IEEE Access
Multidisciplinary ⁞ Rapid Review ⁞ Open Access Journal

Received June 24, 2019, accepted July 22, 2019, date of publication July 25, 2019, date of current version August 22, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2931136

Machine Learning Based File Entropy Analysis for Ransomware Detection in Backup Systems

KYUNGROUL LEE[1], **SUN-YOUNG LEE**[2], **AND KANGBIN YIM**[2]

[1]R&BD Center for Security and Safety Industries, Soonchunhyang University, Asan 31538, South Korea
[2]Department of Information Security Engineering, Soonchunhyang University, Asan 31538, South Korea

Corresponding author: Kangbin Yim (yim@sch.ac.kr)

This work was supported in part by the National Research Foundation of Korea (NRF) grant funded by the Korean Government (MSIT) under Grant 2018R1A4A1025632, and in part by the Soonchunhyang University Research Fund.

ABSTRACT With the advent of big data and cloud services, user data has become an important issue. Although a variety of detection and prevention technologies are used to protect user data, ransomware that demands money in exchange for one's data has emerged. In order to detect and prevent ransomware, file- and behavior-based detection methods have been investigated. Nevertheless, we are still facing from ransomware threats, as it is difficult to detect and prevent ransomware containing unknown malicious codes. In particular, these methods are limited in that they cannot detect ransomware for backup systems such as cloud services. For instance, if files infected with ransomware are synchronized with the backup systems, the infected files will not be able to be restored through the backed-up files. In this paper, we utilize an entropy technique to measure a characteristic of the encrypted file (i.e., uniformity). Machine learning is applied for classifying infected files based file entropy analysis. The proposed method can recover the original file from the backup system by detecting ransomware infected files that have been synchronized to the backup system, even if the user system is infected by ransomware. Conducted analysis results confirm that the proposed method provides a high detection rate with low false positive and false negative rates compared with the existing detection methods.

INDEX TERMS Backup system, artificial intelligence, machine learning, malicious code detection, ransomware, entropy, data reliability, data security.

I. INTRODUCTION

With the advent of big data and cloud services, user data has become an important element, and ensuring reliability and integrity of user data is one of the core requirements for these services. As these services prioritize protecting user data, attackers have attempted to attack data and hold it hostage in order to demand money, and this kind of attack is called ransomware [1]. Ransomware is a compound word of ransom and malware, and is an attack technique that penetrates the system and then encrypts the user's files to prevent the user from being able to read them or locks the system entirely. Once the attacker seizes the system, the attacker then demands money for decrypting the encrypted file or unlocking the system [2], [3].

Such ransomware attacks are not new attacks, as they first appeared in 1989 in the form of "PC CYBORG/AIDS information Trojan". The ransomware that appeared at this time was not used much by attackers before being turned off and discarded for almost 15 years. However, with the recent expansion of the Internet usage throughout the population as well as the emergence of e-commerce and big data and cloud services, ransomware was revitalized in 2005 [4] with examles including PC BYBORG, Reveton [5], CryZip [6], May Archieve [6], FAVEAC [7], FastBsod [7], CryptoLocker [8], GPCoder [6], SimpleLocker [5], TeslaCrypt [9], CryptorBit [10], KeRanger [11], and CryptoWall [12].

Despite the fact that many types of ransomware have appeared and caused serious damages, one of the reasons that it is difficult to protect a system from ransomware is that it is difficult to detect and prevent unknown ransomware for the following reasons: first, after ransomware penetrates and infects a system, the user cannot recover the system and files because there is no decryption key. Moreover, it is also difficult for users and anti-virus software to detect ransomware when attackers conceal their penetration behaviors and control of the system. For example, attackers can hide their identity using the Tor network, which is an anonymous network. In addition, even if the user does pay the required

The associate editor coordinating the review of this manuscript and approving it for publication was Chi-Yuan Chen.

amount of money, it can cause secondary damage by failing to decrypt the data or unlock the system. Serious damages have been caused by various types of ransomware attacks, and attackers generated about 27 million dollars in revenue according to ZDNet in 2013 [14]. Therefore, it is urgent to detect ransomware in the early stages and restore the files or systems infected by ransomware.

In order to solve these problems, various methods have been investigated to detect ransomware and prevent it from infecting systems, and these can be classified into four categories: file-based detection, system-based behavior detection, resource-based behavior detection, and connection-based behavior detection. The file-based detection method detects malicious signatures in files of a particular format. However, this method is limited in that it cannot detect unknown ransomware [12]. The system-based behavior detection method blocks ransomware by monitoring malicious behavior of the system and detects ransomware by verifying the integrity of files and directories. This method lead to high false positives and false negatives, and the integrity verification and behavior monitoring are time-consuming [5]. The resource-based detection method detects ransomware based on the usage rate of resources consumed for encrypting files. However, this method requires a lot of time to obtain the usage of resources such as CPU and I/O, and has the disadvantages of high rates of false positives and false negatives [5]. The connection-based behavior detection method is a way to detect and prevent ransomware by monitoring the connection to the outside, because ransomware must receive the encryption key from the server. However, this method does not detect ransomware that does not connect to the network [10]. Therefore, most the currently used methods have high false positives and false negatives, indicating a failure to effectively detect ransomware, and they cannot detect ransomware for backup systems such as cloud services.

Ransomware infections in general systems such as PCs lead to serious problems. For instance, when all of a user's files are encrypted by a ransomware attack, the users will not be able to recover their files, and if an attack locks their system, it will prevent the system from running. Therefore, it is possible to protect the system by detecting and preventing several types of ransomware using the existing detection methods described above. Nevertheless, most existing methods are limited by only considering ransomware detection in general systems. In other words, the methods do not consider files that are synchronized to the backup system such as cloud, and serious problems arise due to the inability to detect files infected with ransomware that are delivered to the backup system. The primary purpose of using a backup system is for a user to back up their files. If a user's files are encrypted by ransomware, the user can restore their original files by synchronizing or copying files from backup systems including cloud services such as Dropbox and Google One Drive, USB storage, and external disks. Nevertheless, if the files infected by ransomware are synchronized to the backup system, the files cannot be restored through the backed up files. For example, if the system data is infected by ransomware and then the encrypted file is transferred to the backup system, the original file will be replaced with the encrypted file. After the file is replaced, the user cannot restore the original file from the backup system.

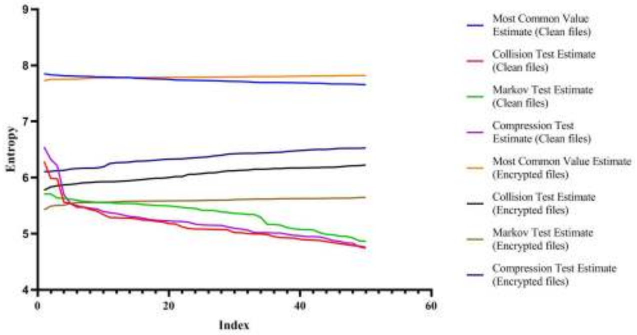

FIGURE 1. Comparison result of the entropy of compressed files (50 clean files, 50 encrypted files).

Thus, it is crucial to have ability to recover files that have been backed up by determining whether or not the file being synchronized to the backup system is infected by ransomware. In order to solve this problem, this paper proposes a method to detect files infected with ransomware based on the entropy of the files. The proposed method uses a feature that appears in encrypted files based on the behavior of the ransomware encrypting the files. One of the features of the cipher text is uniformity. In this paper, entropy is used as one of the methods to measure uniformity. Entropy can be measured using a variety of methods, with NIST 800-90b being representative among them.

In order to evaluate the proposed ransomware detection method, we compared the entropy of the original files and the encrypted files for files that are crucial to the system and users, such as system files, document files, image files, source code files, executable files, and compressed files. By measuring the entropy of these files, it is relatively easy to detect ransomware in system files, source code files, and executable files. However, some of the entropy of document files, image files, and especially compressed files, is lower than the entropy of the clean files. This means that it is difficult to effectively detect ransomware in these files, as shown in Fig. 1. For this reason, in this paper, to solve the problem of detecting infected files by ransomware which is synchronized to the backup system by defining the entropy reference value, a method for accurately and efficiently classifying the infected files by file formats is derived using machine learning. Namely, the goal of this paper is to verify and select an optimal model based on various machine learning models.

The contributions of this paper are as follows.

- The existing ransomware detection methods do not detect ransomware-infected files in the backup system. However, this paper effectively detects files infected with ransomware delivered to the backup system in real time by using the reference value derived through machine learning based on entropy according to different file formats. This method detects files stored in the backup system safely and allows the user to restore their original files from the backup system, even if the user system is infected by ransomware.

- The existing ransomware detection methods have a low detection rate as well as high false positives and false negatives. Moreover, these also require collecting resource and network information, and they have a disadvantage of consuming a large amount of resources due to having to monitor all of the activities in the system. However, the proposed method can detect ransomware-infected files based on the entropy of the file transferred to the backup system and can continuously apply the optimal reference value using machine learning. Therefore, its detection rate is much higher than those of the existing detection methods, and it also has advantages of low false positives and false negatives.
- The proposed method is extremely effective and accurate because it selects the optimal model to detect the infected files by ransomware based on various machine learning models and derives the optimal entropy reference value. Moreover, the backup system has an artificial intelligent element by automatically applying the entropy reference value and the model based on the specific user, and this feature is expected to lead to remarkable developments in ransomware detection.

The structure of this paper is as follows. In section 2, we describe the prior knowledge required for the proposed ransomware detection method, while section 3 describes the system configuration and data classification. In section 4, we describe the experimental results of the detection of files infected with ransomware. Finally, section 5 concludes the paper.

II. PRIOR KNOWLEDGE

This section describes the prior knowledge required for the proposed ransomware detection method and also describes entropy, which is the essential information used to detect ransomware-infected files, and the methods used to measure entropy. As we use machine learning to classify clean files and infected files by ransomware based on the measured entropy, we describe the necessary machine learning models. The goal of this paper is to derive a suitable machine learning model. The considered machine learning models include K-Nearest Neighbors (KNN), linear model, decision tree, decision tree ensemble, kernel trick, and deep learning. In addition, model evaluation results are derived by evaluating these models. The evaluation models include cross validation, cross validation splitter, Leave-One-Out (LOOCV), and shuffle split cross validation.

A. ENTROPY MEASUREMENT METHODS

One method of evaluating the performance of cipher text generated from cryptography involves measuring entropy. Entropy is a measure of uniformity, which represents the disorder or randomness of the generated cipher text. For instance, if the value of the cipher text is in the range from 0×00 to $0\times\mathrm{FF}$, the probability of each generated value should be the same [16]. If there is a statistical bias, such as toward a certain value or range of values, the vulnerability that can be decrypted based on the probability of a value being generated with a higher or lower value occurs [17]. In order to solve this vulnerability, the encryption technology is designed so that the probability of the occurrence of each value generated by the cipher text is almost the same, which is called uniformity.

Uniformity can be measured based on entropy, and the available measurement methods include Poisson distribution [18], hamming distance [19], and spontaneous emission [20]. Among them, NIST published NIST 800-90b and provided methods and tools that can be used to measure [21]. According to the measurement methods used, these can be classified into a statistic-based measurement method and a predictor-based measurement method [22]. In this paper, we use statistic-based measurement methods to speed up the measurement: these are the most common value estimate, the collision test estimate, the Markov test estimate, and the compression test estimate [23]. The methods measure entropy based on frequently occurring probabilities, random repetitive patterns, and dependencies between successive values. The most common value estimate obtains entropy by using the probability that a value will appear frequently in the input data set, shown as Equation (1). The collision test estimate defines arbitrary repetitive patterns as collisions Equation (2) and estimates the probability of output values that appear often based on the times when collisions occur. The Markov test estimate measures the dependence between successive values from a set of input data Equation (3), and the compression test estimate measures the entropy rate based on the compression capacity of the data set Equation (2) [21]. The entropy measurement results for each file format are shown in Fig. 2.

$$min - entropy = -\log_2(pu) \tag{1}$$

$$min - entropy = -\log_2(p)\ or\ \log_2(k) \tag{2}$$

$$min - entropy = -1/d\log_2(pmax) \tag{3}$$

For most types of files, the most common value estimate has the highest entropy while the Markov test estimate has the lowest entropy. Examining the entropy results by file formats yields that the source code files have the lowest entropy whereas the document files and compressed files have the highest entropy.

B. MACHINE LEARNING MODELS

This section describes the machine learning models used to detect files infected by ransomware based on the entropy measurement methods described above. The machine learning models used in this paper include KNN, linear model, decision tree, decision tree ensemble, kernel trick, and neural network.

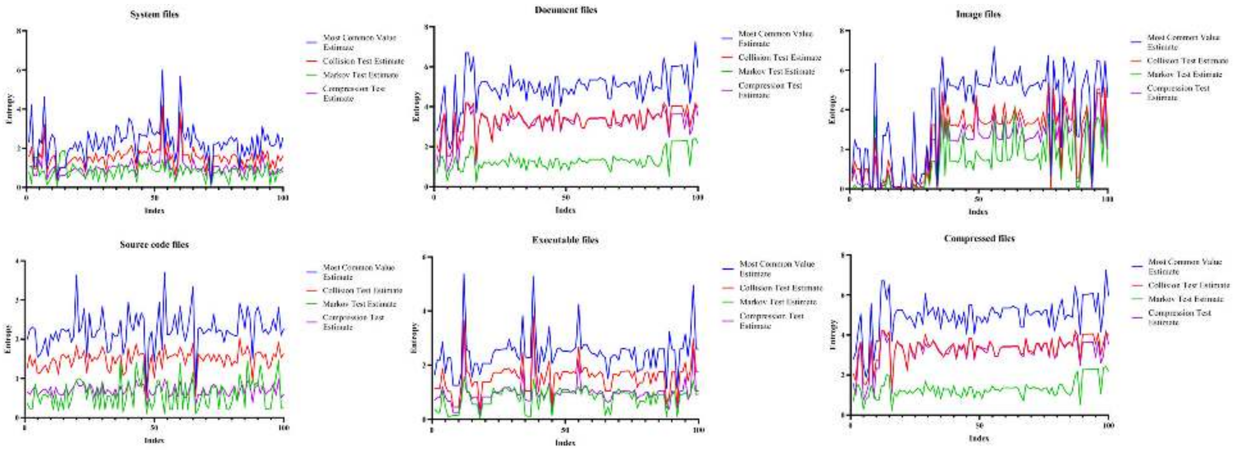

FIGURE 2. Entropy measurement results by file formats.

1) KNN

KNN uses one nearest training data point to identity the closet neighbors and use it for prediction [24]. This model classifies arbitrary k neighbors and designates classes with more neighbors as a label based on the number of neighbors belonging to each class. Therefore, the decision boundary is divided into the area designated by each class, then the data are classified based on the boundary. The important parameters in this model are the number of neighbors and how the distance between data points is estimated, and the performance of the model depends on these parameters.

2) LINEAR MODEL

The linear model performs the prediction using the linear function of the input features. The generalized prediction function is as follows.

$$\hat{y} = w[0]\times x\,[0] + w[1]\times x\,[1] + \ldots + w[p]\times x\,[p] + b \quad (4)$$

where $x[0]$ to $x[p]$ are characteristics for one data point while w and b are parameters to be learned by the model; namely, w is the slope and b is the intercept that meets the y-axis. $\hat{y}$ is the predicted value generated by the model. This model includes various regression models such as linear regression, Ridge regression, Lasso regression, logistic regression, and Linear Support Vector Classified (SVC). Linear regression identifies parameters w and b that minimize the mean squared error between the target y in the prediction and training sets [25]. The mean squared error is obtained by adding the difference between the predicted value and the target value, then dividing the result by the number of samples, as shown in Equation (5).

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(yi - \hat{y}i)2 \quad (5)$$

where n is the number of samples. The slope parameter w is referred to as a weight or a coefficient, while the parameter b is referred to as an offset or an intercept. However, this model is limited by the fact that it cannot control the complexity. For this reason, the Ridge regression model is used to control complexity.

The Ridge regression model involves minimizing the absolute value of the weight w to the extent possible, which minimizes the effects of all characteristics on the output [26]; this is called regularization. In other words, this means that the model is constrained to not be overly fit, and the regulatory method used in this model is called L2 regulation. This model applies the square of the L2 norm of the coefficient as a penalty, as shown in Equation (6).

$$Ridge = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)2 + a\sum_{j=1}^{m}w_j^2 \quad (6)$$

Here, if a is increased, the effect of the penalty is increased by decreasing the weight. Otherwise, if a is decreased, the effect of penalty is decreased with increasing weight.

The Ridge regression model and Lasso regression model can be used to apply regulation to the linear regression model. The Ridge regression model uses L2 regulation while the Lasso regression model uses L1 regulation [27]. As shown in Equation (7), the Lasso regression model uses the L1 norm of the coefficient vector, which is the sum of the absolute values of the coefficients, as a penalty.

$$Lasso = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)2 + a\sum_{j=1}^{m}|w_j| \quad (7)$$

In this model, like in the Ridge regression model, if a is increased, the effect of the penalty is increased with decreasing weight. Otherwise, if a is decreased, the effect of the penalty is decreased with increasing weight. Generally, the Ridge regression model is preferred over the Lasso regression model. Nevertheless, when there are a lot of features and only a few of them are important, the Lasso regression model might be better.

Some examples of linear classifier models are logistic regression models and the linear support vector machine model [28], [29]. These models are classifiers that classify two classes using lines, planes, and hyperplanes, as shown in Equation (8).

$$\hat{y}=w[0]\times x\,[0]+w[1]\times x\,[1]+\ldots+w[p]\times x\,[p]+b>0 \tag{8}$$

These models compare the weighted sum of the features with the threshold zero. If the result is less than 0, the class is predicted as -1, and if it is greater than 0, it is predicted as $+1$; finally, the class is classified based on the prediction result.

3) DECISION TREE

The decision tree model is widely used for classification and regression problems, and continues learning yes or no questions to reach a decision [30]. This model divides the data into a list of yes or no questions, and the data are repeatedly divided until each divided area (leaf of the decision tree) has one target value (one class or one regression analysis result). A leaf node consisting of only one target is called a pure node. The disadvantage of this model is that if the model splits the data until all of the leaf nodes become pure nodes, the model becomes very complex and overfitting. There are two ways to control overfit: pre-pruning and post-pruning. Pre-pruning is a strategy that discontinues the tree generation early, while post-pruning is a strategy that deletes or merges with fewer data points following tree generation.

4) DECISION TREE ENSEMBLE

Ensemble is a technique for linking multiple machine learning models to create a more robust model [31], and this model has been proved to be effective in various data sets of classification and regression problems. Some examples of the decision tree-based ensemble models are the ransom forest model and the gradient boosting decision tree model. The random forest model is a model that calculates the average results after bundling several decision trees in order to address the limitation that the decision tree is overfit to the training data [32]. There are several ways to generate different decision trees, with two common ones involving a random selection of data points and a random selection of features in split tests. The gradient boosting decision tree model generates trees sequentially in a manner that complements the errors of the previous tree, and this model is not random [33]. Instead, it uses powerful pre-pruning, and it is a good model according to its performance because it usually uses five deep trees.

5) KERNEL TRICK

A well-known kernel trick is the kernel support vector machine [34]. This model is an extension of the input data to generate a more complex model that cannot be defined as a simple hyperplane. For this reason, straight lines and hyperplanes are not flexible, and linear models are very limited for low dimensional data sets. In order to solve this problem, a new feature is added by multiplying features or exponentiating features. However, there is a disadvantage in that the computation cost is increased as the features increase. In order to reduce the computational cost, there is a kernel trick that can learn the classifier at a high dimension without generating many new features. This model does not actually extend the data, but instead calculates the distance (specifically, the scalar product) between the data points to the extended features. Based on the results, the SVM classifies each training data point as the decision boundary between the two classes, and the data points located at the boundary are called support vectors. Thus, in order to predict for a new data point, this model measures the distance to each support vector, which is computed by the Gaussian kernel. The equation used to obtain the distance is shown in Equation (9).

$$k_{rbf}(x_1, x_2) = \exp(-\gamma||x_1 - x_2|| \tag{9}$$

where x1 and x2 are data points and $||x1\text{-}x2||$ is the Euclidean distance. Gamma γ is a parameter that controls the width of the Gaussian kernel. This model determines the range of influence of one training sample with the gamma parameter. Small values indicate large areas, while large values have a limited range. The C parameter limits the Euclidean distance, which is a regulatory parameter similar to the parameter described in the linear model, and this limits the importance of each point.

6) NEURAL NETWORK (DEEP LEARNING)

In this paper, we utilize Multi-Layer Perceptrons (MLP), a neural network model that is relatively easy to use for classification and regression [34]. This model repeats the process of generating a weighted sum, which is a process from the linear regression model shown in Equation (4), to the output many times, as opposed to just once. This model is a model that computes the hidden unit which composes the intermediate stage and calculates the weight sum again. Such a hidden unit constitutes a hidden layer, and the deep running is performed due to the hidden layer. The most important parameter in this model is the number of units in the hidden layer, and the disadvantage is that the computation cost increases with an increasing number of hidden units.

C. MODEL VALIDATION

This section describes a method for evaluating the machine learning models. The model validation methods used in this paper include cross-validation, cross-validation splitter, LOOCV, and shuffle split cross-validation.

Cross-validation involves dividing data several times and then learning various models [35]. The most widely used cross-validation method is k-fold cross-validation, which typically uses five or ten folds. For example, if five-fold cross-validation, this method divides the subset of almost similar size into five subsets, then creates a series of models. The first model uses the first fold as a test set, while the remaining folds (2 through 5) as training sets. This validation method is a verification method that repeats up to the fifth model. Under these conditions, there is a case where a set is merely classified and a wrong validation result is derived. In order to solve this problem, stratified k-fold cross validation is used. This validation method classifies the data so that the class ratio in the fold is equal to the class ratio of the entire data set, which is referred to as the cross-validation method including splitter.

Leave-one-out cross validation is a k-fold cross validation method including only one sample in a fold [36]. This method selects one data point for each iteration and then uses it as a test set.

A highly flexible cross validation strategy is shuffle split cross validation. This method divides a certain size into training sets and test sets, where each split is a certain number of iterations. This method is useful when the number of iterations must be adjusted independently of the size of the training sets or test sets.

III. PROPOSED SYSTEM CONFIGURATION AND DATA CLASSIFICATION

The proposed ransomware detection system is shown in Fig. 3. For detection, the backup system measures entropy according to the different file formats of users and derives a reference value for detecting files infected by ransomware using machine learning models based on entropy. The reference values derived for each user and for each file format are then transmitted to the user's client software. Each user's client software has a built-in ransomware detection module, and the detection module measures the entropy of the files synchronized to the backup system. Through comparing the entropy of the files and the entropy reference value received from the backup system, the module can detect files infected with ransomware. Moreover, if sufficient system resources are available, machine learning models can be included in the detection module, and the synchronized files are used as a test set to detect files infected by ransomware.

The backup system stores the backup data of each user. In order to detect files infected with ransomware, the entropy reference value of each user's backup data is measured. The entropy reference value has a different reference value for each file format, so the entropy of each file format should be measured. The entropy measurement methods use the most common value estimate, the collision test estimate, the Markov test estimate, and the compression test estimate, which are all relatively fast.

The backup system uses machine learning models to identify files infected by ransomware and to derive the entropy reference value. The machine learning models used in this paper are KNN, linear model, decision tree, decision tree ensemble, kernel trick, and deep learning. Based on these models, the proposed system first extracts the optimal entropy reference value for each file format by performing learning and classification for each user, and it also has an artificial intelligence element as it continuously extracts and updates the reference value. Next, the extracted entropy reference value is passed to the user's client software. The ransomware detection module embedded in the client software then measures the entropy of files synchronized to the backup system. The detection module detects infected files by comparing the measured entropy of the synchronized files with the reference value of the file format received from the backup system. Moreover, it is also a good idea to determine the entropy reference value as the average of a large number of files. Nevertheless, different characteristics of entropy of user files may be derived for each user. Therefore, the backup system learns and measures the entropy of the files stored in the backup system for each user, thereby deriving an optimal reference value specific to the user files, which leads to more accurate detection of the files infected by ransomware.

The data used for machine learning models in the proposed system consists of file format, the entropy measured

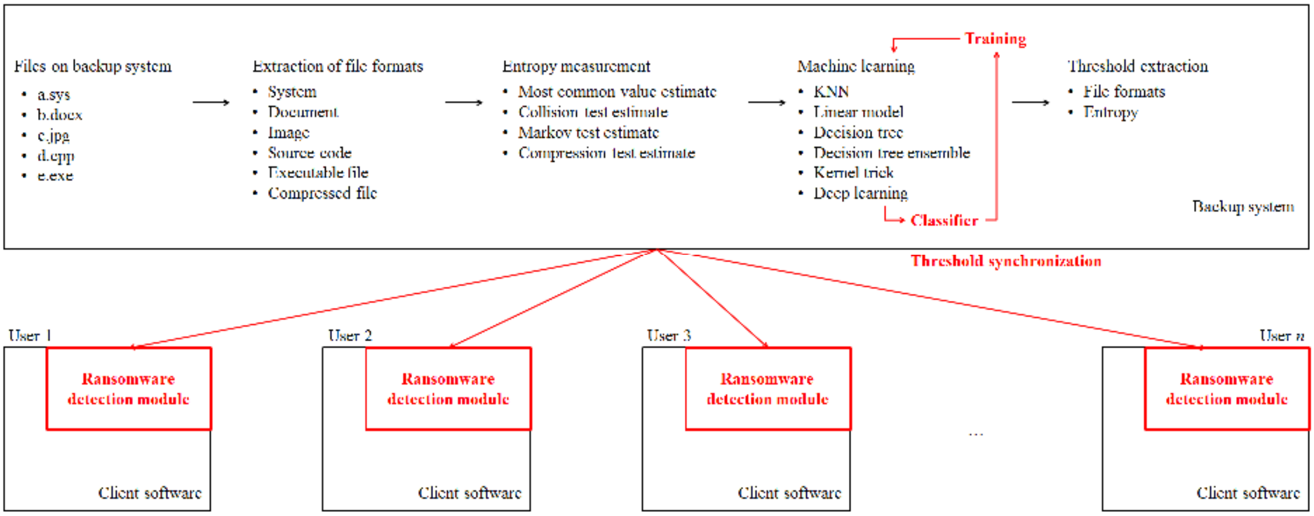

FIGURE 3. Proposed ransomware detection system.

TABLE 1. Data configuration.

File format	Most common value estimate	Collision test estimate	Markov test estimate	Compression test estimate	Ransomware-infected file
1	5.978333	4.155263	1.414193	4.0879	0
1	7.649653	5.747665	5.378375	6.015537	1
2	7.233733	4.141459	2.421299	4.196178	0
2	7.634892	6.46745	5.388627	6.315642	1
3	7.164502	4.242225	3.577339	3.924578	0
3	7.69319	5.805663	5.504605	6.345644	1
4	3.701553	1.852332	0.109477	0.965185	0
4	7.74035	6.073426	5.585261	6.252423	1
5	5.360685	3.662946	1.576247	3.363707	0
5	7.820584	5.928158	5.631384	6.535997	1
6	7.790195	4.719231	4.873316	4.655534	0
6	7.750831	6.423027	5.598241	6.748249	1

TABLE 2. Training set score and test set score.

Machine learning model		Parameters	Training set score	Test set score
KNN		K=3	1.0	1.0
Linear model	Linear regression	-	0.91	0.90
	Ridge regression	alpha=10	0.91	0.89
	Lasso regression	alpha=1, max_iter=100000	0.16	0.16
	Logistic regression	C=0.01, l1 regularization	0.912	0.887
	Linear SVC	-	1.0	1.0
Decision tree		depth=4	0.998	0.997
Decision tree ensemble	Random forest	n_estimator=2	0.996	0.987
	Gradient boosting	max_depth=1	0.998	0.997
Kernel trick	SVM	C=1000	1.0	1.0
Deep learning	MLP	alpha=1, max_iter=1000	1.0	1.0

by the most common value estimate, the entropy measured by the collision test estimate, the entropy measured by the Markov test estimate, the entropy measured by the compression test estimate, and whether or not the file is infected by ransomware (0/1).

The file formats are denoted as follows: system files, 1; document files, 2; image files, 3; source code files, 4; executable files, 5; and compressed files, 6. The entropy is measured on the basis of 8 bits, and the results have a value ranging from 0 to 8. The existence of ransomware infection is denoted as 0 for a clean file and 1 for an encrypted file. Entropy is measured using data collected for a total of 1,200 files consisting of 100 files for each file format and 100 encrypted files for each file format. Examples of the collected data and the results of entropy measurement are shown in Table 1.

IV. EXPERIMENT RESULT

This section presents the experimental results based on the machine learning models described in Section 2, the model validation, and the results of the performance evaluation.

A. SUPERVISED MACHINE LEARNING RESULTS

Random numbers of training sets and test sets were classified for supervised machine learning, and Table 2 shows the stores of the training sets and the test sets. As a result, KNN, linear SVC, SVM, and MLP have completely learned the data in both the training set and the test set. Decision tree, random forest, and gradient boosting tree have learned the data at a very high level, while linear regression, Ridge regression and logistic regression have learned the data at a high level. On the other hand, Lasso regression has shown very poor learning stores.

Most of the parameters used in Table 2 are set to default values, and the performances of the machine learning models depend on the parameters, so we derive the parameters with the best scores. In order to ensure more accurate measurement, training sets, validation sets, and test sets were included and their scores are shown in Table 3.

As shown in Table 3, the results indicated that the scores of Lasso regression, logistic regression, decision tree, random forest, and gradient boosting tree were improved. Specifically, Lasso regression increased the training set score from 0.16 to 0.90, representing an improvement of 562%. The test set score increased from 0.16 to 0.88, an improvement of 550%.

Logistic regression improved the training set score by 109% from 0.91 to 1.0 while the test set score improved by 112% from 0.887 to 1.0. The decision tree improved the training set score by 100.2% from 0.998 to 1.0 while the test set score improved by 100.3% from 0.997 to 1.0. In the random forest, the training set score improved 100.4% from 0.996 to 1.0 while the test set score improved 101% from 0.987 to 1.0. The gradient boosting tree improved the

TABLE 3. Training set score, validation set score, and test set score with optimal parameters.

Machine learning model		Parameters	Training set score	Validation set score	Test set score
KNN		K=3	1.0	1.0	1.0
Linear model	Linear regression	-	0.91	0.91	0.90
	Ridge regression	alpha=10	0.91	0.91	0.89
	Lasso regression	alpha=0.01, max_iter=100000	0.90	0.91	0.88
	Logistic regression	C=10000, l2 regularization	1.0	1.0	1.0
	Linear SVC	-	1.0	1.0	1.0
Decision tree		depth=1	1.0	1.0	1.0
Decision tree ensemble	Random forest	n_estimator=3	1.0	1.0	1.0
	Gradient boosting	max_depth=1, learning_rate=0.001	1.0	1.0	1.0
Kernel trick	SVM	C=1	1.0	1.0	1.0
Deep learning	MLP	alpha=0.00001, max_iter=10	1.0	1.0	1.0

TABLE 4. Cross-validation result with optimal parameters.

Machine learning model		Validation model	Cross-validation score
KNN		k-fold cross-validation	0.997
Linear model	Linear regression	k-fold cross-validation	0.907
	Ridge regression	k-fold cross-validation	0.906
	Lasso regression	k-fold cross-validation	0.900
	Logistic regression	LOOCV	0.997
	Linear SVC	Cross-validation	0.997
Decision tree		Cross-validation	1.0
Decision tree ensemble	Random forest	Shuffle-split cross-validation	0.996
	Gradient boosting	Cross-validation	0.997
Kernel trick	SVM	Cross-validation	0.997
Deep learning	MLP	LOOCV	0.991

TABLE 5. Performance evaluation results (accuracy, precision, recall, F1-score).

Machine learning model		Accuracy	Precision	Recall	F1-score
KNN		1.0	1.0	1.0	1.0
Linear model	Linear regression	-	-	-	-
	Ridge regression	-	-	-	-
	Lasso regression	-	-	-	-
	Logistic regression	1.0	1.0	1.0	1.0
	Linear SVC	1.0	1.0	1.0	1.0
Decision tree		1.0	1.0	1.0	1.0
Decision tree ensemble	Random forest	1.0	1.0	1.0	1.0
	Gradient boosting	0.997	1.0	0.994	1.0
Kernel trick	SVM	1.0	1.0	1.0	1.0
Deep learning	MLP	1.0	1.0	1.0	1.0

training set score by 100.2% from 0.998 to 1.0 while the test set score improved by 100.3% from 0.997 to 1.0. By applying the optimal parameters, all of the models aside from linear regression, Ridge regression, and Lasso regression were able to adequately learn the data.

B. MODEL VALIDATION RESULT

In order to derive the model validation results, an optimal validation model was derived based on the cross validation methods described in section 2. The results are shown in Table 4.

The experiment results showed that only the decision tree has a perfect cross validation score. KNN, logistic regression, linear SVC, random forest, gradient boosting tree, SVM, and MLP have high level cross-validation scores, while linear regression, Ridge regression, and Lasso regression have relatively high cross-validation scores. Nevertheless, all cross-validation scores are 0.9 or higher, and the results suggest that the machine learning models can be used to appropriately detect files infected by ransomware.

In the validation model, cross validation and k-fold cross validation are used for verification in many machine learning models, while LOOCV and shuffle split cross validation are used for validation in several models.

C. PERFORMANCE EVALUATION RESULT

In order to obtain the performance of the machine learning models used in the proposed system, the accuracy, precision, recall, and F1 score of each model were evaluated. As shown in Equation (10), the accuracy is the number of correctly classified numbers (TP and TN, True Positive and True Negative, respectively) divided by the total number of samples. Precision measures how many samples are positive (TP) in the positive classified (TP+FP), as shown in Equation (11). Recall measures how many of all positive samples (TP+FN) are classified as positive (TP), as shown in Equation (12). F1-score is a harmonic mean of precision and recall, and these are summarized together, as shown in Equation (13). All performance metrics range from the lowest value at 0 to the highest value at 1. The results of these performance evaluations are shown in Table 5.

The results show that linear regression, Ridge regression, and Lasso regression do not have any meaningful performance evaluation results. The accuracy, precision, recall, and F1-score are all 1.0 in KNN, logistic regression, linear SVC,

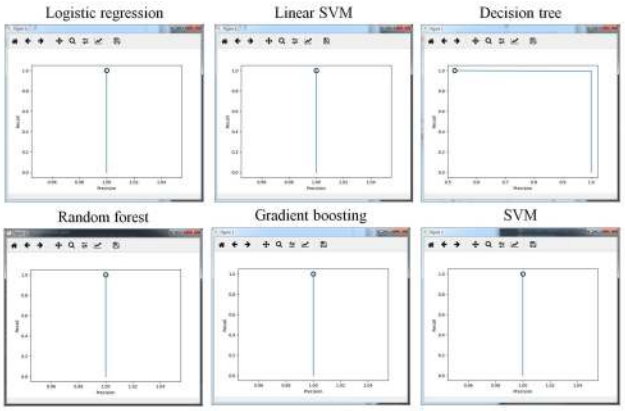

FIGURE 4. Precision-recall curve evaluation result.

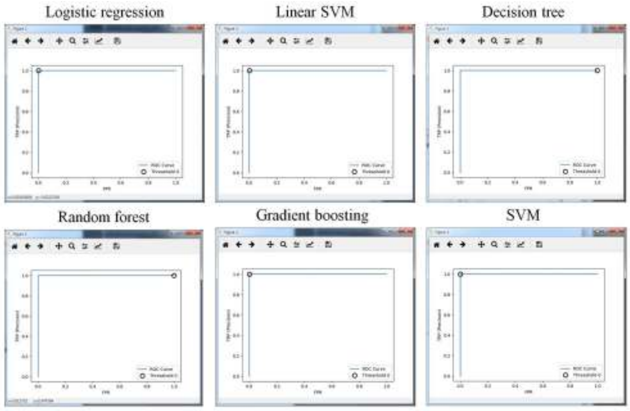

FIGURE 5. ROC curve evaluation result.

TABLE 6. Performance evaluation results (AUC).

Machine learning model		AUC
KNN		1.0
Linear model	Linear regression	1.0
	Ridge regression	1.0
	Lasso regression	1.0
	Logistic regression	1.0
	Linear SVC	1.0
Decision tree		1.0
Decision tree ensemble	Random forest	1.0
	Gradient boosting	1.0
Kernel trick	SVM	1.0
Deep learning	MLP	1.0

decision tree, random forest, SVM, and MLP except for the gradient boosting tree. This indicates that the performances of the machine learning models used in this paper are almost perfect.

$$ACCURACY = \frac{TP + TN}{TP + TN + FP + FN} \tag{10}$$

$$PRECISION = \frac{TP}{TP + FP} \tag{11}$$

$$RECALL = \frac{TP}{TP + FN} \tag{12}$$

$$F1 - score = 2 \times \frac{PRECISION \cdot RECALL}{PRECISION + RECALL} \tag{13}$$

There is a precision-recall curve to visualize all thresholds or to visualize precision and recall individually.

Namely, this depicts a curve graph with an ordered list of precision and recall values for all possible thresholds, and the curve on the graph indicates that better classifiers are closer to the upper right. That is, the upper right point is where both the precision and the recall are high at a certain threshold value. In this paper, the precision-recall curves of logistic regression, linear SVC, decision tree, random forest, gradient boosting tree, and SVM are presented in Fig. 4.

As shown in Fig. 4, the results of all of the models being evaluated are at the upper right of the curve. Therefore, the performances of the machine learning models used in this paper are almost perfect.

A Receiver Operating Characteristics (ROC) curve is a widely used tool to analyze the characteristics of the classifier at various thresholds. This curve, similar to the precision-recall curve, considers all of the threshold values of the classifier and represents the false positive rate (FPR) against the true positive rate (TPR). The TPR is the recall rate while the FPR is the rate of misclassification as false positive. The formula of FPR is shown in Equation (14).

$$FPR = \frac{FP}{FP + TN} \tag{14}$$

The ROC curve indicates good performance when the curve is close to the upper left. In this paper, the ROC curves of logistic regression, linearSVC, decision tree, random forest, gradient boosting tree, and SVM are shown in Fig. 5.

As shown in Fig. 5, the results of all of the models being evaluated are at the upper left of the curve. Therefore, the performances of the machine learning models used in this paper are almost perfect.

The Area under the curve (AUC) is the last evaluation indicator. This summarizes the ROC curve as the area value under the ROC curve, and the AUC result always has a value between the worst at 0 and the best at 1. Thus, this means evaluating the ranking of positive samples; namely, the probability that randomly selected positive class point scores will be higher than the randomly selected negative class point scores. If the AUC value is 1, the score of all positive points will be higher than the score of all negative points. The AUC evaluation results of the machine learning models used in this paper are shown in Table 6.

As shown in Table 6, all of the machine learning models used have an AUC score of 1.0. This means that the performances of the machine learning models are almost perfect.

Finally, the detection rates between the proposed method and the existing methods are compared in Table 7. The existing methods include the file-based detection method to detect ransomware based on file information or signature, the resource-based detection method using resources usages such as I/O requests and CPU, the system-based detection

TABLE 7. Comparison of the proposed methods with existing detection methods.

Detection methods		DETECTION RATE
File-based detection	[38]	0.99
	[40]	0.91
	[47]	0.987
Resource-based detection	[44]	0.963
	[45]	0.977
System-based detection	[48]	0.969
Machine learning-based detection [39]	SVM	0958
	Random Forest	0.958
	Native Bayes	0.935
	Simple Logistic	0.982
Deep learning	[43]	0.95
	[46]	0.939
Others	[41]	0.966
	[42]	0.978
Proposed method	KNN	1.0
	Logistic regression	1.0
	Linear SVC	1.0
	Decision tree	1.0
	Random forest	1.0
	Gradient boosting	0.997
	SVM	1.0
	MLP	1.0

method that detects ransomware based on system information such as user activities and messages, the machine learning-based detection method using API call information, the deep learning-based detection method, and other detection methods. As shown in Table 7, various detection rates were obtained, ranging from a minimum of 91% to a maximum of 99%. Nevertheless, the proposed method has a higher detection rate than the existing methods in all machine learning models such as logistic regression, linear SVC, decision tree, random forest, gradient boosting tree, SVM, and MLP. Therefore, the proposed method more effectively detects files infected by ransomware than the existing ransomware detection methods, and also protects the user's original files by detecting infected files synchronized to the backup system.

V. CONCLUSION

In this paper, we proposed a method to detect files infected with ransomware using machine learning models measuring the entropy of files for the backup system. Parts of the image files and compressed files have high entropy, and detecting ransomware-infected files with only the entropy reference value is limited. In order to overcome this limitation, the proposed method detects infected files using machine learning based on entropy according to various file formats. The experiment results showed that the entropy-based method classified the infected files at a very high rate in most machine learning models and also had very low rates of false positive and false negative. Finally, the performance metrics such as accuracy, precision, recall, F1-score, precision-recall curve, ROC curve, and AUC were all evaluated highly. Therefore, the proposed method effectively detects files infected by ransomware. In this way, files infected by ransomware are not synchronized to the backup system, and the original files can be recovered by restoring the files stored in the backup system even if the user system is infected by ransomware. In the future, we will obtain results for various file formats, and study a method with which to artificially detect ransomware by deriving the optimized values and parameters for each user based on the backup files of each user.

CONFLICTS OF INTEREST

The authors declare no conflict of interest.

REFERENCES

[1] A. Gazet, "Comparative analysis of various ransomware virii," *J. Comput. Virol.*, vol. 6, no. 1, pp. 77–90, Feb. 2008.

[2] Wikipedia. *Ransomware*. Accessed: Aug. 9, 2019. [Online]. Available: https://en.wikipedia.org/wiki/Ransomware

[3] C. Everett, "Ransomware: To pay or not to pay?" *Comput. Fraud Secur.*, vol. 2016, no. 4, pp. 8–12, Apr. 2016.

[4] B. N. Giri, N. Jyoti, and M. Avert, *The Emergence of Ransomware*. Auckland, New Zealand: AVAR, 2006.

[5] S. Song, B. Kim, and S. Lee, "The effective ransomware prevention technique using process monitoring on Android platform," *J. Mobile Inf. Syst.*, vol. 2016, Mar. 2016, Art. no. 2946735.

[6] L. Xin and Q. Liao, "Ransomware: A new cyber hijacking threat to enterprises," in *Handbook of Research on Information Security and Assurance*. Hershey, PA, USA: IGI, 2009.

[7] P. B. Pathak, "A dangerous trend of cybercrime: Ransomware growing challenge," *Int. J. Adv. Res. Comput. Eng. Technol.*, vol. 5, no. 2, pp. 371–373, Feb. 2016.

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

[9] Cisco Blogs. (Apr. 2015). *Threat Spotlight: TeslaCrypt–Decrypt It Yourself*. [Online]. Available: http://blogs.cisco.com/security/talos/teslacrypt

[10] M. M. Ahmadian, H. R. Shahriari, and S. M. Ghaffarian, "Connection-monitor & connection-breaker: A novel approach for prevention and detection of high survivable ransomwares," in *Proc. 12th Int. Iranian Soc. Cryptol. Conf. Inf. Secur. Cryptol.*, Sep. 2015, pp. 79–84.

[11] *KeRanger*. Accessed: Aug. 9, 2019. [Online]. Available: https://en.wikipedia.org/wiki/KeRanger

[12] D. Kim and S. Kim, "Design of quantification model for ransom ware prevent," *World J. Eng. Technol.*, vol. 3, pp. 203–207, Oct. 2015.

[13] K. Lee, K. Yim, and J. T. Seo, "Ransomware prevention technique using key backup," *Concurrency Comput., Pract. Exper.*, vol. 30, no. 3, p. e4337, Oct. 2017.

[14] V. Blue, *CryptoLocker's Crimewave: A Trail of Millions in Laundered Bitcoin*. ZDNet, Dec. 2013. Accessed: Aug. 9, 2019. [Online]. Available: https://www.zdnet.com/article/cryptolockers-crimewave-a-trail-of-millions-in-laundered-bitcoin/

[15] A. Kharraz, S. Arshad, C. Mulliner, W. Robertson, and E. Kirda, "A large-scale, automated approach to detecting ransomware," in *Proc. 25th USENIX Secur. Symp.*, Aug. 2016, pp. 757–772.

[16] Z. Li, C. Xiang, and C. Wang, "Oblivious transfer via lossy encryption from lattice-based cryptography," *J. Wireless Commun. Mobile Comput.*, vol. 2018, Sep. 2018, Art. no. 5973285.

[17] C. Boura and A. Canteaut, "On the boomerang uniformity of cryptographic sboxes," *IACR Trans. Symmetric Cryptol.*, vol. 2018, no. 3, pp. 290–310, Sep. 2018.

[18] M. Cheraghchi, "Expressions for the entropy of basic discrete distributions," *IEEE Trans. Inf. Theory*, vol. 65, no. 7, pp. 3999–4009, Jul. 2019.

[19] C. Shen, H. Li, G. Sahin, H.-A. Choi, and Y. Shah, "Golay code based bit mismatch mitigation for wireless channel impulse response based secrecy generation," *IEEE Access*, vol. 7, pp. 2999–3007, 2018.

[20] M. Sahrai, B. Arzhang, D. Taherkhani, and V. T. A. Boroojerdi, "Control of the entanglement between triple quantum dot molecule and its spontaneous emission fields via quantum entropy," *Phys. E, Low-Dimensional Syst. Nanostruct.*, vol. 67, pp. 121–127, Mar. 2015.

[21] NIST. *Recommendation for the Entropy Sources Used for Random Bit Generation*. Accessed: Aug. 9, 2019. [Online]. Available: https://csrc.nist.gov/csrc/media/publications/sp/800-90b/draft/documents/sp800-90b_second_draft.pdf

[22] X. Guo, R. Liu, P. Li, C. Cheng, M. Wu, and Y. Guo, "Enhancing extractable quantum entropy in vacuum-based quantum random number generator," *Entropy*, vol. 20, no. 11, p. 819, Oct. 2018.

[23] H. Martin, P. Martin-Holgado, P. Peris-Lopez, Y. Morilla, and L. Entrena, "On the entropy of oscillator-based true random number generators under ionizing radiation," *Entropy*, vol. 20, no. 7, p. 513, Jul. 2018.

[24] M.-L. Zhang and Z.-H. Zhou, "ML-KNN: A lazy learning approach to multi-label learning," *Pattern Recognit.*, vol. 40, no. 7, pp. 2038–2048, Jul. 2007.

[25] D. Liu, X. Lin, and D. Ghost, "Semiparametric regression of multidimensional genetic pathway data: Least-squares kernel machines and linear mixed models," *Biometrics*, vol. 63, no. 4, pp. 1079–1088, Dec. 2007.

[26] S. An, W. Liu, and S. Venkatesh, "Fast cross-validation algorithms for least squares support vector machine and kernel ridge regression," *Pattern Recognit.*, vol. 40, no. 8, pp. 2154–2162, Aug. 2007.

[27] V. Roth, "The generalized LASSO," *IEEE Trans. Neural Netw.*, vol. 15, no. 1, pp. 16–28, Jan. 2004.

[28] W. Cheng and E. Hüllermeier, "Combining instance-based learning and logistic regression for multilabel classification," *Mach. Learn.*, vol. 76, nos. 2–3, pp. 211–225, 2009.

[29] B. Schlkopf and A. Smola, *Learning With Kernels*. Cambridge, MA, USA: MIT Press, 2002.

[30] C. Sinclair, L. Pierce, and S. Matzner, "An application of machine learning to network intrusion detection," in *Proc. 15th Annu. Comput. Secur. Appl. Conf. (ACSAC)*, Dec. 1999, pp. 371–377.

[31] R. E. Banfield, L. O. Hall, K. W. Bowyer, and W. P. Kegelmeyer, "A comparison of decision tree ensemble creation techniques," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 1, pp. 173–180, Jan. 2007.

[32] W. Lin, Z. Wu, L. Lin, A. Wen, and J. Li, "An ensemble random forest algorithm for insurance big data analysis," *IEEE Access*, vol. 5, pp. 16568–16575, 2017.

[33] Y. Zhang and A. Haghani, "A gradient boosting method to improve travel time prediction," *Transp. Res. C, Emerg. Technol.*, vol. 58, pp. 308–324, Sep. 2015.

[34] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017.

[35] Y.-D. Zhang, Z.-J. Yang, H.-M. Lu, X.-X. Zhou, P. Phillips, Q.-M. Liu, and S.-H. Wang, "Facial emotion recognition based on biorthogonal wavelet entropy, fuzzy support vector machine, and stratified cross validation," *IEEE Access*, vol. 4, pp. 8375–8385, 2016.

[36] M. Kearns and D. Ron, "Algorithmic stability and sanity-check bounds for leave-one-out cross-validation," *Neural Comput.*, vol. 11, no. 6, pp. 1427–1453, Aug. 1999.

[37] F. Mercaldo, V. Nardone, A. Santone, and C. A. Visaggio, "Ransomware steals your phone. Formal methods rescue it," in *Proc. Int. Conf. Formal Techn. Distrib. Objects, Compon., Syst.*, Jun. 2016, pp. 212–221.

[38] Z.-G. Chen, H.-S. Kang, S.-N. Yin, and S.-R. Kim, "Automatic ransomware detection and analysis based on dynamic API calls flow graph," in *Proc. Int. Conf. Res. Adapt. Convergent Syst.*, Sep. 2017, pp. 196–201.

[39] J. Lee, K. Jeong, and H. Lee, "Detecting metamorphic malwares using code graphs," in *Proc. ACM Symp. Appl. Comput.*, Mar. 2010, pp. 1970–1997.

[40] F. Karbalaie, A. Sami, and M. Ahmadi, "Semantic malware detection by deploying graph mining," *Int. J. Comput. Sci. Issues*, vol. 9, no. 1, pp. 1–7, Jan. 2012.

[41] N. Nissim, R. Moskovitch, L. Rokach, and Y. Elovici, "Novel active learning methods for enhanced PC malware detection in windows OS," *Expert Syst. Appl.*, vol. 41, no. 13, pp. 5843–5857, 2014.

[42] J. Saxe and K. Berlin, "Deep neural network based malware detection using two dimensional binary program features," in *Proc. 10th Int. Conf. Malicious Unwanted Softw. (MALWARE)*, Oct. 2015, pp. 11–20.

[43] A. Kharaz, S. Arshad, C. Mulliner, W. Robertson, and E. Kirda, "UNVEIL: A large-scale, automated approach to detecting ransomware," in *Proc. 25th USENIX Secur. Symp.*, 2016, pp. 757–772.

[44] A. Continella, A. Guagnelli, G. Zingaro, G. De Pasquale, A. Barenghi, S. Zanero, and F. Maggi, "ShieldFS: A self-healing, ransomware-aware filesystem," in *Proc. 32nd Annu. Conf. Comput. Secur. Appl.*, Dec. 2016, pp. 336–347.

[45] A. Tseng, Y. Chen, Y. Kao, and T. Lin, "Deep learning for ransomware detection," *IEICE Tech. Rep.*, vol. 116, no. 282, pp. 87–92, 2016.

[46] P. Wang and Y.-S. Wang, "Malware behavioural detection and vaccine development by using a support vector model classifier," *J. Comput. Syst. Sci.*, vol. 81, no. 6, pp. 1012–1026, 2015.

[47] A. Saracino, D. Sgandurra, G. Dini, and F. Martinelli, "Madam: Effective and efficient behavior-based Android malware detection and prevention," *IEEE Trans. Dependable Secure Comput.*, vol. 15, no. 1, pp. 83–97, Jan. 2018.

KYUNGROUL LEE received the B.S., M.S., and Ph.D. degrees from Soonchunhyang University, South Korea, in 2008, 2010, and 2015, respectively.

He has been with the R&BD Center for Security and Safety Industries (SSI), Soonchunhyang University, as a Research Professor, since 2016, where he is currently with the Laboratory of Information Systems Security Assurance.

His research interests include security protocols, vulnerability analysis, hardware security, reverse engineering, platform security, and offensive security analysis. Related to these topics, he has worked on more than 20 research projects and published more than a 100 research papers. He has served as a Committee Chair for the international conferences and workshops and the editorial member for the IJITST.

SUN-YOUNG LEE received the B.S. and M.S. degrees from the Department of Computer Science, Pukyong National University, Busan, South Korea, in 1993 and 1995, respectively, and the Ph.D. degree from the Department of Communication and Information Engineering, The University of Tokyo, Tokyo, Japan, in 2001.

She is currently a Professor with the Department of Information Security Engineering, Soonchunhyang University. She has served as an Executive Board Member for the Korea Institute of Information Security and Cryptology. Her research interests include cryptography, information theory, digital right management, and healthcare information security.

KANGBIN YIM received the B.S., M.S., and Ph.D. degrees from the Department of Electronics Engineering, Ajou University, Suwon, South Korea, in 1992, 1994, and 2001, respectively.

He is currently a Professor with the Department of Information Security Engineering, Soonchunhyang University. His research interests include vulnerability assessment, code obfuscation, malware analysis, leakage prevention, secure platform architecture, and mobile security. Related to these topics, he has worked on more than 60 research projects and published more than a 100 research papers.

Prof. Yim has served as an Executive Board Member for the Korea Institute of Information Security and Cryptology, the Korean Society for Internet Information, and The Institute of Electronics Engineers of Korea. He has also served as a Committee Chair for the international conferences and workshops and the guest editor for the journals such as JIT, MIS, JCPS, JISIS, and JoWUA.