A Secure Encryption-Based Malware Detection System

Zhaowen Lin[1,2,3], Fei Xiao[1,2,3], Yi Sun[2,3,4], Yan Ma[1], Cong-Cong Xing[*5] and Jun Huang[6]

[1] Network and Information Center, Institute of Network Technology, Beijing University of Posts and Telecommunications, Beijing, 100876 – China

[2] Science and Technology on Information Transmission and Dissemination in Communication Networks Laboratory, Shijiazhuang, 050081 – China

[3] National Engineering Laboratory for Mobile Network Security, Beijing University of Posts and Telecommunications, Beijing, 100876 – China

[4] Network and Information Center, Institute of Network Technology/ Institute of Sensing Technology and Business, Beijing University of Posts and Communications, Beijing, 100000 – China

[5] Deptatrment of Mathematics/Computer Science, Nicholls State University, Thibodaux, LA 70310 – USA

[6] School of CIE, Chongqing University of Posts and Telecommunications, Chongqing, 400065– China

fige-mail: linzw@bupt.edu.cn; xiaofei@bupt.edu.cn; sybupt@bupt.edu.cn; mayan@bupt.edu.cn; cong-cong.xing@nicholls.edu; jhuang@cqupt.edu.cn]

*Corresponding author: Cong-Cong Xing

Received June 28, 2017; revised October 8, 2017; accepted October 30, 2017; published April 30, 2018

Abstract

Malware detections continue to be a challenging task as attackers may be aware of the rules used in malware detection mechanisms and constantly generate new breeds of malware to evade the current malware detection mechanisms. Consequently, novel and innovated malware detection techniques need to be investigated to deal with this circumstance. In this paper, we propose a new secure malware detection system in which API call fragments are used to recognize potential malware instances, and these API call fragments together with the homomorphic encryption technique are used to construct a privacy-preserving Naive Bayes classifier (PP-NBC). Experimental results demonstrate that the proposed PP-NBC can successfully classify instances of malware with a hit-rate as high as 94.93%.

Keywords: Malware detection, detection mechanism, API call fragments, homomorphic encryption, privacy-preserving Naive Bayes classifier

1. INTRODUCTION

Malicious software, generally called malware, can be characterized as being able to compromise computer systems by replicating, propagating, self-executing itself [1]. Despite the effectiveness of machine learning as one of the major malware detection techniques, we unfortunately face a growing challenge: certain malware may be aware of some of the techniques used in malware detection systems, and thus may attempt to evade being detected by employing some new techniques. A typical such example can be found in [2] where specifically crafted training data is injected into the system to purposely increase the error rate of the learning of the support vector machine.

In order to prevent the malware from having such detection-evading capabilities, malware detection mechanisms need to be protected. Therefore, designing and implementing a new malware detection system, which can detect the malware effectively and protect the detection mechanism itself at the same time, is of practical significance. Although there are extensive studies on the malware detection in the literature (e.g., see [3], [4], [5], [6], [7], [8], [9]), most of these studies overlook the issue of protecting the malware detection mechanism.

In this paper, we propose a secure malware detection system (SMD-DE) in which the malware detection mechanism is encrypted. This system uses Application Programming Interface (API) call fragment sequences to characterize the behavior of malware instances, and provides a privacy-preserving Naive Bayes classifier [10] (PP-NBC) by using the homomorphic encryption technique [11], [12]. The contributions of our work are summarized as follows.

1) Aiming to enhance the detection of malware and contribute to the study on the protection of detection mechanisms, we propose a secure malware detection system in which the detection mechanism is encrypted and thus protected.
2) A privacy-preserving Naive Bayes classifier which combines the behavior-based inspection technique (for detecting malware) with the homomorphic encryption technique (for protecting the detection mechanism) is constructed.

3) The proposed secure malware detection system is based on a Naive Bayes classifier which has a considerably high successful malware detection rate (94.93%).

The remainder of this paper is organized as follows. Section 2 discusses related work. Section 3 provides an introduction to information gain, Paillier homomorphic encryption, Naive Bayes classifier and secure malware detection protocol. The proposed SMD-DE system is described in Section 4, which is followed by the evaluation of SMD-DE in Section 5. Section 6 concludes the paper with some suggestions for possible future research directions.

2. RELATED WORK

The subject of malware detection is vitally important in the field in network security. Since programs send their requests to the operating system by using API calls, the characteristic or pattern of API class has long been recognized as an important indicator for revealing programs' behaviors and is thus excellent candidate for mining malicious programs' behaviors. For example, Eskandari et al. [13] used n-gram algorithm to preserve the ordering of the API calls and constructed a behavior-based malware detection system so that different orders of API calls can indicate different behaviors of programs.

The approach of extracting the API calls from programs can be either static or dynamic. The former is known as a reverse engineering method that extracts API calls from the source code of the program without actually running the file [13], as exemplified by the work of Ye et al. [14]. Unfortunately, static approach has many limitations since it cannot handle packed malware and can be conquered by obfuscation techniques [15] that generate a new "hard-to-read" copy of a program preserving the behavior of the original program, or use some embedded encryption engine to make a new copy of a program by a random key [16]. The dynamic approach overcomes the limitations of the static approach by running programs in a virtual environment and subsequently mining the program behaviors. An example of the dynamic approach can be found in [17] where graphs are used to organize and analyze API calls. In this paper, we use the dynamic approach to extract the API calls.

As introduced in [11], homomorphic encryption (HE) is one of the encryption methods which provides a means for securely transmitting and storing confidential information. HE has been successfully used in medical record processing [18], [19], genomics [20], voting [21], and multiparty computations [22] and securing outsourced Big Data computation [23]. As a constantly growing component of the cryptography, HE involves some basic operations such as addition, subtraction, multiplication and division, and can be generally used to protect the privacy of information.

There exist many malware detection studies in the literature. For example, a simplified malware detection system is implemented by Elhadi et al. [3]. Saxe et al. [4] proposed a deep-learning based malware detection method with notably high malware detection rate. Fan et al. [5] successfully utilized the hooking technique to trace and monitor the behaviors of malware, and Maiorca [6] employed proactive approaches to develop a defense system for predicting possible evasion attacks. However, none of them has considered the privacy-preserving [24], [25] issue in protecting the malware detection mechanism. Although privacy-preserving computation has been studied extensively in the area of medical record processing (e.g., see [26], [27], [28]), it has been rarely applied to the field of malware detections. Our work in this paper proposes a secure and privacy-preserving classifier that utilizes the HE technology and the secure protocol described in [29].

3. PRELIMINARIES

3.1 Information Gain

Information gain (IG) is usually denoted as mutual information in information theory which is used to denote information exchange and to effectively select certain properties.

Suppose we have the following API fragments set $\{api_1, api_2, \ldots, api_n\}$. Let C_1 denote the class of malicious programs, and C_2 denote the class benign programs. Also, let $P(V_l, C_i)$ $(l = 1, 2, \ldots, n;\ V_l = 0, 1;\ i = 1, 2)$ denote the probability of api_l being in the class of C_i if $V_l = 1$, and the probability of api_l *not* being in the class of C_i if $V_l = 0$. Furthermore, let $P(V_l)$ and $P(C_i)$ denote the probabilities of api_l and of the C_i class in the entire sample set, respectively. As such, the information gain associated with api_l, $IG(l)$, is defined as

$$IG(l) = \sum_{V_l \in \{0,1\}} \sum_{i=1}^{2} P(V_l, C_i) \log_2 \frac{P(V_l, C_i)}{P(V_l) P(C_i)} \tag{1}$$

3.2 Paillier Homomorphic Encryption

In order to construct a PP-NBC, we use the well-known Paillier homomorphic encryption [30], which is a probabilistic public key encryption algorithm. In Paillier cryptosystem the product of ciphertexts encrypts the sum of plaintexts. The Paillier homomorphic encryption scheme consists of the following three steps: key generation, encryption, and decryption.

Key Generation: Randomly choose two large prime numbers p and q of equivalent length. Let $\mathrm{N} = \mathrm{pq}$ and $\lambda = lcm(p-1, q-1)$. Choose a random integer $g \in \mathbb{Z}_{N^2}^*$ (the set of nonzero integers modulo N^2) and calculate $\mu = \left(L\left(g^{\lambda}\ mod\ N^2\right)\right)^{-1} \bmod N$, where $L(x)$ is defined as $L(x) = \frac{x-1}{N}$. Then, the public (encryption) key would be (N, g) and the private (decryption) key would be (λ, μ).

Encryption: Let $m \in \mathbb{Z}_N$ (the set of integers modulo N) be a plain message to be encrypted, choose a random $r \in \mathbb{Z}_{N^2}^*$ and then m can be encrypted as follows (where $pk = (N, g)$)

$$Enc(m,pk) = g^m r^N \bmod N^2 \quad (2)$$

Decryption: If c is the encrypted message $Enc(m,pk)$, then m can be restored by

$$m = \left(L\left(c^{\lambda} \bmod N^2\right)\mu\right) \bmod N \quad (3)$$

Let $D(\cdot)$ represent the operation of decryption. We notice that in Paillier homomorphic cryptographic systems, any two encrypted messages

$$Enc(x,pk) = g^x r_1^N \bmod N^2 \quad \text{and}$$
$$Enc(y,pk) = g^y r_2^N \bmod N^2$$

satisfy the following property

$$\begin{aligned} &D(Enc(x,pk) \otimes Enc(y,pk) \bmod N^2) \\ &\quad = D\left(\left((g^x r_1^N) \otimes (g^y r_2^N) \bmod N^2\right) \bmod N^2\right) \\ &\quad = D(g^{x+y}(r_1 r_2)^N \bmod N^2) \\ &\quad = (x+y) \bmod N \end{aligned}$$

where $\otimes$ denotes the product of two ciphered texts (see [30] for derivation details). This shows that the encryption operation in Paillier cryptographic systems is homomorphic.

3.3 Naive Bayes Classifier

As one of the most practical and effective models in malware detection, Naive Bayes classifier is a supervised learning method with an assumption that the attributes of instances are independent of each other. The basic idea of the Naive Bayes classifier is to compute the probability of the input data belonging to each designated class according to the probability distribution of the training data.

We use API fragment sequences to construct the Naive Bayes classifier. In any API fragment sequence $x = (x_1, x_2, \dots, x_n)$, each API fragment x_j is different and independent of another API fragment x_k. There are two designated classes C_1 and C_2 associated with the proposed PP-NBC with C_1 representing the class of malware and C_2 representing the class of benign programs, and $P(C_i)$ $(i = 1, 2)$ is the probability of the class C_i with respect to the total pool of samples. Let $X = (X_1, X_2, \dots, X_n)$ be the sequence of code fragments to be inspected (which is, typically, the user input), we use $P(X_j = x_j | C_i)$ to represent the probability of the case that X_j matches x_j and x is of class C_i. As is typically done for numerical stability reasons, we use the logarithm of the probability distributions. As such, the class that the input sequence X belongs to can be computed or classified as follows

$$C(X) = \arg \max_{C_i \in \{C_1, C_2\}} \{\log_2[P(C_i) \prod_{j=1}^{n} P\left(X_j = x_j \mid C_i\right)]\} \quad (4)$$

3.4 The SMD-DE Protocol

There are three parties (server, agency and client) that will be involved in the SMD-DE protocol. The server receives the encrypted detection mechanism and the public key from the agency which is responsible for training and updating the malware detection rules. The server utilizes the received detection mechanism to inspect the programs sent to the server from the client and produces some encrypted results which will be sent back to the agency. The agency decrypts the encrypted results by using the private key and then sends the decrypted results to the server which will subsequently forward them to the client. The overall steps of the SMD-DE protocol are described as follows (which are also illustratively labeled in **Fig. 1**).

Step 1: The server receives the public key PK (of Paillier homomorphic encryption) and the encrypted malware detection mechanism from the agency.

Step 2: The server receives programs from the client, analyzes and extracts some behavior information of these programs.

Step 3: The server utilizes Paillier's additive homomorphism to detect the programs sent from the client and generates some encrypted results, which will be transmitted to the agency.

Step 4: The agency decrypts the encrypted result using the private key and passes the decrypted results back to the server.

Step 5: The server returns the decrypted results to the client. (The client may then take some subsequent appropriate actions to the programs based on the recommendation from the server.)

4. THE SMD-DE SYSTEM

We in this section elaborate the proposed SMD-DE system.

4.1 System Overview

The proposed SMD-DE system is secure in the sense that it constructs a PP-NBC which can perform the detection of malware with the detection rules being encrypted. In other words, it will be extremely difficult for any malware to learn any information about the contents of the detection rules. Generally speaking, the proposed SMD-DE system consists of three components: the agency responsible for creating detection rules (AC), a cloud platform (CP), and the programs to be detected (PD), as depicted in **Fig. 1**.

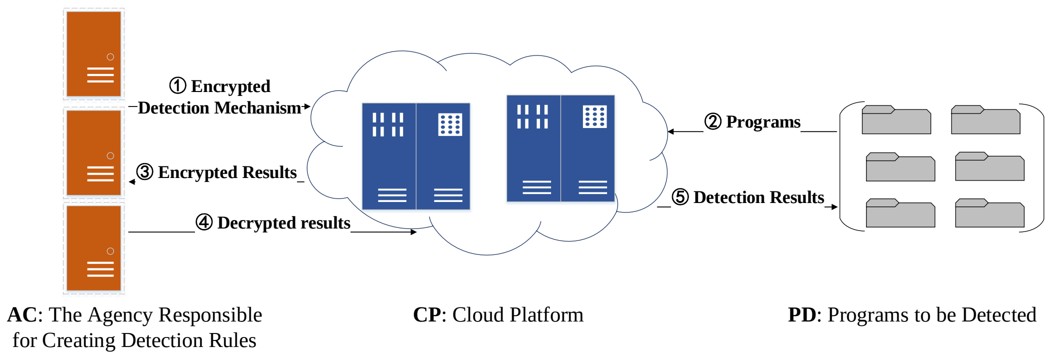
① Encrypted Detection Mechanism
③ Encrypted Results
④ Decrypted results
② Programs
⑤ Detection Results
AC: The Agency Responsible for Creating Detection Rules
CP: Cloud Platform
PD: Programs to be Detected

Fig. 1. The model of the SMD-DE system

Specifically, the purpose and functionality of each component are described as follows.

1) AC refers to an industrial or academic entity that constructs and updates the welfare detection mechanism on the basis of a large scale of training data. AC provides services to the CP by sending encrypted malware detection rules to the CP.
2) CP offers unlimited storage space, and stores and manages the data in the proposed SMD-DE system. CP receives programs sent from the PD, analyzes the behavior of these programs, and performs malware detection upon the programs by using the encrypted malware detection mechanism sent to it from the AC.
3) PD simply refers to some (suspicious) programs that need to be inspected for the possibility of being malware. These programs will be sent to the CP for malware detection.
4) Note that the terms AC, CP, and PD described here correspond respectively to agency, server, and client mentioned at the beginning of Section 3.4.

4.2 The Proposed SMD-DE Mechanism

The SMD-DE mechanism consists of the following two modules: the PP-NBC construction and the secure malware detection. While appropriately chosen executable files (malware and benign programs) are used to train and construct the PP-NBC in the PP-NBC construction module, the action of detecting malware is performed in the malware detection module in a secure manner by utilizing the constructed PP-NBC.

4.2.1 The PP-NBC Construction Module

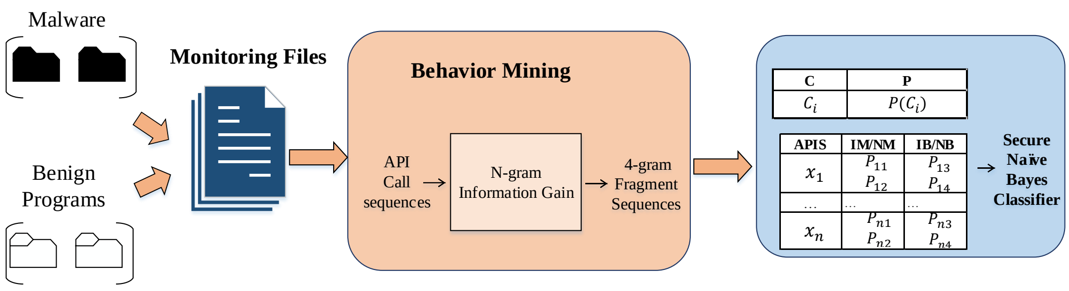

Fig. 2. Construction of the PP-NBC

The process of constructing the proposed PP-NBC is shown in **Fig. 2** and the PP-NBC is implemented in AC component of the SMD-DE system. Each input file (a malicious program or a benign program) is executed in a sandbox and the execution behaviors are logged into some monitoring files. By reading through these monitoring files, an API call sequence can be extracted from which a subsequent 4-gram-fragment sequence can be formed. The technique used to extract the 4-gram-fragment substrings and the call frequencies is the n-gram-based method [31] ($n = 4$ in our work), which is an effective means for classifying API calls and can anticipate the next API call in consecutive API calls based on statistical facts. An example of the 4-gram-based method is shown in **Fig. 3**.

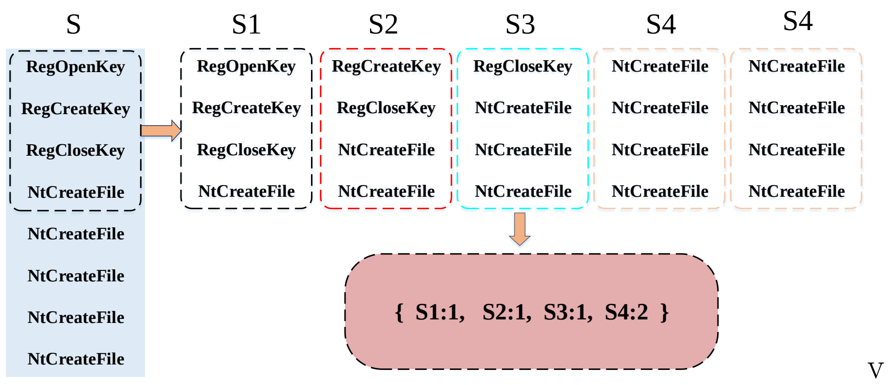

Fig. 3. An example of the 4-gram-based method

In **Fig. 3**, S is an original API call sequence. S1 is a 4-gram API call fragment formed by the top 4 API calls in S. The sliding window of the size of 4 slides from top to bottom along S, one API call at a time, resulting in four more 4-gram API call fragments: S2, S3, S4, and another S4. By counting the occurrence of each individual 4-gram in the resulted collection {S1, S2, S3, S4, S4}, we see clearly that S1, S2, and S3 all have frequency 1 but S4 has frequency 2.

The pseudocode algorithm for constructing the PP-NBC is given in Algorithm 1. Line 1 initializes two datasets H and D where H is used to store an initial sequence of 4-gram API call fragments and D the sequence of *unique* 4-gram API calls and their frequencies. Lines 2 and 3 describe the task of AC's executing the input files (malicious programs and benign programs), recording the behaviors of these programs into report files, and subsequently extract an API call sequence $S = (S_1, S_2, \dots, S_k)$ by analyzing these report files. Note that in practical situations, the size of S is typically hundreds of thousands, so the condition of $k \geq 4$ can be easily satisfied. Lines 5-8 extract all 4-grams API call fragments from S and put them into the dataset H, and Lines 10-12 initialize the frequency of each 4-gram API-call fragment in H to be (the default value) 1. Lines 14-23 subsequently extract, from H, one (unique) copy of each 4-gram API-call fragment and its occurrence frequency in H and write them into the dataset D. As stated in Line 25, AC builds two tables ∂_1 and ∂_2 (where ∂_2 is preliminary) by using the information contained in D. These two tables are conceptually illustrated by **Table 1** and **Table 2** below, respectively.

Algorithm 1: Construction of the PP-NBC

```
==============================================================
Input: Executable program files $E_1, E_2, \dots, E_q$.
Output: Encrypted tables $Enc(\partial_1)$ and $Enc(\partial_2)$.
1.  $H = \emptyset$; $D = \emptyset$;      // database initializations
2.  AC runs all executable program files and records the behaviors of the executions into
    analysis report files $R_1, R_2, \dots, R_q$;
3.  AC extracts API calls from these analysis report files and obtains an API call
    sequence $S = (S_1, S_2, \dots, S_k)$, where each $S_i$ denotes an API call;  // $k\ must \geq 4$
4.
5.  for ($i = 1$; $i \leq k - 3$; $i$++)
6.      $h_i$= take the subsequence $(S_i, \dots, S_{i+3})$ of $S$;  // a 4-gram API-call fragment
7.      add $h_i$ to $H$;  // construct $H$
8.  end for
9.
10. for ($i = 1$; $i \leq H.length$; $i$++)
11.     $count_i = 1$;        // default frequency for each $h_i$ in $H$
12. end for
13.
14. for ($i = 1$; $i \leq H.length$; $i$++)
15.     for ($j = 1$; $j \leq H.length$; $j$++)
16.         if ($i \neq j$ && $h_i = h_j$ && $h_i$ is not in $D$)
17.             $count_i = count_i + 1$;
18.         end if
19.     end for
20.     if $h_i$ is not in $D$
21.         add $h_i$ and $count_i$ to $D$;  // each 4-gram API fragment in $D$ is unique
22.     end if
23. end for
24.
25. AC builds Table $\partial_1$ and a preliminary version of Table $\partial_2$ by using the unique
    4-gram API fragments and their frequencies in $D$.
26.
27. AC computes the IG for each unique 4-gram API fragment by accessing the contents
    in $\partial_1$ and $\partial_2$. Assume the 4-gram API-call fragment portion of $D$ is the list
    $(x_1, x_2, \dots, x_n)$.
28. for each $x_j$ in $(x_1, x_2, \dots, x_n)$
29.     $IG(j) = \sum_{V_j \in \{0,1\}} \sum_{i=1}^{2} P(V_j, C_i) \log_2 \frac{P(V_j, C_i)}{P(V_j)P(C_i)}$      // info gain for each $x_j$
```

```
30. end for
31.
32. Sort the list $(x_1, x_2, \ldots, x_n)$ into descending order by the $IG(j)$ of each $x_j$.
33. AC finalizes Table $\partial_2$ by taking the first 500 $x_j$'s from the sorted list (coupled with
    their corresponding frequencies).
34.
35. for ($i = 1$; $i \le 2$; $i$++)
36.     $[P_i] = [F \log_2 P(C_i)]$    // encryption of $\partial_1$
37.     for ($j = 1$; $j \le n$; $j$++)
38.         $[P_{i,j}(x_j)] = [F \log_2 P(x_j | C_i)]$    // encryption of $\partial_2$
39.     end for
40. end for
================================================================
```

Table 1. ∂_1

C	**P**
C_i	$P(C_i)$

Table 2. ∂_2

APIS	**IM**	**NM**	**IB**	**NB**
x_1	P_{11}	P_{12}	P_{13}	P_{14}
x_2	P_{21}	P_{22}	P_{23}	P_{24}
…	…	…	…	…
x_n	P_{n1}	P_{n2}	P_{n3}	P_{n4}

In **Table 1**, C_i is either C_1 or C_2 where C_1 represents the malware class and C_2 the benign program class. In **Table 2**, the APIS column contains all 4-gram API-call fragments $(x_1, x_2, \ldots, x_n)$; the IM, NM, IB, and NB columns contain the probabilities of each x_j being in the malware class, not in the malware class, in the benign program class, and not in the benign program class, respectively. Lines 27-33 compute the information gain $IG(j)$ for each x_j by using the information in tables ∂_1 and ∂_2, reorder the list $(x_1, x_2, \ldots, x_n)$ from large to small by the information gain of each x_j, and finalize the table ∂_2 by replacing its contents with the first 500 x_j's from the sorted list (together with their associated IM, NM, IB, and NB data). Finally, Lines 35-38 encrypt the two tables ∂_1 and ∂_2. In particular, Line 36 encrypts the contents of table ∂_1 where the result of a constant F (typically 2^9) multiplying the logarithm of the probability of C_i is converted to an integer first (using the method described in [32]) and then encrypted by the Paillier encryption method ($[p]$ denotes a Paillier-encrypted integer p here). In a similar manner, Line 38 encrypts table ∂_2 where $P(x_j | C_i)$ means the probability of x_j with respect to class C_i.

4.2.2 The Malware Detection Module

As mentioned earlier in this section, AC sends the encrypted detection mechanism and the public key to CP which stores and manages the received encrypted detection mechanisms. Also, CP executes the programs that it receives from the PD in a virtual environment and generates the behavior files of the programs to be detected. The procedure of the malware detection is given in Algorithm 2 below where the inputs to the algorithm are user input file behavior sequence $X = (X_1, X_2, \dots, X_n)$, PP-NBC constructed by the AC, and the public encryption key PK, and the output is the class (malware or benign) of which the user input file will be classified.

Algorithm 2: Secure malware detection

```
==========================================================
Inputs: $X = (X_1, X_2, \dots, X_n)$, PP-NBC, Public key PK
Output: $C_{out}$   // the result of classifying the user input file
1.  CP receives the encrypted mechanism PP-NBC sent from AC;
2.  for $(i = 1;\ i \le 2;\ i{+}{+})$
3.      $[M_i] = [P_i]\prod_{j=1}^{n}[P_{i,j}(X_j = x_j)]$;
4.  end for
5.  CP transmits the encrypted data $[M_i]$ to AC for decryption;
6.  AC sends the decrypted data $M_i$ back to CP;
7.  if $(M_1 > M_2)$
8.      $C_{out} = C_1$;
9.  else
10.     $C_{out} = C_2$;
11. end if
==========================================================
```

Line 1 simply instructs the CP to receive the PP-NBC from the AC. Lines 2-4 compute the encrypted comprehensive malware indicator $[M_i]$ for each i. While Line 5 sends each encrypted $[M_i]$ to the AC for decryption, Line 6 sends the decrypted result back to the CP. Lines 7-11 compare the decrypted M_1 and M_2 and choose the class associated with the larger C_i as the output.

5. EVALUATION

In this section, we analyze the proposed security mechanism and present the experimental results on testing the proposed secure malware detection system.

5.1 Security Analysis

In the proposed SMD-DE system, AC holds the encrypted detection mechanism and PD holds the programs to be detected. Roughly speaking, since the malware detection mechanism PP-NBC, which is sent to the CP by the AC, is encrypted, it is highly unlikely that the construction details of the PP-NBC will be known to a third party even if it is intercepted during the transmission. Also, note that CP computes the detection results by using an encrypted PP-NBC and the output of the computation is also encrypted. Therefore, the information about the computation performed by the CP should be secure.

A detailed explanation regarding the security of the information flow in the SMD-DE system can be given as follows. The complete data that the AC sends to the PC is $(PK, x, [P_{i,j}(x_j)], C_i, [P_i])$ in which the public key PK, the 4-gram API-call fragment sequence x, and the classes C_i are not encrypted, but the attributes of the two tables ∂_1 and ∂_2 are encrypted, namely, $[P_i] = [F \log_2 P(C_i)]$ and $[P_{i,j}(x_j)] = [F \log_2 P(x_j | C_i)]$. As such, even if $[P_i]$ or $[P_{i,j}(x_j)$ was somehow seized by a malicious attacker during the transmission, the attacker will not be able to read the information in $[P_i]$ or $[P_{i,j}(x_j)]$ because it does not have the private key to decode them. Thus, the transmission of the PP-NBC from the AC to the CP is secure. After receiving $(PK, x, [P_{i,j}(x_j)], C_i, [P_i])$ from the AC and analyzing the programs to be detected from the user/client, CP starts the detection of the programs by computing $[M_i]$ using the encrypted $[P_i]$ and $[P_{i,j}(X_j = x_j)]$ as stipulated in Algorithm 2. Since the computation result $[M_i]$ is encrypted, CP needs to send $[M_i]$ to the AC for decryption. Once receiving the decrypted M_i from the AC, CP will use them to decide the detection result and send the result to the user/client. Hence, CP can complete the process of malware detection for any programs without knowing the actual internal mechanism of the malware detection. In other words, the computation of the malware detection performed at the CP is secure.

Regarding the actions and the malware detection rates of an homomorphically encrypted Naïve Bayes classier (NBC) and a regular (unencrypted) Naïve Bayes classifier, note the following property that the homomorphic encryption is a form of encryption that allows computations to be carried out on ciphertext thereby generating an encrypted result which, when decrypted, matches the result of operations performed on the plaintext [33]. In our proposed PP-NBC, while the frequency attributes of each x_j in the 4-gram API call fragment sequence $x = (x_1, x_2, \ldots, x_n)$ are homorphically encrypted, the sequence itself is unencrypted. When the API-call fragment sequence generated from the user input files is available, it will be matched against x and the corresponding computations will be performed using the encrypted frequency attributes yielding encrypted $[M_i]$'s which will be subsequently decrypted (see Algorithm 2) and used to produce the final result. According to the property of homomorphic encryptions just mentioned, we see that if $M_1 < M_2$ (or $M_1 > M_2$) is the result of the computation using a plaintext NBC, then $M_1 < M_2$ (or $M_1 > M_2$) is also the result of the computation using encrypted NBC, and vice versa. Therefore, in terms of malware detection rate or accuracy, the proposed PP-NBC has the same performance as its unencrypted version.

5.2 Experiment Setup

As stated before, we in this paper characterize the API calls as a sequence of 4-gram fragments. Part of the data regarding the sample files that are used to train the PP-NBC is displayed in **Table 3**, where the first column shows three 4-gram API-call fragments, the MalFre column shows their occurrence frequency in malware samples, the BenFre column shows their occurrence frequency in benign samples, and the last column shows their information gains. Through repeatedly testing the performance of the Naïve Bayes classifier by selecting the first 10, 20, 50, 100,150, …, 800 4-gram fragments in the ordered list, we find that the set of the first 500 4-gram fragments yields the optimal performance.

Table 3. Some sample 4-gram fragments and their attributes

4-gram Fragment x_j	MalFre	BenFre	IG
(LdrGetProcedureAddress, RegOpenKeyExW, RegQueryInfoKeyW, LdrGetProcedureAddress)	128	95	2.1351379
(NtSetInformationFile, NtCreateFile, NtSetInformationFile, NtQueryInformationFile)	221	20	1.9617122
(RegQueryValueExA, RegCloseKey, RegEnumKeyExA, RegCloseKey)	206	21	1.9173703

736 programs including 565 malware instances and 172 benign programs have been collected as sample files to train the Naïve Bayes classifier. While the malicious programs (e.g., Netsky, Mydoom, Bagel, etc.) are collected from the Vxheaven website, the benign programs comprise some Windows Portable Executables and other popular applications (e.g., Putty, Foxmail, CCleaner, FoxitReader_setup, etc.). Also, Ubuntu 12.04 is chosen as the operating system for the host machine to run the Cuckoo Sandbox, and Windows XP is used as the operating system for client machines.

5.3 Experimental Results

We evaluate the proposed malware detection mechanism by using the following commonly used metrics. Given the notions of true positive (the number of incidents where a malicious program is correctly identified as malicious), true negative (the number of incidents where a benign program is correctly identified as benign), false positive (the number of incidents where a benign program is incorrectly identified as malicious), and false negative (the number of incidents where a malicious program is incorrectly identified as benign), the concept of accuracy is defined as

$$accuracy = \frac{TP+TN}{P+N} \qquad (10)$$

where TP stands for the true positive, TN the true negative, P the sum of the true positive and the false negative, N the sum of the false positive and the true negative.

The k-fold cross validation [34] is a model validation technique that is widely used for assessing the effectiveness of malware detections. In k-fold cross validation, the original data set is randomly partitioned into k equal-sized subsets out of which $k-1$ subsets are randomly chosen as the training data and the remaining single subset is used as the testing data. In our work, we set $k=10$ based on empirical experiences. The resulting Receiver Operating Characteristic (ROC) curve, which shows the correlation between true positive rate (TP/P) and false positive rate (FP/N), is depicted in **Fig. 4**. By using **Fig. 4**, it can be calculated that the accuracy of our malware detection model is as high as 94.93%.

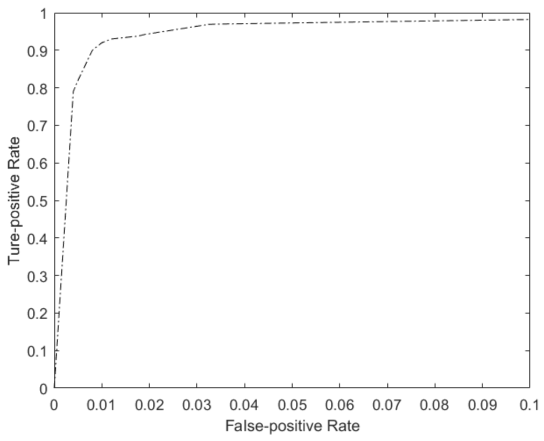

Fig. 4. The ROC curve of the malware detection mechanism

Table 4 compares our work with some of the peer investigations in the literature regarding the malware detections. Specifically, Saxe and Berlin [4] studied the malware detection using a static approach. Although this approach is effective, it cannot deal with some of the techniques (such as packing and obfuscation) that are commonly employed in malware programs. Elhadi, Maarof, and Barry [3] and Fan et al. [5] successfully utilized an API-call-based dynamic approach to overcome the problems of the static approach, but failed to consider the scenario of metamorphic or polymorphic malicious programs. While the issue of malware is addressed in [6] by devising some aggressive malware detection rules, it, unfortunately, falls short to be able to protect the malware detection mechanism itself. Aiming to tackle all these issues, we proposed a dynamic, and privacy-preserving malware detection mechanism. It can be seen from **Table 4** that our work, in addition to having these new features, has a compatible malware detection accuracy with peer studies.

Elhadi, Maarof, and Barry [3] and Fan et al. [5] successfully utilized an API-call-based dynamic approach to overcome the problems of the static approach. Morever, Maiorca [6] devise some aggressive malware detection rules, it, unfortunately, falls short to be able to protect the malware detection mechanism itself. Aiming to tackle all these issues, we proposed a dynamic, and privacy-preserving malware detection mechanism. It can be seen from **Table 4** that our work, in addition to having these new features, has a compatible malware detection accuracy with peer studies.

Table 4. Comparison of our study to other peer malware detection models

Peer work	**Approach**	**Malware-identifying technique**	**Malware antagonism considered?**	**Accuracy**	**Self-protection of detection mechanism?**
Saxe & Berlin [4]	Static	Contextual byte ; PE ; PE metadata; String 2d histogram	No	>94%	No
Elhadi et al. [3]	Dynamic	API call graph	No	>94%	No
Fan et al. [5]	Dynamic	API records in ARFF format	No	>94%	No
Maiorca [6]	Static and Dynamic	API references String	Yes	>94%	No
Our method	Dynamic	4-gramAPI fragment sequences	Yes	>94%	Yes

6. CONCLUSION AND FUTURE WORK

One of the prominent challenges in the area of malware detections is that most existing malware detection mechanisms are unable to effectively deal with the situations where a malware program attempts to antagonistically confuse the malware detection mechanism by first prying into the malware detection mechanism and then changing or obfuscating its code to generate new variations of the malware according to the information obtained. Toward resolving this issue and realizing that an efficient way to do so is to protect the malware detection mechanism, we proposed in this paper a secure malware detection system SMD-DE which consists primarily of a PP-NBC whose internal malware detection mechanism is encrypted by the Paillier homomorphic encryption system. Experimental results demonstrate that the proposed SMD-DE can correctly detect instances of malware with an accuracy of 94.93%.

As our future studies, we plan to investigate the detection of evasive malware and its related detection mechanism protection issues.

Acknowledgment

This work is supported by the National High Technology Research and Development Program of China (863 Program) (Grant No. 2013AA014702), the Fundamental Research Funds for the Central Universities (BUPT2016RC48, Grant 2014ZD03-03), National Natural Science Foundation of China (Grant No. 61601041).

References

[1] Grimes, R. Malicious mobile code: Virus protection for Windows. *" O'Reilly Media, Inc."*, 2001. Article (CrossRef Link)

[2] Biggio, B., Nelson, B., Laskov, P., "Poisoning attacks against support vector machines," in *Proc. of Langford, J., Pineau, J., editors, 29th International Conference on Machine Learning (ICML)*, pp. 1467-1474, June 26-July 01, 2012. Article (CrossRef Link)

[3] Elhadi, E., Maarof, M. A., Barry, B., "Improving the detection of malware behaviour using simplified data dependent api call graph," *International Journal of Security and Its Applications,* Vol. 7, No. 5, pp. 29-42, October, 2013. Article (CrossRef Link)

[4] Saxe, J., Berlin, K., "Deep neural network based malware detection using two dimensional binary program features," in *Proc. of 2015 10th International Conference on Malicious and Unwanted Software (MALWARE)*, pp. 11-20, October, 2015. Article (CrossRef Link)

[5] Fan, C. I., Hsiao, H. W., Chou, C. H., Tseng, Y. F., "Malware detection systems based on API log data mining," in *Proc. of 2015 IEEE 39th Annual Computer Software and Applications Conference (COMPSAC)*, Vol. 3, pp. 255-260, July, 2015. Article (CrossRef Link)

[6] Maiorca, D., "Design and implementation of robust systems for secure malware detection (Doctoral dissertation, Universita'degli Studi di Cagliari)," 2016. Article (CrossRef Link)

[7] Ye, Y., Wu, L., Hong, Z., and Huang, K., "A Risk Classification Based Approach for Android Malware Detection," *KSII Transactions on Internet and Information Systems*, vol. 11, no. 2, pp. 959-981, February, 2017. Article (CrossRef Link)

[8] Abdulla, S. and Altaher, A., "Intelligent Approach for Android Malware Detection," *KSII Transactions on Internet and Information Systems*, vol. 9, no. 8, pp. 2964-2983, August, 2015. Article (CrossRef Link)

[9] Xiao, X., Wang, Z., Li, Q., Li, Q., and Jiang, Y., "ANNs on Co-occurrence Matrices for Mobile Malware Detection," *KSII Transactions on Internet and Information Systems*, vol. 9, no. 7, pp. 2736-2754, July, 2015. Article (CrossRef Link)

[10] Schultz, M. G., Eskin, E., Zadok, F., Stolfo, S. J., "Data mining methods for detection of new malicious executables," in *Proc. of Proceedings of the 2001 IEEE Symposium on Security and Privacy*, pp. 38-49, May, 2001. Article (CrossRef Link)

[11] Frederick, R., "Core concept: homomorphic encryption," in *Proc. of Proceedings of the National Academy of Sciences*, Vol.112, no. 28, pp. 8515-8516, July, 2015. Article (CrossRef Link)

[12] Sun, Y., Wen, Q., Zhang, Y., Zhang, H., Jin, Z., "Efficient secure multiparty computation protocol for sequencing problem over insecure channel," *Mathematical Problems in Engineering 2013*, Article ID 172718, September, 2013. Article (CrossRef Link)

[13] Eskandari, M., Khorshidpur, Z., Hashemi, S., "To incorporate sequential dynamic features in malware detection engines," in *Proc. of Intelligence and Security Informatics Conference (EISIC)*, pp. 46-52, August, 2012. Article (CrossRef Link)

[14] Ye, Y., Wang, D., Li, T., Ye, D., "IMDS: Intelligent malware detection system," in *Proc. of Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1043-1047, August, 2007. Article (CrossRef Link)

[15] Canfora, G., Di Sorbo, A., Mercaldo, F., Visaggio, C. A., "Obfuscation techniques against signature-based detection: a case study," in *Proc. of Proceedings of 1st Workshop on Mobile System Technologies (MST)*, pp. 21-26, May, 2015. Article (CrossRef Link)

[16] Kim, M., Lauter, K., "Private genome analysis through homomorphic encryption," *BMC medical informatics and decision making*, Vol. 15, no. 5, S3, 2015. Article (CrossRef Link)

[17] Elhadi, A. A. E., Maarof, M. A., Barry, B. I., Hamza, H., "Enhancing the detection of metamorphic malware using call graphs," *Computers & Security*, Vol. 46, pp. 62-78, October, 2014. Article (CrossRef Link)

[18] Kocabas, O., Soyata, T., "Utilizing homomorphic encryption to implement secure and private medical cloud computing," in *Proc. of Cloud Computing (CLOUD), 2015 IEEE 8th International Conference*, pp. 540-547, June, 2015. Article (CrossRef Link)

[19] Sun, Y., Wen, Q., Zhang, Y., Li, W., "Privacy-preserving self-helped medical diagnosis scheme based on secure two-party computation in wireless sensor networks," *Computational and mathematical methods in medicine*, vol. 2014, pp. 9, July, 2014. Article (CrossRef Link)

[20] Kim, M., Lauter, K., "Private genome analysis through homomorphic encryption," *BMC Med Inform Decis Making*, 15(Suppl 5):3, December, 2015. Article (CrossRef Link)

[21] Yi, X., Okamoto, E., "Practical internet voting system," *Journal of Network and Computer Applications*, Vol. 36, no. 1, pp. 378-387, January, 2013. Article (CrossRef Link)

[22] Bunn, P., Ostrovsky, R., "Secure two-party k-means clustering," in *Proc. of Proceedings of the 14th ACM conference on Computer and communications security*, pp. 486-497, October, 2007. Article (CrossRef Link)

[23] Fun, T. S. and Samsudin, A., "A Survey of Homomorphic Encryption for Outsourced Big Data Computation," *KSII Transactions on Internet and Information Systems*, vol. 10, no. 8, pp. 3826-3851, August, 2016. Article (CrossRef Link)

[24] Kissner, L., Song, D., “Privacy-preserving set operations,” in *Proc. of Annual International Cryptology Conference*, Springer Berlin Heidelberg, pp. 241-257, August, 2005. Article (CrossRef Link)

[25] Xuezhen, H., Jiqiang, L., Zhen, H., Jun, Y., “A new anonymity model for privacy-preserving data publishing,” *China Communications*, Vol. 11, no. 9, pp. 47-59, November, 2014. Article (CrossRef Link)

[26] Barni, M., Failla, P., Kolesnikov, V., Lazzeretti, R., Sadeghi, A. R., Schneider, T., “Secure evaluation of private linear branching programs with medical applications,” in *Proc. of European Symposium on Research in Computer Security*, pp. 424-439, September, 2009. Article (CrossRef Link)

[27] Barni, M., Failla, P., Lazzeretti, R., Paus, A., Sadeghi, A. R., Schneider, T., Kolesnikov, V., “Efficient privacy-preserving classification of ECG signals,” in *Proc. of First IEEE International Workshop on #Information Forensics and Security*, pp. 91-95, December, 2009. Article (CrossRef Link)

[28] Bos, J. W., Lauter, K., Naehrig, M., “Private predictive analysis on encrypted medical data,” *Journal of biomedical informatics*, Vol. 50, pp. 234-243, August, 2014. Article (CrossRef Link)

[29] Bost, R., Popa, R. A., Tu, S., Goldwasser, S., “Machine Learning Classification over Encrypted Data,” in *Proc. of The 22nd Internet Society Annual Network and Distributed System Security Symposium*, pp. 8-11, February, 2015. Article (CrossRef Link)

[30] Paillier, P., “Public-key cryptosystems based on composite degree residuosity classes,” in *Proc. of International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 223-238, May, 1999. Article (CrossRef Link)

[31] Lee, T., Choi, B., Shin, Y., Kwak, J., “Automatic malware mutant detection and group classification based on the n-gram and clustering coefficient,” *The Journal of Supercomputing*, pp. 1-15, December, 2015. Article (CrossRef Link)

[32] Tschiatschek, S., Pernkopf, F., “On Bayesian network classifiers with reduced precision parameters,” *IEEE transactions on pattern analysis and machine intelligence*, Vol. 37, no. 4, pp. 774-785, August, 2015. Article (CrossRef Link)

[33] X. Yi, R. Paulet, E., “Bertino. homomorphic Encryption and Applications,” *Springer International Publishing*, 2014. Article (CrossRef Link)

[34] Zhang, Y., Wang, S., Phillips, P., Ji, G., “Binary PSO with mutation operator for feature selection using decision tree applied to spam detection,” *Knowledge-Based Systems*, Vol. 64, pp. 22-31, July, 2014. Article (CrossRef Link)

Zhaowen Lin was born in Jieyang, Guangdong, China in 1979. He received his doctor degree from Beijing University of Posts and Telecommunications(BUPT). He is currently an associate professor, doctoral supervisor in BUPT. His research fields concern network architecture, network management and security.

Fei Xiao was born in Linyi, Shandong, China in 1989. She is a Ph.D. candidate in Beijing University of Posts and Telecommunications(BUPT) of Computer Sciences and Technology. Her research interests include network security and privacy-preserving data mining.

Yi Sun was born in Enshi, Hubei, China in 1987. She received her doctor degree from State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications in June, 2015. Now she is a lecturer of Institute of Sensing Technology and Business, Beijing University of Posts and Communications, China. Her research interests include information security, privacy-preserving data mining, secure multiparty computation, healthcare big data.

Yan Ma was born in Beijing, China in 1955. He received his bachelor degree from Beijing University of Posts and Telecommunications(BUPT). He is currently a professor in BUPT. His research fields concern network architecture, network management and security.

Cong-Cong Xing is Professor of Computer Science/Mathematics at Nicholls State University, Thibodaux, Louisiana, USA. He received his Ph.D. in Computer Science and Engineering from Tulane University, New Orleans, USA, joining the Nicholls State University faculty in 2001. His research interests include theoretical foundations of programming languages, category theory, mobile/wireless computing and analysis, and malware detections. He is active in research in these areas.

Jun Huang (M’12-SM’16) received the Ph.D. degree from the Institute of Network Technology, Beijing University of Posts and Telecommunications, China, in 2012. He is a professor of communication and information system with the Chongqing University of Posts and Telecommunications. He was a visiting scholar in the Global Information and Telecommunication Institute, Waseda University, a research fellow in the Electrical and Computer Engineering Department, South Dakota School of Mines and Technology, a visiting scholar in the Computer Science Department, the University of Texas at Dallas, and a guest researcher at the National Institute of Standards and Technology. He received a runner-up of best paper award from ACM SAC 2014 and a best paper award from AsiaFI 2011. He has authored more than 90 publications including papers in prestigious journal/conferences such as the IEEE Network, the IEEE Communications Magazine, the IEEE Wireless Communications Magazine, the IEEE Transactions on Broadcasting, the IEEE Transactions on Vehicular Technology, the IEEE Transactions on Emerging Topics in Computing, the IEEE Transactions on Sustainable Computing, the IEEE Internet of Things Journal, the IEEE Transactions on Cloud Computing, IWQoS, SCC, ICCCN, GLOBECOM, ICC, ACM SAC, and RACS. He is an associate editor of IEEE Access and the KSII Transactions on Internet and Information Systems. He chaired and co-chaired multiple conferences in the communications and networking areas and organized multiple workshops at major IEEE and ACM events. His current research interests include network optimization and control, machine-to-machine communications, and Internet of Things. He is a senior member of the IEEE.