


<b>ITC 3/50</b> Information Technology and Control Vol. 50 / No. 3 / 2021 pp. 495-506 DOI 10.5755/j01.itc.50.3.25816	<b>Ransomware Detection Based on Opcode Behaviour Using K-Nearest Neighbours Algorithm</b>	
	Received 2020/04/16	Accepted after revision 2021/08/04
	 <a href="http://dx.doi.org/10.5755/j01.itc.50.3.25816">http://dx.doi.org/10.5755/j01.itc.50.3.25816</a>	

**HOW TO CITE:** Stiawan, D., Daely, S. M., Heryanto, A., Afifah, N., Idris, M. Y., Budiarto, R. (2021). Ransomware Detection Based on Opcode Behaviour Using K-Nearest Neighbours Algorithm. *Information Technology and Control*, 50(3), 495-506. <https://doi.org/10.5755/j01.itc.50.3.25816>

# Ransomware Detection Based on Opcode Behaviour Using K-Nearest Neighbours Algorithm

**Deris Stiawan, Somame Morianus Daely, Ahmad Heryanto**

Dept. of Computer Engineering, Universitas Sriwijaya, Palembang; Indonesia;  
e-mails: [deris@unsri.ac.id](mailto:deris@unsri.ac.id), [somamemorianusdaely@gmail.com](mailto:somamemorianusdaely@gmail.com), [hery@unsri.ac.id](mailto:hery@unsri.ac.id)

**Nurul Afifah**

Dept. of Informatics Engineering, Universitas Sriwijaya, Palembang; Indonesia; e-mail: [afifahnurul95@gmail.com](mailto:afifahnurul95@gmail.com)

**Mohd. Yazid Idris**

School of Computing, Faculty of Engineering, Universiti Teknologi Malaysia, Johor; Malaysia; e-mail: [yazid@utm.my](mailto:yazid@utm.my)

**Rahmat Budiarto**

College of Computer Science and Information Technology, Albaha University, Albaha; Saudi Arabia;  
e-mail: [rahmat@bu.edu.sa](mailto:rahmat@bu.edu.sa)

**Corresponding author:** [deris@unsri.ac.id](mailto:deris@unsri.ac.id)

Ransomware is a malware that represents a serious threat to a user's information privacy. By investigating how ransomware works, we may be able to recognise its atomic behaviour. In return, we will be able to detect the ransomware at an earlier stage with better accuracy. In this paper, we propose Control Flow Graph (CFG) as an extracting opcode behaviour technique, combined with 4-gram (sequence of 4 "words") to extract opcode sequence to be incorporated into Trojan Ransomware detection method using K-Nearest Neighbors (K-NN) algorithm. The opcode CFG 4-gram can fully represent the detailed behavioural characteristics of Trojan Ransomware. The proposed ransomware detection method considers the closest distance to a previously identified ransomware pattern. Experimental results show that the proposed technique using K-NN, obtains the best accuracy of 98.86% for 1-gram opcode and using 1-NN classifier.

**KEYWORDS:** Malware, Ransomware, Opcode behaviour, CFG, K-NN, Accuracy.

---

## 1. Introduction

Ransomware usually encrypts files on a victim's computer and demands payment before offering to unlock them back. Ransomware attacks have recently been on the rise, causing millions of dollars' worth of losses all over the world. This phenomenon happens because the existing detection methods are slow and less accuracy. Therefore, it is significant to increase the accuracy of these malware detection methods. K-NN is a technique for prediction and classification [1]. The classification is based on the nearest neighbour class to the training dataset of feature space, where K shows the number of neighbours that determines the number of classes. Every decision is made by considering the similarity of the majority of neighbours in the training process [16]. This work uses opcode N-gram features [21] for the extraction process. In general, a program is run by executing an opcode sequence (operational code in machine language), then the opcode can be used to describe the program's behaviour. Opcode distribution on malware files is significantly different from normal files [4]. One approach is to extract opcode sequences based on control flow graphs of opcode sequences in the malware files [19]. Extraction of training datasets using opcode N-gram feature extraction and taking into account the control flow of opcode sequences [26] on malware files including normal files, can increase accuracy in detecting malware malicious behaviour that distinguishes it from normal files. The training process in this work uses the malware files and normal files. The static feature extraction consists of byte N-gram features, opcode N-gram features, portable executables, string features, and function features.

Hashemi et al. [10] took two previous works as references. The first one is the work by Ding et al. [8], which reports an accuracy rate of 92% for malware detection using CFG-KNN algorithm. In that study, the authors used opcode graphics extracted from executable files. The second one is the work by Chakkaravarthy et al. [25] who used the control flow graph of malware samples to generate an execution tree to get the execution path. Hence, in their research, Hashemi et al. [10] concluded to combine all possible opcode pathways and use the N-gram method to extract behavioural features. By using the K-NN classification algorithm, the malware detection achieved the highest accuracy of 98.80%.

The rest of this paper is arranged as follows. Section 2 presents some related works, while section 3 gives research methodology, followed by Section 4 that discusses the evaluation on the performance. Then, Section 5 presents the results of the experiments. Lastly, Section 6 concludes the work.

---

## 2. Related Work

Chittooparambil et al. [6] analysed the classification of existing ransomware along with their detection and prevention methods. They classified the ransomware families from the year 1989 to 2017 and discovered that there is not much difference in their patterns. The main objective of their research was to understand the operation of ransomware in Microsoft Windows operating systems through investigating the five stages of ransomware operation. In their experiments, the researchers focused on three different families of ransomware (Scareware, Lockscreen and Crypto-Ransomware), and the five-stages of operation of the ransomware; Installation, Communication, File Search, Encryption, and Extortion. Their conclusion, from the evaluation of the existing methods was that there is no attempt to detect or stop the ransomware in the initial two stages.

Agrawal et al. [2] adapted the deep learning methods to be used for detecting ransomware from emulation sequences. The researchers presented specialised recurrent neural networks for capturing local event patterns in ransomware sequences using the concept of attention mechanism. The researchers demonstrated the performance of enhanced Long Short-Term Memory (LSTM) models on a sequence dataset derived by emulating ransomware executable targeting the Windows environment. The researchers performed a detailed analysis of ransomware executables in order to identify structural properties that can be exploited by machine learning systems. They recognised the presence of slight repeating patterns within long sequences of ransomware potentially corresponding to repeated encryption operations. Then the researchers presented a novel recurrent neural network component for exploiting the repeating patterns by incorporating attention mechanisms on the inputs of a sequence learning module.

The researchers also introduced an enhanced neural cell to incorporate attention in learning from ransomware sequences, called Attended Recent Inputs (ARI) and subsequently used it to modify the LSTM, named as ARI-LSTM. The researchers conducted an experiment using a dataset of unique file sequences consisting of ransomware and benign executables for the Windows operating system captured from client computers to train their model. Their empirical results showed that ARI-LSTM performs significantly better than LSTM for the task of ransomware detection. The researchers have shown that incorporating attention at the inputs of a sequence can be used to solve problems sensitive to relations within recent inputs.

Classifying ransomware into 10 classes which are labeled using *avclass* tool was carried out by Ouerdi et al. [20]. In their study, the researchers used multi-layer perceptron artificial neural networks (MLP-ANNs). The objective of the work was to investigate whether the neural networks are an effective means for the classification of the different types of ransoms. The researchers implemented the MLP-ANN in a Java programming environment and focused on exploiting the Malware'O'Matic (MoM) platform and online databases in order to get real examples of ransoms that encrypt disk.

The experimental results showed that the classification by the MLP-ANNs did not lead to a satisfying result. They concluded that it may be due to one of two reasons: either the choice of artificial neuron networks for ransomware classification was not really a good choice or the misclassification of the ransomware was the irrelevance of the strings contained in ransomware files. They stated that the classification was totally based on the extraction of the common strings between ransoms of each class. The researchers suggested that future works should be carried out on a classification algorithm by K-means in order to find the relevant clusters.

Meanwhile, Craciun et al. [7] investigated the development of ransomware programs and how they were released in certain market segments throughout the deep web via RaaS, exploits or spam. The researchers also highlighted some mistakes that were made, which allowed recovering the encrypted data, along with the ransomware authors preference for specific encryption types, how they got to distribute the silent agreement between ransoms, coin -Miners, bot-

nets and some edge cases of encryption, which may prove to be exploitable in the short-coming future.

A study by Kabakus [11] discussed a static analysis to detect malware in an Android ecosystem. The researcher proposed a novel Android malware detection approach based on static analysis techniques and attempts to prove the effectiveness of the novel static analysis features' in terms of detecting malware in an Android ecosystem. Each feature used by the proposed approach is evaluated by using different types of machine learning techniques in order to highlight its impact on detecting malware and inform the digital investigators. The researcher used three publicly available dataset; Android Genome Project, Debrin, and F-Droid datasets. Meanwhile, for comparison the researcher used several machine learning algorithms including: KNN1-KNN5, Bayes Net, Naïve Bayes, Logistic Regression, SMO-polykernel, J48, Random Forest 100, Random Forest 1000, Random Tree, Bagging and AdaBoost. The experimental result shows that the proposed approach outperforms the above machine learning algorithms and it is very effective in terms of detecting Android malware. The accuracy of the proposed static analysis approach was calculated to be as high as 0.987 for 10,865 mobile applications. The researcher suggests enhancing the proposed approach by considering the source code analysis to interpret real intensions of API calls.

From the previous works on malware recognition/identification, the authors of this paper have summarised that investigating ransomware features in their low-level presentation, (i.e.: the opcode), may provide a better accuracy. In addition, the use of control flow graph analysis and N-gram feature extraction method has shown its superiority. Thus, this work continues the investigation on finding the best patterns of ransomware features in opcode level and will continue to use the combination of the control flow graph analysis and N-gram feature extraction in an attempt to come out with a better ransomware detection system.

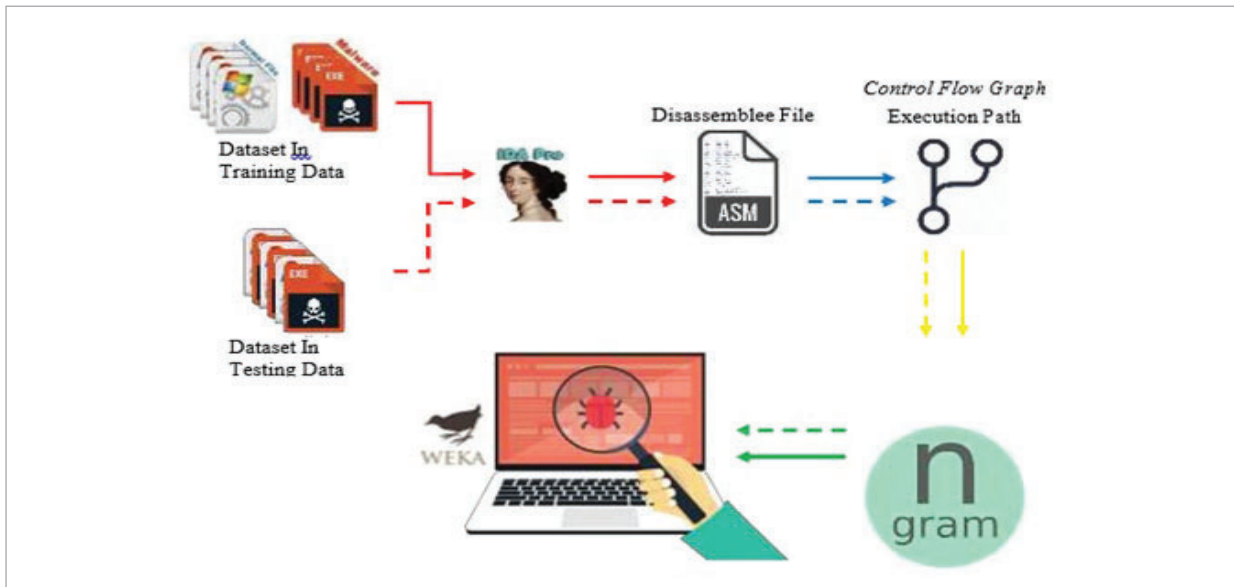
### 3. Research Methodology

All processes in this work only use one physical device that enables IDA Pro and Python language programming.

Figure 1 shows the design of the research activities.

**Figure 1**

Research Activities



### 3.1. Dataset Representation

Dataset in this research consists of 3,000 normal files and 3,000 ransomware files in Windows PE (Portable Executable) format. Normal files are collected from the System32 folders on Windows 7, Windows 8, Windows 10, while malware files consist of Trojan Ransomware taken from the VX Heaven Virus Collection Database.

#### 3.1.1. Representation of Normal File

The normal file dataset in this study comes from the Windows executable file of Windows 8 operating system. Normal files are taken from the System32 folder with the .exe file format. The normal representation on this file is the same as the malware dataset that will be displayed in assembly language presentation. Control Flow Graph will be displayed to find out how normal files work. Every program is run by executing a sequence of instructions (opcode) [23].

Figure 2 shows an assembly view of a normal file in opcode sequence [24] of the execution when the program runs. It is part of the executable control of the program before running the main program using the call instructions. The path of the opcode [14] sequence from the normal file is straightforward, where it uses 4 instructions, which contain 1 call instruction and 1 jump instruction.

**Figure 2**

Assembly View in Normal Files

```
.text:0000000140007C50
.text:0000000140007C50  sub  rsp, 28h
.text:0000000140007C54  call sub_140007BE0
.text:0000000140007C59  add  rsp, 28h
.text:0000000140007C5D  jmp  shortsub_140007C68
.text:0000000140007C5F  db   0CCh
```

#### 3.1.2. Representation of Ransomware File

The ransomware data are taken from the Vx Heaven Virus database.

Similar to the normal file, the malicious code malware dataset is represented in assembly language and Control Flow Graph before the analysis process is carried out [5]. In the disassemble results of the malware dataset file in Figure 3 we are able to see the executable part of the ransomware files. It can also be seen in Figure 3 that the executable part of the ransomware is different from the normal file. The following is an explanation of the ransomware executable assembly view section. Call, is an instruction that a function uses to call a certain sub-

**Figure 3**

Assembly View in Ransomware Files

```

.text:004010CC      public start
.text:004010CC      start proc near
.text:004010CC
.text:004010CC      StartupInfo = ptr44h
.text:004010CC
.text:004010CC      push ebp
.text:004010CD      mov  ebp, esp
.text:004010CF      sub  esp, 44h
.text:004010D3      call ds:GetCommandLineA
.text:004010FE      cmp  byte ptr [esi],
.text:00401109      cmp  byte ptr [esi], 0

```

routine. A subroutine consists of a set of instructions that has a specific task and a function that has a specific task [9].

### 3.2. Control Flow Graph

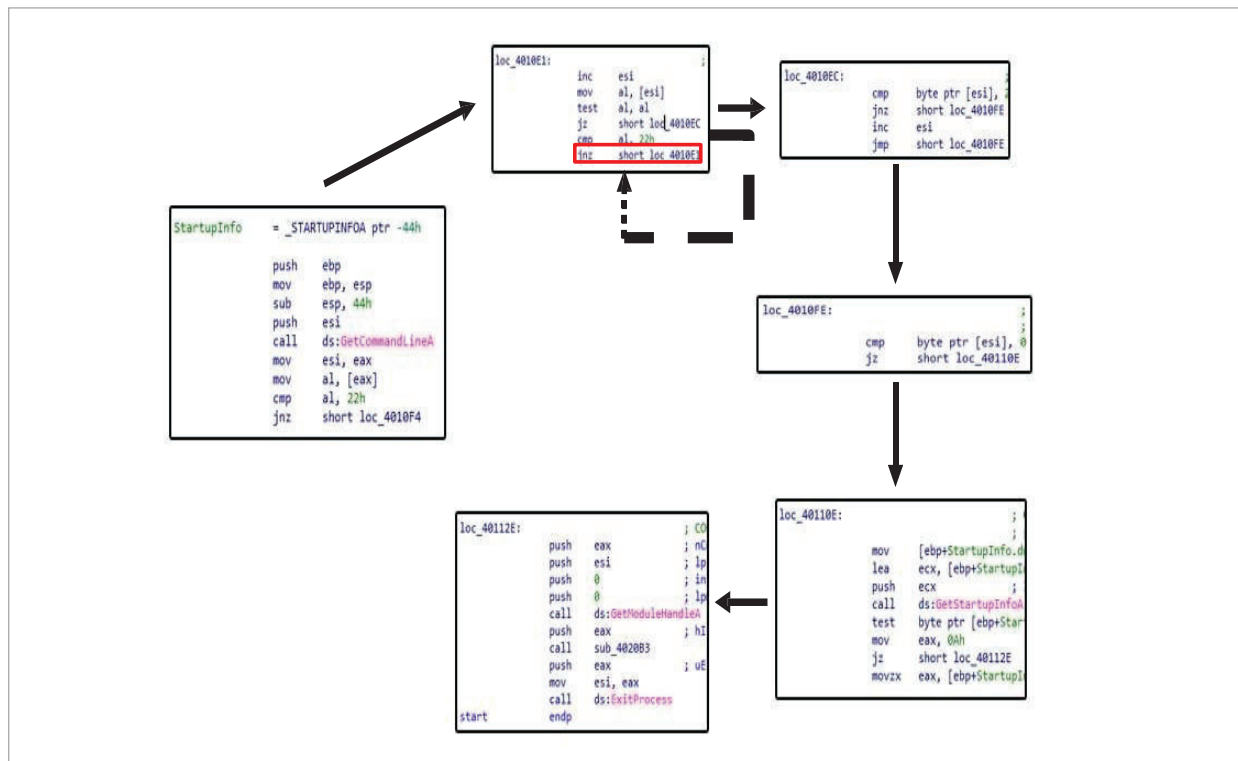
Control Flow Graph (CFG) is a graph that represents the flow of the program in the form of graph node(s) [15]. The executable part of the disassemble process from normal files and malware files that have been done previously can be represented in CFG. Figure 4 exhibits the CFG view of the normal executable file and malware file of the disassemble process.

The CFG executable of normal file and ransomware file in Figure 4 are very different. CFG in a normal file has only one execution path with 1 jump, while a ransomware file has many execution paths with many jumps using opcode jump type instruction if a condition is satisfied. The executable part is carried out when the program is executed.

The result of detection process in Figure 4 shows that CFG executable ransomware has many jump instructions. The number of jumps is created at the beginning of the execution technique (Engineering Confusion) to avoid detection. An analysis is needed to set the opcode

**Figure 4**

Control Flow Graph in executable file





behaviour. Analysing CFG from the ransomware executable will help to find the execution path in a different opcode sequence from the disassemble file. A sequence of opcode malware in disassemble process is achieved when an extracted path, has only one execution of the opcode sequence and when the program is executed, another opcode sequence is found in a different opcode sequence. By analysing the CFG, the executable malware will be able to find the execution path in a different opcode sequence. The purpose of this CFG analysis is to find all execution paths that can possibly be executed.

### 3.3. Feature Extraction

Having done the CFG analysis, every opcode sequence found in the execution path will be extracted to the form of sequence of opcode instructions [15]. In this work, every execution path that had been obtained from the disassemble file is saved in a file with .asm format, and then opcode strings will be selected using Python language. Opcode sequence extraction is an input file that contains all the execution paths that have disassembled the executable malware or normal files obtained from the CFG analysis process.

In the process of extracting opcode sequences, memory locations and registers are ignored, only the opcode sequence is taken. Next, the N-gram extraction feature is used to extract the features. The N-gram extraction feature in Figure 5 is applied in natural

language processing and document classification because it is good at capturing substring statistics and implicit data features. Liangboonprakong et al. [15] used N-gram in malware detection to analyse the opcode sequence to describe the behaviour of malware. The feature extraction in Figure 5 shows that the N-gram extraction process is an important part, which affects all phases of classification-based detection method that includes feature extraction, feature selection, and classification. This work uses  $N = 1$ ,  $N = 2$ ,  $N = 3$ , and  $N = 4$  for the extraction process in N-gram opcode. The N-gram opcode extraction will be sorted by highest to lowest occurrence frequency.

### 3.4. Feature Selection

The feature selection process is carried out to select the best representatives of features from the extracted features [12]. For every extraction process from  $N = 1$  to  $N = 4$ , the top 10 opcodes will be chosen with the most frequent appearances to be used in the training classification process. Each executable file is represented as a vector. The 10 highest frequency opcode patterns in the extraction process in every executable file are converted to vectors. Every opcode sequence is converted into a number according to the opcode instruction parameter. The opcode instruction parameter was previously used as a parameter for the opcode sequence extraction process.

### 3.5. K-Nearest Neighbors Classifier

K-Nearest Neighbors (K-NN) algorithm is a kind of supervised learning [13] used to detect unknown files and classify them according to pre-existing classes [18]. The K-NN is one of many classification algorithms used to set the unknown objects based on the majority of objects that have the same attributes and closest distance [22].

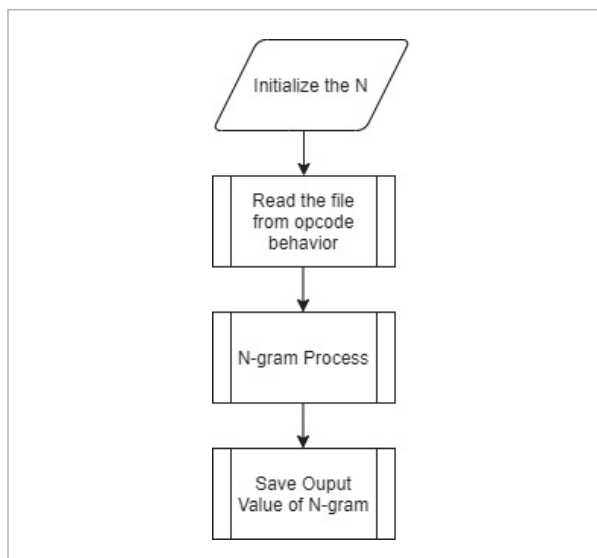
Determining the distance between two objects can be achieved using the Euclidean Distance  $d(x_i, y_i)$ , where  $x_i$  is Latitude and  $y_i$  is Longitude as explained in Equation (1).

$$d(x_i, y_i) = \sqrt{(x_i - y_i)^2}. \quad (1)$$

Steps in K-NN algorithm are as follows [22].

- 1 Initialise the parameter of K (number of closest neighbours)
- 2 Sort objects that have the smallest distance.

**Figure 5**  
N-gram Extraction Process



- 3 Collect the Y category (Nearest Neighbor Classification).
- 4 Using the Nearest Neighbor category, the value of the calculated query instance can be predicted.

This work chooses the number of K=1 to K=10 to investigate the number of K with the best accuracy.

## 4. Performance Evaluation

The authors of this paper use Confusion Matrix to measure the performance of the detection method, by referring to the work in [17]. A detection system's performance is evaluated by three categories, namely Accuracy, Precision, and Recall. The parameters required to measure them are true positive (TP), true negative (TN), false positive (FP), and false negative (FN). The Accuracy measures the level of accuracy of the method in detecting malware from all datasets as in Equation (2).

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (2)$$

Precision is the ratio of the number of malware detected correctly to the total number of datasets detected as malware. The Precision measures all the true positive and false positive. Precision formula is represented in Equation (3).

$$Precision = \frac{TP}{FP+TP} \quad (3)$$

The next performance measurement is Recall, which explains the ratio of the number of malware detected correctly (TP) to the total number of malware datasets tested, which are (TP) and (FN).

The formula to measure the Recall is expressed in Equation (4).

$$Recal = \frac{TP}{TP+FN} \quad (4)$$

## 5. Experimental Results

### 5.1. Feature Extraction Results

N-gram extraction feature is used to extract opcode sequences with N = 1 to N = 4. As we expected, the opcode N-gram extraction of malware files and normal

files for each different N produces a different vocabulary. Figure 6 shows the different length of vocabulary results in opcode N-gram extraction of malware files and normal files.

**Figure 6**  
N-gram Extraction

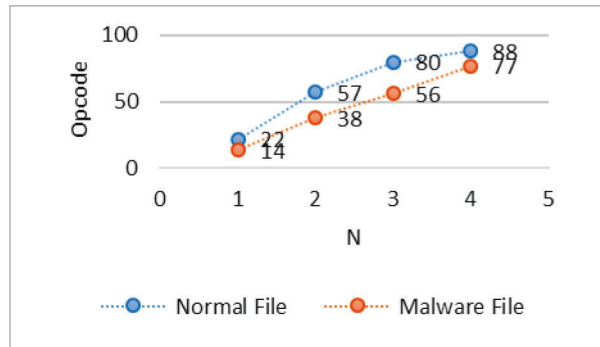


Figure 6 shows the opcode N-gram extracted features of the executable malware files and normal files. 1-gram extraction produces 22 normal files and 14 malware files, while 2-gram extraction produces 57 normal files and 38 malware files.

Meanwhile, 3-gram extraction produces 80 normal files and 56 malware files, and 4-gram extraction produces 88 normal files and 77 malware files. It is observed that 4-gram extraction produces the highest opcode features in the extraction process.

### 5.2. Classification Results

The experiments on the classification use a dataset that consists of 3000 malware files and 3000 normal files as shown in Table 1.

During the testing stage, K-NN algorithm detects new objects that have been recognised previously during the training stage. In the experimental scenario, classification/detection results during the training stage and testing stage will be compared. This scenario

**Table 1**

Number of training and testing data for each experiment

Experiment 1	Experiment 2	Experiment 3
4500 training data	3000 training data	1500 training data
1500 testing data	3000 testing data	4500 testing data

is done to investigate the performance of the K-NN algorithm when the number of data in the training stage is either greater than, equal to, or less than the number of data in the testing stage. As shown in Table 1, the scenario carries out three experiments. In Experiment 1, the proportion of training data and testing data is 75% and 25%. In Experiment 2, a 50/50 distribution of training dataset and testing dataset was used. Lastly, in Experiment 3, the proportion of training data and testing data was 25% and 75%, respectively.

Table 2 shows that overall results of the Precision of K-NN algorithm in detecting malware reached 100%, except for (K = 2 and N = 3). In the case of (K = 2 and N = 3) the Precision is 69%, because some normal files were identified as malwares.

The highest Recall value of 98% was achieved for (K=1, N=3, whereas the highest Accuracy of 98.53% was achieved for (K=4, N=1).

Among the files detected as malware, a normal file was also detected as malware file. Nevertheless, the Recall value was still 100%, which means that all malware files were detected correctly as malware files. In other words, the algorithm recognises a malware as true malicious software. Results of Recall  $\neq$  100% indicate that there are malware files detected as normal files. The highest accuracy of 98.86% was achieved for (K = 1, N = 1).

In Table 3, the scenario where the number of data in the training stage is less than the number in the testing stage, the K-NN algorithm reached 100%

**Table 2**

Confusion Matrix of Experiment 1 results (in %)

K	N=1			N=2			N=3			N=4		
	Prec	Rec	Acc	Pre	Rec	Acc	Pre	Rec	Acc	Pre	Rec	Acc
1	100	98	<b>98.86</b>	100	96	98.2	100	95	97.33	100	92	96
2	100	97	98.4	100	96	97.93	<b>69</b>	100	78.8	100	93	96.2
3	100	97	98.46	100	96	98.26	100	94	96.93	100	93	96.33
4	100	97	98.33	100	97	98.53	100	93	96.66	100	93	96.66
5	100	97	98.53	100	96	98.26	100	94	96.73	100	92	96

**Table 3**

Confusion Matrix of Experiment 2 results (in %)

K	N=1			N=2			N=3			N=4		
	Pre	Rec	Acc	Pre	Rec	Acc	Pre	Rec	Acc	Pre	Rec	Acc
1	100	96	98.06	100	95	97.42	<b>69</b>	<b>98</b>	76.48	100	91	95.33
2	100	96	97.88	100	96	97.77	100	93	96.62	100	91	95.73
3	100	96	97.62	100	96	97.8	100	92	96.88	100	91	95.39
4	100	97	<b>98.53</b>	100	95	97.26	100	93	96.33	100	91	95.48
5	100	95	97.57	100	94	96.75	100	93	96.48	100	91	95.28
6	100	96	97.75	100	95	97.06	<b>99</b>	92	95.77	100	91	95.48
7	100	96	97.91	100	94	97.26	100	92	95.95	100	91	95.35
8	100	96	98.02	100	95	97.28	100	91	95.57	100	90	95
9	100	95	97.53	100	94	96.8	100	92	96.02	100	89	94.44



**Table 4**

Confusion Matrix of Experiment 3 results (in %)

K	N=1			N=2			N=3			N=4		
	Pre	Rec	Acc	Pre	Rec	Acc	Pre	Rec	Acc	Pre	Rec	Acc
1	100	97	98.6	75	99	83.8	69	99	77.7	100	92	96.96
2	91	98	94.16	100	97	98.6	100	94	96.8	100	92	96.16
3	100	97	98.53	100	96	98.1	100	94	97	100	91	95.76
4	100	97	98.63	100	96	98	100	94	97.23	100	91	95.26
5	100	97	98.56	100	95	97.46	100	93	97.5	100	92	95.93
6	100	98	<b>98.83</b>	100	95	97.7	100	94	97.1	100	91	95.66
7	100	96	97.76	100	96	98	100	92	96.1	100	91	95.39
8	100	97	98.46	100	96	97.96	100	93	96.6	100	91	95.66
9	100	97	98.43	100	95	97.43	100	94	96.93	100	91	95.36

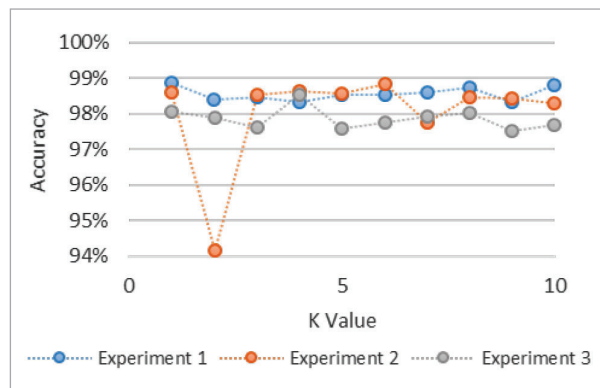
Precision, except for the cases of ( $K = 1, N = 3$ ) and ( $K = 6, N = 3$ ), which were 69% and 99%, respectively. These results indicate that from all the data detected as malware files, there were also normal files. In terms of Recall values, the results showed that the K-NN algorithm can detect malware files because the number of malware files detected as normal files were the least among all detections, as conducted in Experiment 2.

While referring to Table 4, it can be seen that the average of the best Precision in Experiment 3 is for  $N = 4$ , which reached 100%. This shows that in Experiment 3, the K-NN algorithm with  $N = 4$  detected no normal files as malware files during the testing stage.

The highest average of Recall value was for  $N = 1$ , while the lowest average Recall result was for  $N = 4$ . Considering also the Recall values, the K-NN algorithm achieved its highest accuracy of 98.83% in the experiment where  $N = 1$  and the number of  $K = 6$ . Figure 7 presents the accuracy scores for the three experiments. The opcode resulted from 1-gram feature extraction has the highest level of accuracy in detecting malware using the K-NN algorithm. Figure 7 also shows that the K-NN algorithm's Accuracy can reach more than 98%, even though the number of training data is smaller than the number of testing data. The highest accuracy in opcode N-gram is when  $N = 1$  extraction feature. The K-NN algorithm can detect malware with high accuracy using the opcode N-gram where  $N = 1$  extraction feature, compared to

**Figure 7**

Accuracy of the proposed method



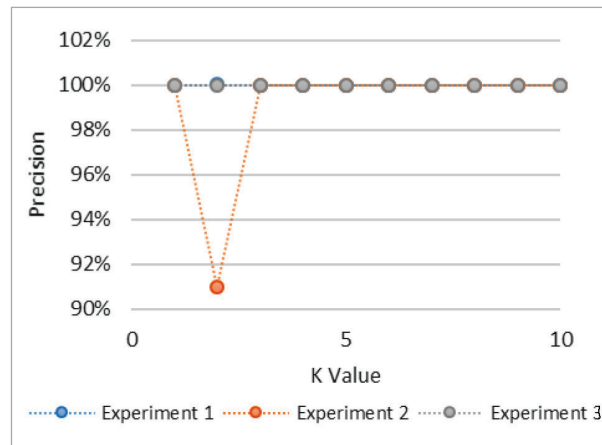
the opcode N-gram extraction feature, where  $N = 2$ ,  $N = 3$ , and  $N = 4$ .

Figure 8 presents the Precision results. The highest Precision result is 100%. The Precision results are influenced by the number of false positive (FP) instances. The highest value of FP is reached when a lot of normal files in the dataset are detected as malware files. The best Precision result is 100% and is obtained when no normal files are detected as malware files.

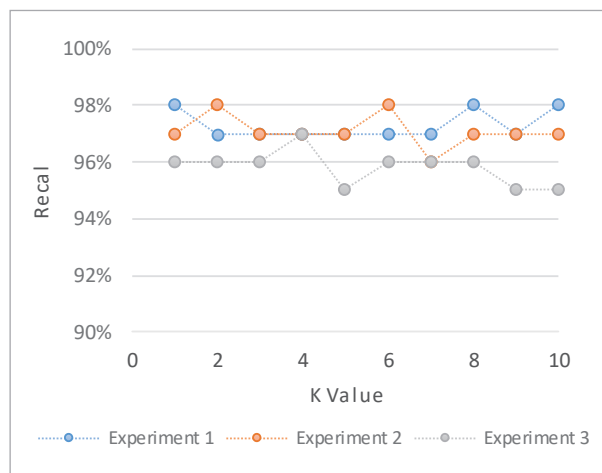
Figure 9 shows the plot of Recall score calculation results. The highest Recall score obtained is 100% with the condition that all malwares are detected.

**Figure 8**

Precision of the proposed method

**Figure 9**

Recal value of the proposed method

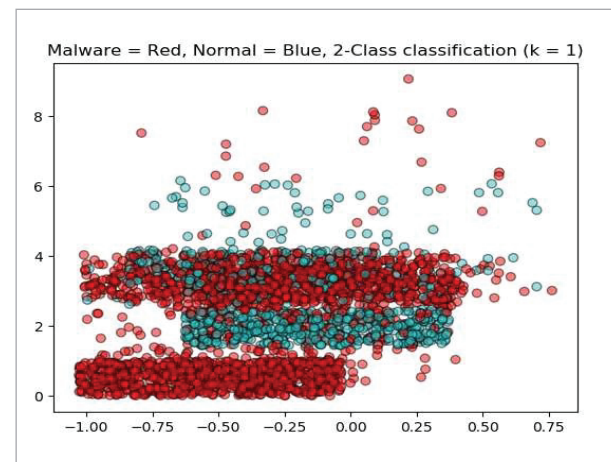


The best optimal detection score with the K-Nearest Neighbors algorithm is 98.86%. From the experience during the experiments, Precision and Recall also affect the results of Accuracy.

Visual of the segregation of ransomware files and normal files using K-NN is shown in Figure 10. It can be seen that only a few red dots spread a high distance from the blue dots, while most of red dots are closer to the blue dots. It is caused by the fact that the opcode parameters used to recognise ransomware/malware are similar as a normal file.

**Figure 10**

Visualization of ransomware and normal files using K-NN



## 6. Discussion

Table 5 shows the performance comparison between the proposed technique and similar techniques.

**Table 5**

Comparison of CFG 4-gram with existing works on the detection accuracy

Features Engineering	Accuracy
CFG 4-gram (Proposed)	98.86%
CFG [4]	92.00%
TXT1 [4]	88.30%
TXT2 [4]	89.70%
TF-IDF [26]	97.05%
ELF [27]	91.00%

Ding et al. [8] used CFG, TXT1, and TXT2 as features' engineering technique and reported an accuracy detection of 92%, 88.30%, and 89.70%, respectively. They also revealed that the use of the TXT technique for extracting opcode sequence has a limitation, where the extracted opcode sequence does not represent real behavior of the Executable. Meanwhile, an experiment by Xu et al. [27] that implements IT-IDF yielded a detection accuracy of 97.05% and it performed better than the results obtained by Ahmed

et al. [3], who used ELF as the features' engineering technique, with a detection accuracy of 91%. Results in Table 5 shows that the proposed technique, which uses CFG 4-Gram, outperforms the other selected techniques. It achieves very good accuracy detection (i.e., 98.86%), because the behavior pattern extracted by the CFG 4-Gram technique can fully represent the behavioural characteristics of an Executable.

## 7. Conclusion and Future Work

In this paper, using CFG 4-gram for feature engineering produces an efficient result in supporting the Trojan Ransomware detection method through the K-NN

algorithm. The highest score of 98.86% accuracy in ransomware detection is obtained during Experiment 1 for the opcode 1-gram and using 1-NN classifier algorithm. Comparing to other selected feature engineering techniques, overall, the proposed technique outperforms them.

In the future, the authors plan to investigate more on relevant and significant ransomwares' features through experiments on combination of several features engineering techniques. The experiments will be conducted on various ransomware datasets to obtain more accurate attack patterns to be used to distinguish ransomware packets from normal packets and also by considering a combination with different classifier algorithms.

## References

- Adeniyi, D. A., Wei, Z., Yongquan, Y. Automated Web Usage Data Mining and Recommendation System Using K-Nearest Neighbor (KNN) Classification Method. *Applied Computing and Informatics*, 2016, 12, 90-108. <https://doi.org/10.1016/j.aci.2014.10.001>
- Agrawal, R., Stokes, J. W., Selvaraj, K., Marinescu, M. Attention in Recurrent Neural Networks for Ransomware Detection. *Proceedings of 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, 3222. <https://doi.org/10.1109/ICASSP.2019.8682899>
- Ahmed, F. S., Mustapha, N., Mustapha, A., Kakavand, M., Foozy, C. F. M. Preliminary Analysis of Malware Detection in Opcode Sequences within IoT Environment. *Journal of Computer Science*, 2020, 16(9), 1306-1318. <https://doi.org/10.3844/jcssp.2020.1306.1318>. <https://doi.org/10.3844/jcssp.2020.1306.1318>
- Bieber, D., Sutton, C., Larochelle, H., Tarlow, D. Learning to Execute Programs with Instruction Pointer Attention Graph. *Neural Networks*, 2020, ArXiv, abs/2010.12621.
- Bilar D. Opcodes as Predictor for Malware. *International Journal of Electronic Security and Digital Forensics*, 2007, (1), 156-168. <https://doi.org/10.1504/IJESDF.2007.016865>
- Chittooparambil, H. J., Shanmugam, B., Azam, S., Kannoorpatti, K., Jonkman, M., Samy, G. N. A Review of Ransomware Families and Detection Methods. *Proceedings of International Conference of Reliable Information and Communication Technology*, 2018, 588-597. [https://doi.org/10.1007/978-3-319-99007-1\\_55](https://doi.org/10.1007/978-3-319-99007-1_55)
- Craciun, V. C., Mogage, A., Simion, E. Trends in Design of Ransomware Viruses. *Proceedings of International Conference on Security for Information Technology and Communications*, 2018, 259-272. [https://doi.org/10.1007/978-3-030-12942-2\\_20](https://doi.org/10.1007/978-3-030-12942-2_20)
- Ding, Y., Dai, W., Yan, S., Zhang, Y. Control Flow-Based Opcode Behavior Analysis for Malware Detection. *Computers and Security*, 2014, 44, 65-74. <https://doi.org/10.1016/j.cose.2014.04.003>
- Ghezelbigloo Z., VafaeiJahan M. Role-Opcode vs. Opcode: The New Method in Computer Malware Detection. *Proceedings of 2014 International Congress on Technology, Communication and Knowledge (ICTCK)*, 2014, 1-6. <https://doi.org/10.1109/ICTCK.2014.7033534>
- Hashemi, H., Azmoodeh, A., Hamzeh, A., Hashemi, S. Graph Embedding as a New Approach for Unknown Malware Detection. *Journal of Computer Virology and Hacking Techniques*, 2016, 13(3), 153-166. <https://doi.org/10.1007/s11416-016-0278-y>
- Kabakus, A. T. What Static Analysis Can Utmost Offer for Android Malware Detection. *Information Technology and Control*, 2019, 48(2), 235-240. <https://doi.org/10.5755/j01.itc.48.2.21457>
- Khammas, B. M., Monemi, A., Stephen Bassi, J., Ismail, I., Mohd Nor, S., Marsono, M. N. Feature Selection and Machine Learning Classification for Malware Detection.

- tion. *Jurnal Teknologi*, 2015, 77(1), 243-250. <https://doi.org/10.11113/jt.v77.3558>
13. Kotsiantis, S. B., Zaharakis, I. D., Pintelas, P. E. Machine Learning: A Review of Classification and Combining Techniques. *Artificial Intelligence Review*, 2006, 26(3), 159-190. <https://doi.org/10.1007/s10462-007-9052-3>
  14. Li, P., Chen, Z., Cui, B. Detecting Malware Based on Opcode N-Gram and Machine Learning. In Xhafa F., Caballé S., Barolli L. (Eds.), *Advances on P2P, Parallel, Grid, Cloud and Internet Computing*. 3PGCIC Lecture Notes on Data Engineering and Communications Technologies, 2017, 13. Springer, Cham. [https://doi.org/10.1007/978-3-319-69835-9\\_9](https://doi.org/10.1007/978-3-319-69835-9_9)
  15. Liangboonprakong, C., Sornil, O. Classification of Malware Families based on N-grams Sequential Pattern Features. *Proceedings of 8th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, 2013, 777-782. <https://doi.org/10.1109/ICIEA.2013.6566472>
  16. Liao, Y., Vemuri, V. R. Use of K-Nearest Neighbor Classifier for Intrusion Detection. *Computers and Security*, 2002, 21(5), 439-448. [https://doi.org/10.1016/S0167-4048\(02\)00514-X](https://doi.org/10.1016/S0167-4048(02)00514-X)
  17. Luque, A., Carrasco, A., Martín, A., de las Heras, A. The impact of Class Imbalance in Classification Performance Metrics Based on the Binary Confusion Matrix. *Pattern Recognition*, 2019, 91, 216-231. <https://doi.org/10.1016/j.patcog.2019.02.023>
  18. Narudin, F. A., Feizollah, A., Anuar, N. B., Gani, A. Evaluation of Machine Learning Classifiers for Mobile Malware Detection. *Soft Computing*, 2014, 20(1), 343-357. <https://doi.org/10.1007/s00500-014-1511-6>
  19. O'Kane, P., Sezer, S., Carlin, D. Evolution of Ransomware. *IET Networks*, 2018, 7(5), 321-327. <https://doi.org/10.1049/iet-net.2017.0207>
  20. Ouerdi, N., Hajji, T., Palisse, A., Lanet, J.-L., Azizi, A. Classification of Ransomware Based on Artificial Neural Networks. *Information Systems and Technologies to Support Learning*, 2018, 384-392. [https://doi.org/10.1007/978-3-030-03577-8\\_43](https://doi.org/10.1007/978-3-030-03577-8_43)
  21. Ranveer, S., Hiray, S. Comparative Analysis of Feature Extraction Methods of Malware Detection. *International Journal of Computer Applications*, 2015, 120(5), 1-7. <https://doi.org/10.5120/21220-3960>
  22. Samaniego, L., Schulz, K. Supervised Classification of Agricultural Land Cover using a Modified k-NN Technique (MNN) and Landsat Remote Sensing Imagery. *Remote Sensing*, 2009, 1(4), 875-895. <https://doi.org/10.3390/rs1040875>
  23. Santos, I., Brezo, F., Ugarte-Pedrero, X., Bringas, P. G. Opcode Sequences as Representation of Executables for Data-Mining-Based Unknown Malware Detection. *Information Sciences*, 2013, 231, 64-82. <https://doi.org/10.1016/j.ins.2011.08.020>
  24. Shabtai, A., Moskovitch, R., Elovici, Y., Glezer, C. Detection of Malicious Code by Applying Machine Learning Classifiers on Static Features: A State-of-the-Art Survey. *Information Security Technical Report*, 2009, 14(1), 16-29. <https://doi.org/10.1016/j.istr.2009.03.003>
  25. Sibi, C.S., Sangeetha, D., Vaidehi, V. A Survey on Malware Analysis and Mitigation Techniques. *Computer Science Review*, 2019, 32, 1-23. <https://doi.org/10.1016/j.cosrev.2019.01.002>
  26. Ucci, D., Aniello, L., Baldoni, R. Survey of Machine Learning Techniques for Malware Analysis. *Computers and Security*, 2019, 81, 123-147. <https://doi.org/10.1016/j.cose.2018.11.001>
  27. Xu, Z., Wen, C., Qin, S., Ming, Z. Effective Malware Detection Based on Behaviour and Data Features. In *Lecture Notes in Computer Science*, 2018, 53-66. Springer International Publishing. [https://doi.org/10.1007/978-3-319-73830-7\\_6](https://doi.org/10.1007/978-3-319-73830-7_6)

