. Machine learning techniques for feature selection could automate the process and address the problems associated with the manual investigation of many system calls.

Feature selection is selecting a subset of features from a given set of features based on feature ranking or feature evaluation criteria [30–32]. There are two main approaches to feature selection, filter and wrapper. Filter methods usually rely on feature ranking criteria, e.g., Chi-square (χ^2), correlation, and mutual information (MI), to evaluate the importance of features and are independent of any classification algorithm [33]. Wrapper features selection methods, on the other hand, use classification algorithms (known as wrapped classification algorithms) for iterative evaluation of the selected subset of features and so are dependent on the classification algorithm [34]. Feature selection has been applied in a variety of application areas, such as image recognition [35,36], intrusion detection [37], malware detection and classification [26, 38], bioinformatics [39] and healthcare [40]. It has proven to be effective in reducing the dimensionality without degradation of, if not improving, the classification performance [31,41].

Some of the existing ransomware detection methods are based on the behavior of ransomware, such as [19,26,42,43]. A few of these methods utilize manual ways of selecting features/indicators of ransomware activity, e.g., [19], while others adopt machine learning-based feature selection techniques to cope with the problem of redundancy and high dimensionality. However, even machine learning-based solutions require human input and thus are not fully automated. For instance, the method [26] requires human input to decide the number of features to be selected for better classification performance.

Particle Swarm Optimization (PSO) is a widespread evolutionary computation (EC) technique commonly used for optimization [44]. It generates a swarm of particles (candidate solutions) and iteratively improves the performance of these particles as measured by a fitness function. PSO has also been used in feature selection problems as a wrapper-based feature selection method [45–49], which is widespread in various domains, such as image recognition [35], and spam detection [50].

1.1. Contributions

This study proposes an automated wrapper-based feature selection method for behavior-based ransomware detection and classification that utilizes PSO to select an appropriate number of features from each feature group based on its relevance to the task.[2]

The main contributions of this study are:

1. In behavior-based ransomware detection and classification, the problem of feature selection from high-dimensional ransomware behavioral data is addressed using a group-based strategy through PSO.
2. The proposed method does not require a predefined number for selecting the features as input, unlike filter methods. In our method, PSO automatically selects an appropriate number of features from each feature group considering its relevance to ransomware detection and classification; thus, the proposed method does not require human input regarding the number of the features to be selected.
3. An insight into the features selected by the proposed method is provided to highlight the significance and relevance of some of these features with ransomware detection and classification.

1.2. Benchmarking

We compared the results of the proposed method in terms of performance and number of selected features with two other state-of-the-art feature selection methods, i.e., Variable-Length Particle Swarm Optimization (VarLenPSO) [47] and Self-adaptive Particle Swarm Optimization (SaPSO) [49]. We, further, wanted to benchmark the results of the proposed method against other existing state-of-the-art machine learning-based ransomware detection methods, for instance, [52–54]. However, these methods require a sequential data set. The data set that we used in this study does not provide the sequence of the performed operations. Neither these methods can be applied to the data set at hand, nor the proposed method is compatible with sequential data sets.

1.3. Organization

The rest of the paper is organized as follows; Section 2 discusses related work and the method used as a baseline to this study. Section 3 presents the proposed feature selection method. Section 4 gives the details of experimental design and settings. Section 5 discusses the results of the experiments. Then in Section, an6 analysis of the results is presented. Section 7 presents the conclusions and some future directions.

2. Literature survey

This Section covers a discussion on existing studies related to behavior-based ransomware detection. Moreover, it provides a discussion on swarm intelligence (SI) and existing state-of-the-art PSO-based feature selection methods. Furthermore, It introduces a state-of-the-art automated behavior-based ransomware detection model, [26] which is considered the baseline method for our experiments.

2.1. Related work

Behavior-based detection requires the identification of patterns that are associated with ransomware. One method to identify these patterns is to run ransomware in a controlled environment and monitor its interaction with the system resources. A study [1] based on an analysis of 1,359 samples identified that the ransomware performed an unusually large number of interactions with the file system while encrypting the victim's

[2] We built upon the work reported in one of our earlier papers [51].

data compared to other benign processes. The abnormal level of filesystem activity suggested that monitoring the file system activity of a process is a useful strategy for distinguishing between ransomware and goodware execution. In particular, the researchers suggested matching the patterns of I/O Request Packets (IRP) and monitoring Master File Table (MFT) changes to detect crypto-ransomware. Most of the work in [1] was related to the identification of behavioral patterns of ransomware in order to suggest a strategy to develop a ransomware detection model.

Building on the finding of [1], a system (*Unveil*) for ransomware analysis and detection was proposed in [21]. The researchers suggested matching the I/O traces of the processes running with user privileges with pre-developed behavioral patterns of ransomware to detect ransomware activity in a system. In addition to I/O traces, the entropy of data in read/write data buffers was also used to monitor the file system activity of user-level processes. Unveil, being a data-centric approach, could not detect ransomware until enough I/O activity had been observed, and this meant that some of the victim's files would be encrypted before the detection. Moreover, Unveil's detection could be bypassed by ransomware running at a kernel level.

ShieldFS [55] and *Redemption* [42] employ a similar data-centric strategy for detection of ransomware as used in Unveil. Both ShieldFS and Redemption introduced self-healing capability in the file system by maintaining a buffer of recently accessed files. This buffer was utilized to restore any files encrypted by the ransomware before detection. The solution was constrained by operational overheads associated with disk I/O. Scaife et al. [22] conducted a dynamic analysis of 14 different families of ransomware in order to investigate their file-centric behavior, e.g., file type change, the entropy of files, measurement of similarity, and deletion of a large number of files. As with other data-centric approaches of ransomware detection, CryptoDrop detected ransomware activity after encryption of 10 files by ransomware on average.

Cabaj et al. [24] proposed that monitoring network activity and intercepting the delivery of the public key to the victim's host system can be one method of preventing ransomware. The authors investigated the network activity of a victim machine during an attack of *Cryptowall* and observed that Cryptowall communicated with Command-and-Control (C&C) servers to obtain an RSA 2048-bit public key. This key was subsequently utilized for encryption of the victim's data. Cusack et al. [23] proposed a solution based on the findings of [24]. The researchers monitored the network traffic of the victim machine to identify ransomware before any files were encrypted by intercepting the delivery of the asymmetric public key. The method suffered from a high false positive rate of about 12.5%. Moreover, any ransomware that does not use the network to deliver keys could easily evade detection.

In [19], Daku et al. used behavior analysis reports of 10 different families of ransomware and proposed a ransomware classification model. The authors manually selected twelve behavioral attributes, such as family name, number of file accesses, and number of created processes to model ransomware behavior for classification.

In [56], an automated behavior-based ransomware analysis and detection technique is proposed. The authors created host logs of normal user activity and ransomware activity using a Cuckoo sandbox. Then three machine learning techniques, namely: term frequency–inverse document frequency (TF–IDF), Fisher's linear discriminant analysis (Fisher's LDA), and extra trees or extremely randomized trees (ET), were utilized for extraction of behavioral patterns from the cuckoo logs. The extracted behavioral patterns were then used to detect ransomware activity. One sample of seven different ransomware families was used for behavior modeling. The authors observed that TF–IDF was the best of the three methods used for feature discrimination in their experiments. The reported early detection (detection before encryption) accuracy for various families remained between 91.8% through 99.9% with no false positives. Normal user activity used in their experiments does not include the activities of goodware having similar behavior to ransomware which usually results in a high false positive rate.

Machine learning has been used in some of the existing ransomware detection methods, for example, in [55], and [23] Random Forest was used for ransomware detection. Besides other methods of machine learning, deep learning methods have also been used in some of the existing ransomware detection methods. For example, Long–Short Term Memory networks (LSTM) are used in [52] to classify sequences of API calls. Similarly, in [53], the author extended the LSTM-based method by incorporating the attended recent input cell for detection of Windows-based ransomware.

2.2. Swarm intelligence for feature selection

SI algorithms, especially PSO [46], are widely used for feature selection for their simplicity, natural representation, and effective search mechanism [57]. Tran et al. [48] proposed VarLenPSO, a new variable-length representation of PSO for improving efficiency in feature selection for high-dimensional data. To cope with high dimensionality, their algorithm proposed a novel initialization strategy, i.e., "population division", for PSO-based feature selection methods, wherein a swarm consisted of particles of variable lengths. Xue et al. [49] proposed SaPSO for large-scale feature selection problems. Unlike other variants of PSO that utilized only one strategy to generate new particles, SaPSO utilized multiple strategies for new particles' generation. The main focus of their work was generating multiple candidate solution generation strategies (CSGS) that were used self-adaptively during the evolutionary process.

Song et al. [58] proposed to distribute a high-dimensional search space in a feature selection problem to multiple low-dimensional search spaces based on the features' significance to class labels. For these smaller low-dimensional search spaces, multiple subswarms were evolved simultaneously to improve the search performance and scalability of the algorithm to cope with high-dimensionality. To cope with the limitations posed by commonly used swarm initialization strategies of PSO in high-dimensional feature selection problems, Song et al. [59] proposed a hybrid feature selection algorithm, "bare-bones particle swarm optimization (BBPSO)", using MI and PSO. The researchers developed operators for optimizing the search mechanism to avoid PSO trapped in local optima.

2.3. Background

An automated behavior-based ransomware detection model called *Elderan* [26] has been used as the baseline method in our experiments. Elderan is a machine learning-based method for ransomware detection that utilizes Cuckoo logs generated through dynamic analysis of ransomware and goodware. The samples are executed in a controlled environment for 30 s to analyze them dynamically at their installation stage. The behavior of these samples was then modeled in terms of program interactions with the system that were subsequently represented as features in the data set. The researchers extracted 30,967 features grouped into seven groups based on their type, e.g., names of the API calls and file names involved in file operations. A filter-based feature selection approach based on mutual information (MI) ranking was utilized to select a subset of features. 100 to 1,500 features were selected in various experiments that were

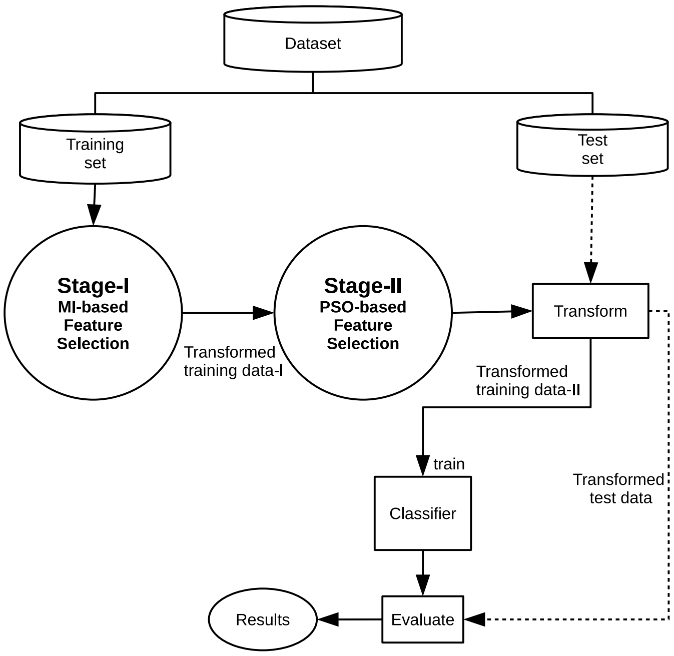

Dataset
Training set
Test set
Stage-I MI-based Feature Selection
Transformed training data-I
Stage-II PSO-based Feature Selection
Transform
Transformed training data-II
train
Classifier
Transformed test data
Results
Evaluate

Fig. 1. The overall algorithm of the proposed method.

subsequently fed to the Regularized Logistic Regression (RLR) classification algorithm to classify the samples into ransomware and goodware. The authors observed that selecting 400 features produced the best classification accuracy in their experiments.

However, Elderan suffered from a high false positive rate, and no experiments were performed for multi-class classification of samples into ransomware families. Furthermore, the authors did not analyze the impact of the selected features coming from various feature groups upon classification accuracy. The feature selection method used in Elderan selects top-ranked features from the entire data set, ignoring the impact of individual feature groups on the classification accuracy.

3. The proposed feature selection method

Based on various feature groups in the data set, we hypothesized that selecting an appropriate number of features from different feature groups may improve classification accuracy. Thus, we adopted a group-oriented strategy for feature selection to highlight the impact of individual feature groups on classification performance.

The feature selection method proposed in this study has two stages, as shown in Fig. 1. At Stage-I, an equal number of top-ranked features are selected from each feature group using a filter-based feature selection method based on MI ranking. At Stage-II, a wrapper-based method for feature selection is used to select an optimal number of features from each feature group.

3.1. Stage-I

Stage-I partitions the data set into m partitions, where m is the number of feature groups in the data set as shown in Fig. 2. Feature selection is then performed on each partition individually, where MI ranking is utilized to select t top-ranked features from each partition. t is an integer that is chosen based on the number of features contained in the API group. This ensures the representation of all features from the API group. API calls are considered important in modeling the behavior of a running process as any running process needs to make appropriate API calls (to invoke corresponding system calls) for getting desired OS services, e.g., I/O with the disk. The output of this stage is transformed data-I that has $m \times t$ features (t features from m groups), which is forwarded as input to Stage-II. At stage-I, selecting the equal number of top-ranked features from each partition removes disparity (in terms of the number of features) among all feature groups and ensures the representation of all feature groups in the transformed data. The pseudo code of Stage-I is presented in Algorithm 1.

Algorithm 1: Stage-I of the proposed method.

```
1 Function Stage-I(data, m=num_groups, t=num_fs, rank_func):
2     P[ ] = partition(data, m)    % Based on the feature groups, partition data into partitions equal to "num_groups".
3     f_sel = [ ]    % array to store the selected feature from all partitions.
4     for $i = 1$ to $m$ do
5         $sel_i$ = select_top_ranked($P[i]$, $t$, rank_func)    % Select $t$ top-ranked features from partition $P[i]$ using mutual information as ranking function
6         $sel_i$ = sort($sel_i$)    % sort (descending) the selected feature w.r.t MI rank.
7         $f\_sel = f\_sel \bigcup sel_i$    % take union of selected features from all partitions
8     end
9     return f_sel    % selected features
```

3.2. Stage-II

Stage-II of the proposed method, as depicted in Fig. 3, adopts PSO as a wrapper-based feature selection method that utilizes Regularized Logistic Regression as a wrapped classification algorithm. To build good behavior profiles for goodware and ransomware that may improve binary and multi-class classification performance, PSO aims to optimize the selection of a suitable

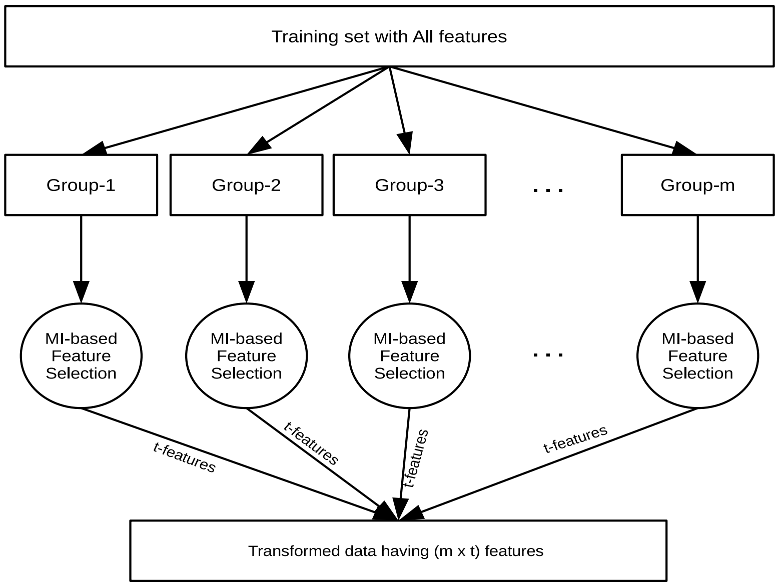

Training set with All features
Group-1
Group-2
Group-3
...
Group-m
MI-based Feature Selection
t-features
Transformed data having (m x t) features

Fig. 2. Stage-I of the proposed method.

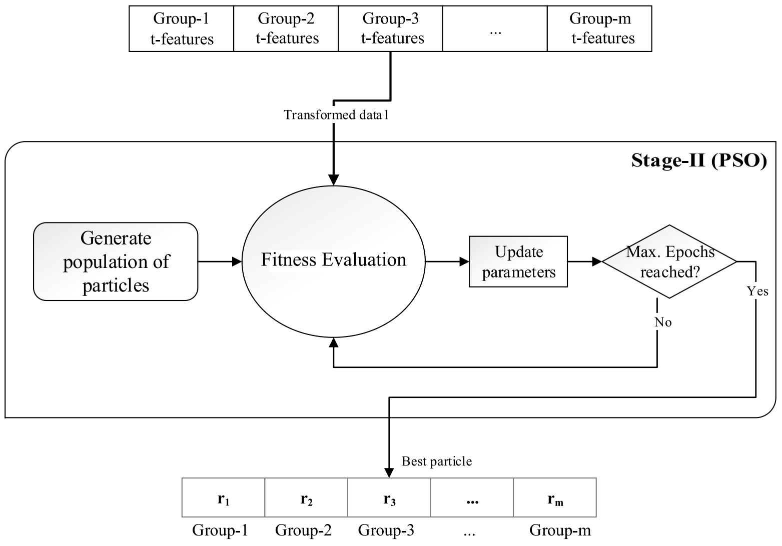


Fig. 3. Stage-II: PSO-based feature selection.

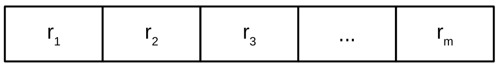


Fig. 4. The PSO particle representation in the proposed method.

number of features from each feature group using the ratios from a PSO particle. For a better understanding of the method, the pseudo code of Stage-II and the fitness calculation process used at this stage are presented in Algorithms 2, and 3, respectively.

Each particle of the proposed method comprises m continuous values, as depicted in Fig. 4, each of which corresponds to a feature group in the data set. The value of every element of the vector is labeled as r_i, where $i \in \{1, 2, \dots, m\}$. Each r_i variable has a continuous value ranging from 0.0 to 1.0. The objective of PSO at this stage is to minimize the classification error. Hence, the fitness function is

$$Fitness = 1 - Accuracy, \tag{1}$$

$$Accuracy = \frac{1}{n}\sum_{i=1}^{n}\frac{correct_i}{total_i} \tag{2}$$

where n represents the total number of classes in the data set, and $correct_i$ are correctly classified instances from a total number of instances represented as $total_i$ of class i. Due to disparity in the number of samples among various classes, we used balanced accuracy to calculate the classification error.

Stage-II calculates the fitness value using the transformed training data-I. This is split into sub-training and sub-test sets as shown in Fig. 5 during the evolutionary process, where j_i top-ranked features are selected using MI from each feature group i. The value j_i is calculated as

$$j_i = r_i \times t_i. \tag{3}$$

The selected features from all m groups are used to transform sub-training and sub-test data used for training and evaluating the wrapper classifier, respectively.

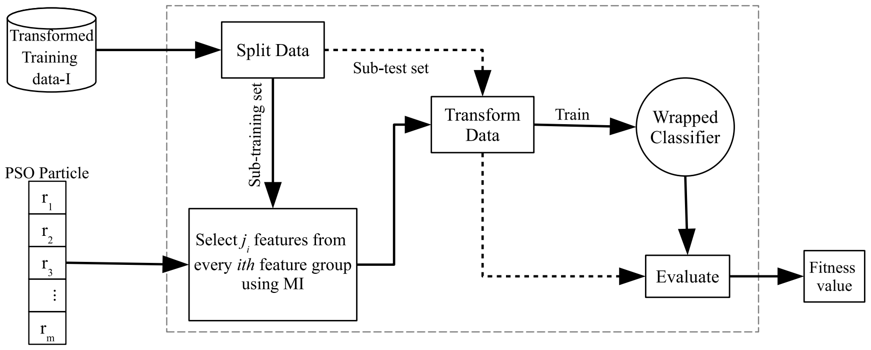


Fig. 5. The process of calculating the fitness value at Stage-II of the proposed method.

Algorithm 2: Stage-II of the proposed method.

```
1  Function Stage-II(data, swarm_size, ind_dim, opt_iter):
2      individual = [dim]     % PSO individual: an array of size
         dim that can hold float values in the range of (0, 1).
3      swarm = gen_swarm(individual, swarm_size)     %
         Randomly generate a swarm of individuals of size
         swarm_size.
4      for iter = 1 to opt_iter do
5          for j = 1 to swarm_size do
6              fitness_val, $f\_sel_2$ = calc_fitness(swarm[j],
                 $train\_data_{trans\_1}$)    % Calculate fitness value
                 and get selected features.
7              if fitness_val < p_best then
8                  update p_best     %update personal best
                     positions and selected features with $f\_sel_2$.
9          end
10         if p_best < g_best then
11             update g_best     %update swarms best known
                 position and features with p_best.
12     end
13     return g_best     %output global best positions and the
         best selected features;
```

Algorithm 3: Fitness value calculation process at Stage-II of the proposed method

```
1  Function calc_fitness(pso_ind, data):
2      $m \leftarrow len(pso\_ind)$
3      $P[\ ] = partition(d, m)$     % Based on the feature groups,
         partition data into partitions equal to "num_groups".
4      $f\_sel_{internal} = [\ ]$     %array to store the selected feature
         from all partitions.
5      for $i = 1$ to $m$ do
6          $r_i \leftarrow pso\_ind[i]$     % ratio: ith value of pso
             individual.
7          $t_i \leftarrow len(P[i])$     % total number of feature of ith
             internal partitions.
8          $j_i = r_i \times t_i$     % number of features to be selected
             from ith partition.
9          $sel_i = select\_top\_ranked(P[i], j_i)$     %Select first $j_i$
             features from partition $P[i]$.
10         $f\_sel_{internal} = f\_sel_{internal} \bigcup sel_i$     %take union of
             selected features for each partition.
11     end
12     $sub\_train\_data_{trans}$ = transform(sub_train_data,
         $f\_sel_{internal}$ )     % transformed data using selected
         features.
13     $sub\_test\_data_{trans}$ = transform(sub_test_data,
         $f\_sel_{internal}$ )
14     train_wrapped_clf($sub\_train\_data_{trans}$)
15     accuracy = evaluate_wrapped_clf($sub\_test\_data_{trans}$)
16     fitness_val = 1-accuracy     % calculate fitness value.
17     return (fitness_val, $f\_sel_{internal}$)     % return fitness
         value and selected features
```

4. Experimental design

This section discusses the experimental setup that includes the details of the data set. Moreover, for reproducibility of this study's results, the experiments' implementation details, including the configuration of the evolutionary parameters, are provided in this section. Performance evaluation metrics and the statistical significance test used in this study are also discussed at the end of this section.

4.1. Data set

We used the Resilient Information Systems Security (RISS)[3] [26] data set in our experiments. A break-up of the number of samples of ransomware families is given in Table 1. No details could be found about which goodware samples were specifically included in the analysis. A group-wise break-up of the number of features of the data set is presented in Table 2. The data set is in binary format, where every feature has a value 0 or 1 representing the absence or presence of the corresponding feature in an instance.

[3] Available at: http://rissgroup.org/ransomware-dataset/

Table 1
A summary of the data set instances [26].

Family	# Instances	Family	# Instances
Citroni	50	Matsnu	59
CryptoLocker	107	Pgpcoder	4
CryptoWall	46	Reveton	90
Kollah	25	TeslaCrypt	6
Kovter	64	Trojan-Ransom	34
Locker	97	Goodware	942

4.2. Implementation

Five different classification algorithms are used in our experiments, namely, Regularized Logistic Regression (RLR), Random

Table 2
Groups of features in the data set.

Group ID	Description	# Features
API	API invocations	232
DROP	Extensions of the dropped files	346
REG	Registry key operations	6622
FILE	File operations	4141
FILES_EXT	Extension of the files involved in file operations	935
DIR	File directory operations	2424
STR	Embedded strings	16267

Forest (RF), Decision Tree (DT), Support Vector Machine (SVM) with linear kernel, and K-nearest Neighbor Classifier (KNN). All of the classification algorithms mentioned above were used with default parameter settings, and no parameter tuning was performed. It is worth mentioning that RLR is used as the wrapped classification algorithm in our model. The rest of the classification algorithms were used to investigate the genarlizability of the selected features on other types of classification algorithms than the wrapped one.

Python 3.7 is used for the implementation of experiments in this study. Scikit-learn libraries [60] version 0.21.3 are used for implementation of the classification algorithms, whereas the Pyswarms [61] library version 1.3 is used for PSO implementation. The source codes of two feature selection methods used for benchmarking, i.e., VarLenPSO [47] and SaPSO [49], were obtained from the corresponding authors. VarLenPSO was implemented in Java, whereas, SaPSO was implemented in Matlab. These methods used swarm intelligence for feature selection on high-dimensional data and have shown good performance on various data sets.

In our experiments, we used k-fold cross-validation (where $k = 4$) to split the data into training and test sets in the outer loop of the experiments. The rationale of using $k = 4$ is based on the number of instances in the *PgpCoder* family of ransomware as it has only four samples. We, therefore, decided to split the data into four folds to ensure the representation of each malware family in all folds (stratified split).

4.3. The evolutionary parameters

The evolutionary parameters of the proposed method are set based on the recommendations given in [62]. Thus, in the proposed method, the cognitive and social parameters ($c1$ and $c2$) were set as 1.149618, inertia (w) as 0.729844, and swarm size as 30. The evolutionary parameters (and any algorithm-specific parameters) of VarLenPSO and SaPSO are set according to the recommendations of the corresponding authors. All the methods, i.e., the proposed method, VarLenPSO, and SaPSO, were run 30 times independently, where each run consisted of 100 epochs.

During the evolutionary process, the data (transformed training data-I) is split further into 3-folds. This split is because the number of samples of *PgpCoder* in the *training* set at this stage was only three. Splitting the data into 3-folds ensures the representation of this family in all folds on the one hand, while it avoids overfitting on the other hand by training and evaluating the wrapper classifier on non-overlapping subsets of data.

4.4. Performance evaluation metrics and benchmarking

The data set we used for experimentation is unbalanced concerning the number of instances of goodware and ransomware. Furthermore, the number of instances of different families of ransomware are varying such that one of the ransomware families has only four instances. Hence, balanced accuracy is used (given in Eq. (2)) for reporting the results in both binary as well as multi-class classification instead of the typical accuracy rate generally used for this purpose.

The training data was used for feature selection at Stage-I and Stage-II of the proposed method to free our experiments from feature selection bias. Finally, the training set was utilized for training the classification algorithms, which were evaluated with unseen (test) data.

The reported accuracy of the baseline method is the average accuracy of all folds obtained from running the method once for each fold. Whereas, for all PSO-based methods, the average accuracy for all folds for a particular run has been considered as the output of that single run. Hence, the average of the outputs for the 30 independent runs is reported in the results table (next section).

The significance of the results is reported as the proposed method in comparison with the other individual methods. The Baseline method is deterministic, and the proposed method is stochastic. We, therefore, used one sample two-tailed t-test for significance calculation of the Baseline method against the proposed method. However, an unpaired two-tailed t-test (being all three methods stochastic) is performed to find the proposed method's significance against VarLenPSO and SaPSO. A confidence interval of 95% is used in these tests. The annotation used for the significance of results is as follows. "↑" with a result indicates significantly better, "=" is used to indicate no significant difference, and the "↓" sign shows significantly worse results than the proposed method.

5. Results and discussions

The classification results using all four methods are presented in Table 3. The results obtained using various feature selection methods are listed in the columns with corresponding names of features selection methods. The reported results are percentage accuracy scores. The best accuracy scores among all four feature selection methods for each classification algorithms are shown in **bold** fonts. The results related to Ransomware detection (binary classification) are discussed in Section 5.1, whereas the results of the multi-class classification are discussed in Section 5.2. Furthermore, an analysis of the number of features selected by the different methods is presented in the last subsection of this Section.

5.1. Results of ransomware detection

The experimental results of the binary classification task indicate that the proposed method produced significantly better accuracy with RLR (97.33%), compared to the Baseline method with 97.21% and the VarLenPSO with 96.88% average accuracy. However, the performance of the proposed method is comparable to that of SaPSO, which achieved 97.39% average accuracy. RLR with any of the PSO-based feature selection methods produced a better accuracy score than the other classification algorithms used in this study. However, with the Baseline method, RF outperformed the other classification algorithms with an average accuracy of 97.48%.

In summary, out of the total 15 comparisons, the proposed method showed poor performance in 4 cases, comparable performance in 4 cases, and significantly better performance in 7 cases. These results show that the overall performance of the proposed method for binary classification is better on average than the other benchmark methods used in this study.

Table 3
Percentage of balanced accuracy ($\bar{x} \pm s$) for binary and multi-class classification with features selected using the Baseline, SaPSO, varLenPSO and the proposed methods.

Task	Classifier	Baseline	VarLenPSO	SaPSO	Proposed method
Binary	RLR	$97.21 \downarrow$	$96.88 \pm 0.26 \downarrow$	$\mathbf{97.39 \pm 0.32} =$	97.33 ± 0.10
	RF	$\mathbf{97.48} \uparrow$	$96.24 \pm 0.39 \downarrow$	$96.09 \pm 0.26 \downarrow$	97.06 ± 0.21
	DT	$95.14 =$	$95.54 \pm 0.48 \uparrow$	$\mathbf{96.19 \pm 0.54} \uparrow$	95.03 ± 0.33
	SVM	$\mathbf{96.62} \uparrow$	$95.84 \pm 0.39 \downarrow$	$96.32 \pm 0.37 =$	96.33 ± 0.20
	KNN	$93.45 \downarrow$	$94.03 \pm 0.41 =$	$93.32 \pm 0.27 \downarrow$	$\mathbf{94.18 \pm 0.21}$
Multi-class	RLR	$53.19 \downarrow$	$52.54 \pm 1.64 \downarrow$	$\mathbf{55.70 \pm 0.75} \uparrow$	54.84 ± 0.37
	RF	$\mathbf{55.82} \uparrow$	$54.17 \pm 1.30 =$	$54.17 \pm 0.80 \downarrow$	54.68 ± 0.90
	DT	$53.31 =$	$51.77 \pm 0.99 \uparrow$	$52.80 \pm 1.65 =$	$\mathbf{53.33 \pm 1.39}$
	SVM	$52.84 \downarrow$	$54.36 \pm 1.31 =$	$\mathbf{55.72 \pm 0.80} \uparrow$	54.56 ± 0.47
	KNN	$48.56 =$	$\mathbf{48.89 \pm 0.87} =$	$47.30 \pm 1.43 \downarrow$	48.36 ± 1.67

Table 4
Class-wise average accuracy (%) with proposed method.

Class	Classification accuracy (%)				
	RLR	RF	DT	SVM	KNN
Goodware	99.19	99.83	96.51	98.03	95.80
Critroni	70.17	73.06	69.88	74.16	64.50
CryptLocker	68.32	69.59	64.57	68.12	68.02
CryptoWall	52.70	45.92	50.19	49.28	49.54
KOLLAH	44.31	42.70	38.73	45.12	39.52
Kovter	76.82	75.16	56.98	66.67	57.97
Locker	58.37	59.51	54.21	56.82	49.09
MATSNU	44.83	46.37	48.71	46.29	34.27
PGPCODER	0.00	0.00	0.00	0.00	0.00
Reveton	75.45	76.83	76.75	74.39	63.99
TeslaCrypt	37.50	40.83	50.83	37.50	43.33
Trojan-Ransom	30.46	26.32	32.58	38.40	14.28

5.2. Results of ransomware classification

For multi-class classification, although the proposed method (54.84%) could not achieve adequate performance as compared to SaPSO (55.70%), it significantly outperformed the Baseline (53.19%) and VarLenPSO (52.54%) as shown in Table 3. Similarly, the results for the other classification algorithms show a variable trend with all feature selection methods used in this study. In summary, the proposed method has achieved poor performance in 4 cases and comparable performance in 6 cases; it significantly outperformed the other methods in 5 cases out of the total 15 comparisons for multi-class classification task shown in Table 3.

It can be observed from the results presented in Table 3 for multi-class classification that the overall classification accuracy with all the classification algorithms is in the range of 48.00% through 56.00%. To provide a deeper insight into multi-class classification accuracy, Table 4 presents the average accuracy achieved using the features selected by the proposed method for each class (given in the first column) with all five classification algorithms. It can be observed that the classification accuracy achieved for goodware is more than 96.00% with all classification algorithms. However, the classification performance for various families of ransomware with different classification algorithms shows a variable pattern. The accuracy on the *Pgpcoder* family of ransomware, for instance, remained consistently 0.00% with all classification algorithms. It had only 3 instances in the training set (2 in the sub-training set during the evolutionary process), which might not be sufficient to train the classifier. Other ransomware families with low classification accuracy include *Trojan-Ransom*, *Matsnu*, and *Kollah*.

In summary, the proposed method showed a mixed performance trend compared to the other state-of-the-art feature selection methods used in this research. Empirically, it was observed that the performance of the proposed method in both ransomware detection and classification was generally better where the classification algorithm was the same as the wrapped classifier (RLR in our case).

Table 5
Number of features ($\bar{x} \pm s$) selected by all methods.

Feature selection methods	Binary	Multi-class
Baseline	400	400
VarLenPSO	13928 ± 677	11476 ± 355
SaPSO	25496 ± 1978	20785 ± 857
Proposed method	823 ± 59	646 ± 40

6. Further analysis

We conducted various analyses to provide insight into the proposed method, including a discussion on the feature reduction achieved by the methods used in this stuy, the algorithm's convergence, group-wise ratios of the selected features, and discussion on the selected features and their selection frequency.

6.1. Analysis on the number of features selected

The number of features selected by the four methods is presented in Table 5. It is evident from the results that the Baseline method selects the least number of features, i.e., 400, as compared to the proposed method, VarLenPSO, and SaPSO. However, human intervention is required in the Baseline method to decide the number of features that produce the best results. In contrast, the proposed method, VarLenPSO, and SaPSO are automated in selecting a suitable number of features and therefore do not require any human involvement. For binary classification, on average, the proposed method selected 2.66%, i.e., 823 features, in contrast to VarLenPSO and SaPSO that selected 44.98% and 83.79% of the total number of features (30,967), respectively.

Similarly, for multi-class classification, the average number of features selected by the proposed method, VarLenPSO, and SaPSO, respectively, were 2.08%, 37.06%, and 67.12% of the total number of the features. This shows that the proposed method achieved a significantly better reduction in data dimensionality than that of SaPSO and VarLenPSO.

6.2. Convergence

The convergence of the proposed method for binary and multi-class classification is shown in Fig. 6. Each point in the graph represents the average fitness value of 30 runs for all folds per epoch, whereas the whiskers represent the standard deviation. It is evident from the convergence graphs that the algorithm initially explores the search space for the first few epochs in both binary and multi-class classification. Afterward, the algorithm starts exploitation and continues for the rest of the evolutionary process. The whiskers in the convergence plot indicate that the algorithm is stable among various runs.

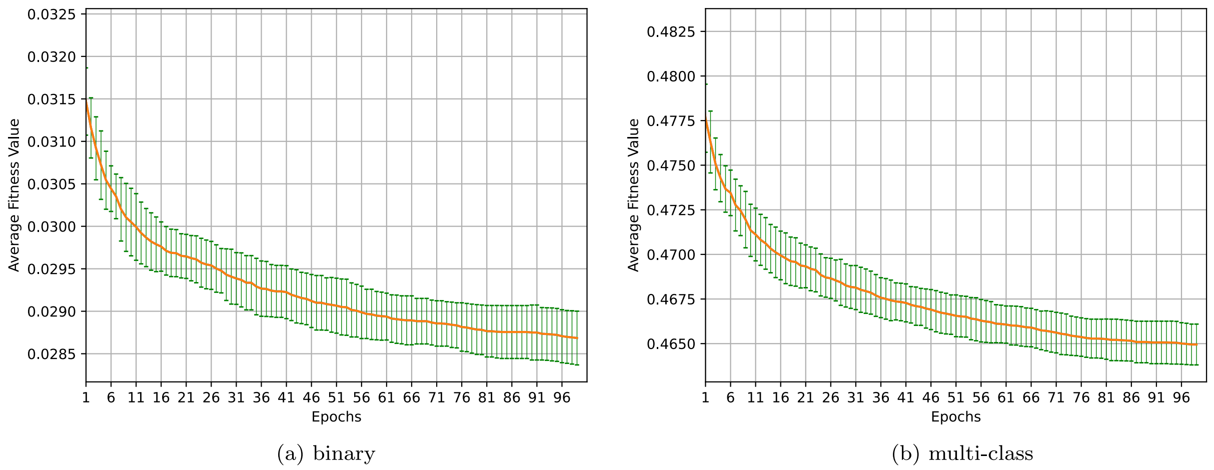


Fig. 6. The convergence graphs of the proposed method generated as the average fitness value per epoch.

6.3. Group-wise ratios of selected features

On average, the proposed feature selection method selected 822 ± 59 and 803 ± 49 features for ransomware detection and classification, respectively. The proposed method uses PSO to select an appropriate number of features from various feature groups of the data set to improve the classification performance; it is worthwhile to analyze the ratios of features selected from the different feature groups.

A comparison of the ratios of selected features from various feature groups by the proposed method and the baseline method is shown in Figs. 7(a) and 7(b) for binary and multi-class classification, respectively. The ratios are stated in the percentage of the average number of features selected from each feature group.

Both methods have selected more features from the API, REG, and STR groups than the other four groups for binary classification. With the baseline method, the top group is REG (48.06%), API and STR are second and third large feature groups with ratios 23.88%, and 8.88%, respectively. However, with the proposed method, the top three contributing feature groups in the case of binary classification are the same as the baseline method with changed positions of REG and API. Here, API emerges as the top contributing group with a ratio of 22.31%, while REG and STR stay at second and third positions with a contribution of 18.34% and 12.95%, respectively.

For multi-class classification, the baseline method follows the same pattern of feature selection as in the case of binary classification, where the top three features contributing groups are REG, API, and STR. The features share from REG, API, and STR are 39.44%, 29.44%, and 10.38%, respectively. In contrast, the proposed method selected most of the features (22.72%) from STR, then 16.41% from FILES-EXT, and 16.13% from API out of the total selected features.

These results validate our hypothesis that selecting the appropriate number of features from various groups impacts classification accuracy.

6.4. Discussion on selected features

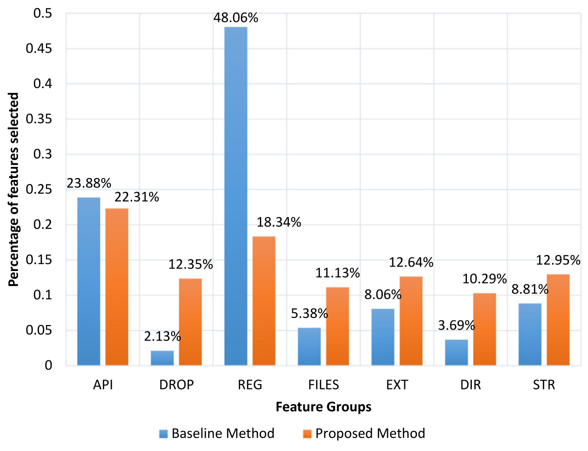
We calculated the selection frequency of individual features. The features selected by the particles with the best fitness value at the end of the evolutionary process were considered here. As four-fold cross-validation is used in the outer loop of the experiments, the total number of runs is equal to 120(= 30 × 4). Hence, the frequency for a feature ranges between 1 (only selected in one run) and 120 (selected in all the runs). The total number of unique features selected by the proposed method is 1,923 in the case of binary classification and 1,797 features for multi-class classification.

6.4.1. Selected features for ransomware detection

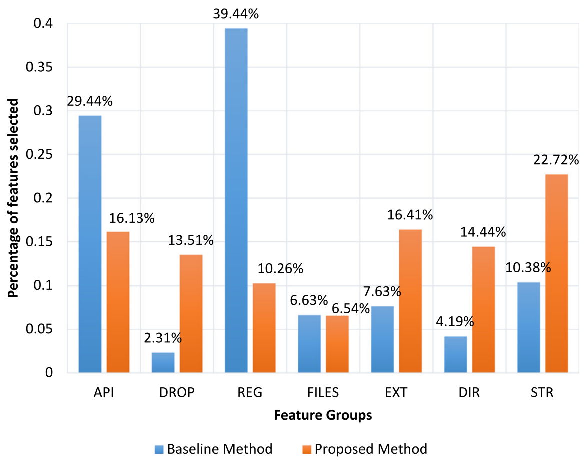
For binary classification, the total number of the features that were always selected, i.e., with selection frequency equal to 120, is 222. The break-up of these features (as shown in Fig. 8) shows that, apart from five features, all of these features came from three feature groups, namely API, REG, and STR with a share of 132, 57, and 28 features, respectively.

The total number of selected features from the API group is 231, from which 132 features are selected every time the method was run, while the rest of the 99 features are selected once or more but not always. The results show that features from the API group played a vital role in connection with binary classification. Analyzing the 132 features selected in all the runs from the API group reveals that these features belong to various groups created based on their functionality [63], where some interesting ones will be discussed here.

Some of these API calls, e.g., `GetNativeSystemInfo`, `GetSystemInfo`, and `GetSystemWindowsDirectoryA` belong to `sysinfoapi` header which are used for system services, such as collecting information about the system. These API calls can be used by any software (goodware or malware) to retrieve information about the host machine. A few other API calls come from the header `Winuser` which is used by `WindowsContrls`. For instance, `GetCursorPos` function is used to retrieve the coordinates of cursor position at a certain time, which may be used to record user activity by malware running on the system. Another important API header is `Winbase` which is used to develop backup in Windows operating systems. A certain number of API calls like `GetComputerNameA`, `FindResourceA`, and `LookupPrivilegeValueW` come from the same group. A signature header for some of the ransomware families that use OS-provided cryptographic libraries is `Wincrypt` header. This header share a part of features selected with maximum frequency from API group and contains functions like `CryptAcquireContextW`, `CryptEncrypt`, `CryptExportKey`, and `CryptGenKey`. Windows standard cryptographic libraries may be used by compression and encryption tools, hence producing false positives for a ransomware detection method relying on tracing the use of these libraries alone. A good number of other API calls, e.g., `CreateDirectoryW`, `GetDiskFreeSpaceExW`, and `FindFirstFileExW` belong to `Fileapi` header which are used for file and directory operations. There is no surprise in having API calls from `Fileapi` header with maximum selection frequency as



(a) binary



(b) multi-class

Fig. 7. The percentage of average number of features selected from each group by the baseline and the proposed methods.

ransomware is a data-centric malware. Moreover, goodware also needs to create directories and files in their installation phase. Yet many other API calls like `RegCreateKeyExA`, `RegOpenKeyExA`, and `RegQueryValueExA` are related to registry key operations that are from the `Winreg` header.

REG is the second largest group, with 57 features were selected in each of the 120 runs. The Operation-wise break-up of these features is given here: 35 keys were associated with the "open" operation, 19 keys with "read", and 3 keys with "write" operation. For the sake of brevity in mentioning the registry keys, we used abbreviations for key hive names, i.e., HKEY_LOCAL_MACHINE (HKLM), HKEY_CURRENT_USER (HKCU), and HKEY_CLASSES_ROOT (HKCR).

All of the three selected registry keys associated with write operation have been marked as indicative of ransomware by Hybrid Analysis.[4] The format of these three keys is given as `HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\MountPoints2\e7136b33-a421-11e5-b597-806d6172696f`, which is common among three keys but with a difference of the number at the end of the key. The 19 registry keys associated with the read operation come from three hives, i.e., `HKLM\SOFTWARE`, `HKLM\SYSTEM`, and `HKCU\Software`. Most of the registry keys accessed for reading were used to retrieve information about the machine and the current user, which is evident from the keyword "current version" in the hive structure. Registry keys that were opened mostly reflect that these keys are used for information retrieval about the machine and the current user, such as the computer's name, mount points for drives, Explorer settings, and TCP/IP parameter settings. One interesting key among registry keys opened is `HKLM\SOFTWARE\Classes\Directory\ShellEx` which is used for setting programs to run (or not to) at Windows startup.

STR is the third largest group with 28 features that were selected in each of the 120 runs, all of which are the names of dynamic link library (DLL) files except one, i.e., `SHAutoComplete`. The names of DLL files include, but are not limited to, `kenel32.dll`, `shell32.dll`, `user32.dll`, and `msi.dll`. The features (with maximum frequency of selection) from other groups include `tmp` from DROP group (this group contains extension of the dropped files), `tmp` from FILES_EXT (for opening a file with extension ".tmp"), and `C:\Documents and Settings\MyUser\Local Settings\Start Menu` and `C:\Documents and Settings\MyUser\Local Settings \Temp` from FILES group.

[4] A free online malware analysis tool that uses Falcon sandbox for automated analysis of submitted files.

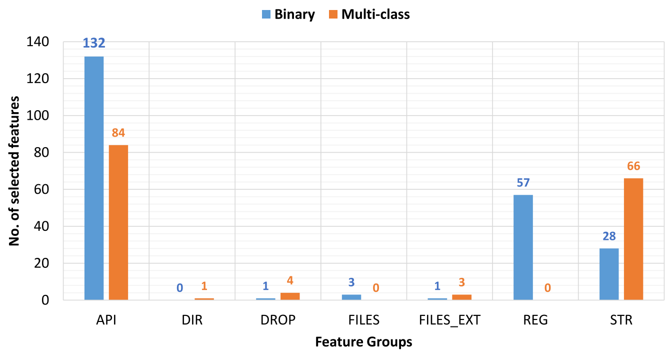


Fig. 8. Group-wise break-up of features with maximum selection frequency.

6.4.2. Selected features for ransomware classification

For the multi-class classification task, the total number of features with maximum selection frequency is 158, and the break-up of these features is shown in Fig. 8.

API is the top contributing group with 84 features, out of which 76 features are the same that are selected 100% of the time for binary classification task. Remaining 8 features from API group are listed as `FindWindowW`, `GetShortPathNameW`, `GetFileAttributesExW`, `getaddrinfo`, `NtEnumerateValueKey`, `GetAsyncKeyState`, `NtOpenSection`, and `ShellExecuteExW`. The second largest group among these features is STR with 66 features. The other groups have a share of 4, 3, and 1 from DROP, FILES_EXT, and DIR, respectively, whereas FILES and REG groups have no representation in these features.

As is the case with binary classification, most of the selected features from this group are names of DLL files. A few names of these DLL files appeared more than once due to change of letter case, for instance `kernel32.dll` appeared five times, `advapi32.dll` and `usr32.dll` occurred thrice, whereas, `ole32.dll`, `sell32.dll`, `comctl32.dll`, `oleaut32.dll`, `msvcrt.dll`, and `wininet.dll` appeared twice in these features. The repetition highlights the fact that strings in the data set are case-sensitive. A better way to avoid selecting such features could be to make all strings either lower or upper case. Enforcing a single case might help reduce the features at the initial stages and prevent any recurring features in results. Only 10 out of 66 above mentioned strings are other than names of

DLL files, where 7 of these strings are related to cryptography, for instance, `CryptAcquireContextA`, `CryptAcquireContextW`, and `CryptReleaseContext`, which can be indicative of ransomware activity in a system.

7. Conclusions

A two-stage PSO-based wrapper feature selection method for ransomware detection and classification is proposed in this study. The proposed method selects an optimal number of features from various groups to improve the detection and classification performance of ransomware, goodware, and various ransomware families. Compared to the Baseline method, the proposed method underperformed or achieved comparable performance in some cases while significantly outperforming the Baseline method in other cases. In contrast to the Baseline method, the proposed method automatically selects an appropriate number of features for binary and multi-class classification tasks. Compared to VarLenPSO and SaPSO, the proposed method performed comparable or significantly better in most cases.

The experimental results indicate that the proposed method reduced 97.33% of the original features on average for binary classification. In contrast, other state-of-the-art feature selection methods, i.e., VarLenPSO and SaPSO, achieved a reduction of 55.02% and 16.21% features on average. For multi-class classification, the feature reduction achieved using the proposed method, VarLenPSO, and SaPSO is 97.92%, 62.94%, and 32.88%, respectively. These results show that the proposed method produced a comparable performance with fewer features than the contemporary methods. Finally, a detailed analysis of features selected by the proposed method provides insight into these features. The data set was observed to contain redundant features, such as recurring ones, due to case sensitivity that preprocessing data could reduce.

Future work to this study may consider incorporating more feature groups into the data set to capture additional behavior such as communication with key servers or C&C. Also, some of the features groups may be ruled out from the data to check on the performance changes. Furthermore, the sequences of the system calls as additional features to help distinguish between ransomware families may be considered.

CRediT authorship contribution statement

Muhammad Shabbir Abbasi: Conceptualization, Methodology, Software, Investigation, Data curation, Validation, Writing – original draft, Visualization. **Harith Al-Sahaf:** Conceptualization, Methodology, Validation, Resources, Writing – review & editing, Supervision. **Masood Mansoori:** Writing – review & editing. **Ian Welch:** Resources, Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

Supported by the Cyber Security Research Programme "AI for Automating Response to Threats" Ministry of Business, Innovation, and Employment (MBIE), New Zealand, part of Catalyst Strategy Funds under Grant *MAUX1912*.

References

[1] A. Kharraz, W. Robertson, D. Balzarotti, L. Bilge, E. Kirda, Cutting the gordian knot: A look under the hood of ransomware attacks, in: Proceedings of the International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, Springer, 2015, pp. 3–24.

[2] Monika, P. Zavarsky, D. Lindskog, Experimental analysis of ransomware on windows and android platforms: Evolution and characterization, Procedia Comput. Sci. 94 (2016) 465–472.

[3] R. Brewer, Ransomware attacks: Detection, prevention and cure, Netw. Secur. 2016 (9) (2016) 5–9.

[4] D. Maiorca, F. Mercaldo, G. Giacinto, C.A. Visaggio, F. Martinelli, R-PackDroid: API package-based characterization and detection of mobile ransomware, in: Proceedings of the Symposium on Applied Computing, ACM, 2017, pp. 1718–1723.

[5] D. Nieuwenhuizen, A Behavioural-Based Approach to Ransomware Detection, MWR Labs Whitepaper, 2017, URL https://labs.mwrinfosecurity.com/publications/a-behavioural-based-approach-to-ransomware-detection/.

[6] N. Hampton, Z. Baig, S. Zeadally, Ransomware behavioural analysis on windows platforms, J. Inf. Secur. Appl. 40 (2018) 44–51.

[7] D.Y. Huang, M.M. Aliapoulios, V.G. Li, L. Invernizzi, E. Bursztein, K. McRoberts, J. Levin, K. Levchenko, A.C. Snoeren, D. McCoy, Tracking ransomware end-to-end, in: Proceedings of the 39th IEEE Symposium on Security and Privacy, IEEE, 2018, pp. 618–631.

[8] S. Homayoun, A. Dehghantanha, M. Ahmadzadeh, S. Hashemi, R. Khayami, Know abnormal, find evil: Frequent pattern mining for ransomware threat hunting and intelligence, IEEE Trans. Emerg. Top. Comput. 8 (2) (2017) 341–351.

[9] J.D. Groot, A History of Ransomware Attacks: The Biggest and Worst Ransomware Attacks of All Time, Digital Guardian, 2019, URL https://digitalguardian.com/blog/history-ransomware-attacks-biggest-and-worst-ransomware-attacks-all-time, last accessed on 04-02-2019.

[10] S. Mohurle, M. Patil, A brief study of wannacry threat: Ransomware attack 2017, Int. J. Adv. Res. Comput. Sci. 8 (5) (2017) 1938–1940.

[11] A.L. Young, M. Yung, Cryptovirology: The birth, neglect, and explosion of ransomware, Commun. ACM 60 (7) (2017) 24–26.

[12] R. Richardson, M. North, Ransomware: Evolution, mitigation and prevention, Int. Manag. Rev. 13 (1) (2017) 10–21,101, URL https://search.proquest.com/docview/1881414570?accountid=14782.

[13] N. Kshetri, J. Voas, Do crypto-currencies fuel ransomware? IT Prof. 19 (5) (2017) 11–15.

[14] B.A.S. Al-rimy, M.A. Maarof, S.Z.M. Shaid, Ransomware threat success factors, taxonomy, and countermeasures: A survey and research directions, Comput. Secur. 74 (2018) 144–166.

[15] B. Dobran, 27 Terrifying ransomware statistics & facts you need to read, 2019, URL https://phoenixnap.com/blog/ransomware-statistics-facts, last accessed on 27-07-2020.

[16] verizon Enterprise, 2018 Data Breach Investigations Report: The Year of Ransomware, Investigation Report, Verizone Enterprise, 2018, URL https://enterprise.verizon.com/resources/reports/dbir/, last accessed on 06-02-2019.

[17] PurpleSec, The growing threat of ransomware, 2020, URL https://purplesec.us/resources/cyber-security-statistics/ransomware/, last accessed on 01-04-2021.

[18] A. Souri, R. Hosseini, A state-of-the-art survey of malware detection approaches using data mining techniques, Hum. Centric Comput. Inf. Sci. 8 (1) (2018) 1–22.

[19] H. Daku, P. Zavarsky, Y. Malik, Behavioral-based classification and identification of ransomware variants using machine learning, in: Proceedings of the 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/12th IEEE International Conference on Big Data Science and Engineering, IEEE, 2018, pp. 1560–1564.

[20] Y.A. Ahmed, B. Koçer, S. Huda, B.A.S. Al-rimy, M.M. Hassan, A system call refinement-based enhanced minimum redundancy maximum relevance method for ransomware early detection, J. Netw. Comput. Appl. 167 (2020) 102753.

[21] A. Kharaz, S. Arshad, C. Mulliner, W. Robertson, E. Kirda, UNVEIL: A large-scale, automated approach to detecting ransomware, in: Proceedings of the 25th USENIX Security Symposium, USENIX, 2016, pp. 757–772.

[22] N. Scaife, H. Carter, P. Traynor, K.R.B. Butler, Cryptolock (and drop it): Stopping ransomware attacks on user data, in: Proceedings of the 36th International Conference on Distributed Computing Systems, IEEE, 2016, pp. 303–312.

[23] G. Cusack, O. Michel, E. Keller, Machine learning-based detection of ransomware using SDN, in: Proceedings of the 2018 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization, ACM, 2018, pp. 1–6.

[24] K. Cabaj, P. Gawkowski, K. Grochowski, D. Osojca, Network activity analysis of CryptoWall ransomware, Prz. Elektrotech. 91 (11) (2015) 201—204.

[25] Z.-G. Chen, H.-S. Kang, S.-N. Yin, S.-R. Kim, Automatic ransomware detection and analysis based on dynamic API calls flow graph, in: Proceedings of the International Conference on Research in Adaptive and Convergent Systems, ACM, 2017, pp. 196–201.
[26] D. Sgandurra, L. Muñoz-González, R. Mohsen, E.C. Lupu, Automated dynamic analysis of ransomware: Benefits, limitations and use for detection, Comput. Res. Repos. (2016) arXiv:1609.03020.
[27] B. Kolosnjaji, A. Zarras, G. Webster, C. Eckert, Deep learning for classification of malware system call sequences, in: Proceedings of the Australasian Joint Conference on Artificial Intelligence, Springer, 2016, pp. 137–149.
[28] T.-S. Chou, K.K. Yen, J. Luo, Network intrusion detection design using feature selection of soft computing paradigms, Int. J. Comput. Intell. 4 (3) (2008) 196–208.
[29] Y.A. Ahmed, B. Kocer, B.A.S. Al-rimy, Automated analysis approach for the detection of high survivable ransomware, KSII Trans. Internet Inf. Syst. 14 (5) (2020) 2236–2257.
[30] J. Cai, J. Luo, S. Wang, S. Yang, Feature selection in machine learning: A new perspective, Neurocomputing 300 (2018) 70–79.
[31] I. Guyon, A. Elisseeff, An introduction to variable and feature selection, J. Mach. Learn. Res. 3 (2003) 1157–1182.
[32] H. Liu, H. Motoda, Computational Methods of Feature Selection, CRC Press, 2007.
[33] J. Li, K. Cheng, S. Wang, F. Morstatter, R.P. Trevino, J. Tang, H. Liu, Feature selection: A data perspective, ACM Comput. Surv. 50 (6) (2017), 1–45.
[34] B. Xue, M. Zhang, W.N. Browne, X. Yao, A survey on evolutionary computation approaches to feature selection, IEEE Trans. Evol. Comput. 20 (4) (2016) 606–626.
[35] K. Mistry, L. Zhang, S.C. Neoh, C.P. Lim, B. Fielding, A micro-GA embedded PSO feature selection approach to intelligent facial emotion recognition, IEEE Trans. Cybern. 47 (6) (2017) 1496–1509.
[36] A. Goltsev, V. Gritsenko, Investigation of efficient features for image recognition by neural networks, Neural Netw. 28 (2012) 15–23.
[37] M.A. Ambusaidi, X. He, P. Nanda, Z. Tan, Building an intrusion detection system using a filter-based feature selection algorithm, IEEE Trans. Comput. 65 (10) (2016) 2986–2998.
[38] A. Feizollah, N.B. Anuar, R. Salleh, A.W.A. Wahab, A review on feature selection in mobile malware detection, Digit. Investig. 13 (2015) 22–37.
[39] L. Wang, Y. Wang, Q. Chang, Feature selection methods for big data bioinformatics: A survey from the search perspective, Methods 111 (2016) 21–31.
[40] S. Huda, J. Yearwood, H.F. Jelinek, M.M. Hassan, G. Fortino, M. Buckland, A hybrid feature selection with ensemble classification for imbalanced healthcare data: A case study for brain tumor diagnosis, IEEE Access 4 (2016) 9145–9154.
[41] H. Liu, L. Yu, Toward integrating feature selection algorithms for classification and clustering, IEEE Trans. Knowl. Data Eng. 17 (4) (2005) 491–502.
[42] A. Kharraz, E. Kirda, Redemption: Real-time protection against ransomware at end-hosts, in: Proceedings of the International Symposium on Research in Attacks, Intrusions, and Defenses, Springer, 2017, pp. 98–119.
[43] A. Kharraz, W. Robertson, E. Kirda, Protecting against ransomware: A new line of research or restating classic ideas? IEEE Secur. Priv. 16 (3) (2018) 103–107.
[44] R. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Proceedings of the 6th International Symposium on Micro Machine and Human Science, IEEE, 1995, pp. 39–43.
[45] B. Xue, M. Zhang, W.N. Browne, Particle swarm optimization for feature selection in classification: A multi-objective approach, IEEE Trans. Cybern. 43 (6) (2013) 1656–1671.
[46] B. Xue, M. Zhang, W.N. Browne, X. Yao, A survey on evolutionary computation approaches to feature selection, IEEE Trans. Evol. Comput. 20 (4) (2016) 606–626.
[47] B. Tran, B. Xue, M. Zhang, Variable-length particle swarm optimization for feature selection on high-dimensional classification, IEEE Trans. Evol. Comput. 23 (3) (2018) 473–487.
[48] B. Tran, B. Xue, M. Zhang, A new representation in PSO for discretization-based feature selection, IEEE Trans. Cybern. 48 (6) (2018) 1733–1746.
[49] Y. Xue, B. Xue, M. Zhang, Self-adaptive particle swarm optimization for large-scale feature selection in classification, ACM Trans. Knowl. Discov. Data 13 (5) (2019) 1–27.
[50] Y. Zhang, S. Wang, P. Phillips, G. Ji, Binary PSO with mutation operator for feature selection using decision tree applied to spam detection, Knowl.-Based Syst. 64 (2014) 22–31.
[51] M.S. Abbasi, H. Al-Sahaf, I. Welch, Particle swarm optimization: A wrapper-based feature selection method for ransomware detection and classifcation, in: Proceedings of the 23rd International Conference on the Applications of Evolutionary Computation, in: Lecture Notes in Computer Science, vol. 12104, Springer, 2020, pp. 181–196.
[52] S. Maniath, A. Ashok, P. Poornachandran, V. Sujadevi, P.S. AU, S. Jan, Deep learning LSTM based ransomware detection, in: Proceedings of the Recent Developments in Control, Automation & Power Engineering, IEEE, 2017, pp. 442–446.
[53] R. Agrawal, J.W. Stokes, K. Selvaraj, M. Marinescu, Attention in recurrent neural networks for ransomware detection, in: Proceedings of the 2019 IEEE International Conference on Acoustics, Speech and Signal Processing, IEEE, 2019, pp. 3222–3226.
[54] H. Zhang, X. Xiao, F. Mercaldo, S. Ni, F. Martinelli, A.K. Sangaiah, Classification of ransomware families with machine learning based on N-gram of opcodes, Future Gener. Comput. Syst. 90 (2019) 211–221.
[55] A. Continella, A. Guagnelli, G. Zingaro, G. De Pasquale, A. Barenghi, S. Zanero, F. Maggi, ShieldFS: A self-healing, ransomware-aware filesystem, in: Proceedings of the 32nd Annual Conference on Computer Security Applications, ACM, 2016, pp. 336–347.
[56] Q. Chen, S.R. Islam, H. Haswell, R.A. Bridges, Automated ransomware behavior analysis: Pattern extraction and early detection, in: Proceedings of the International Conference on Science of Cyber Security, Springer, 2019, pp. 199–214.
[57] S. Fong, S. Deb, X.-S. Yang, J. Li, Feature selection in life science classification: Metaheuristic swarm search, IT Prof. 16 (4) (2014) 24–29.
[58] X.-F. Song, Y. Zhang, Y.-N. Guo, X.-Y. Sun, Y.-L. Wang, Variable-size cooperative coevolutionary particle swarm optimization for feature selection on high-dimensional data, IEEE Trans. Evol. Comput. 24 (5) (2020) 882–895.
[59] X.-f. Song, Y. Zhang, D.-w. Gong, X.-y. Sun, Feature selection using bare-bones particle swarm optimization with mutual information, Pattern Recognit. 112 (2021) 107804.
[60] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in python , J. Mach. Learn. Res. 12 (2011) 2825–2830.
[61] L.J.V. Miranda, et al., PySwarms: A research toolkit for particle swarm optimization in python, J. Open Source Softw. 3 (21) (2018) 433.
[62] Y. Shi, R.C. Eberhart, Parameter selection in particle swarm optimization, in: Proceedings of the International Conference on Evolutionary Programming, Springer, 1998, pp. 591–600.
[63] Micorsoft, Windows technical documentation, 2020, URL https://docs.microsoft.com/en-us/windows/, last accessed on 23-10-2020.