Deep Abstraction and Weighted Feature Selection for Wi-Fi Impersonation Detection

Muhamad Erza Aminanto, Rakyong Choi, Harry Chandra Tanuwidjaja,
Paul D. Yoo, *Senior Member, IEEE* and Kwangjo Kim, *Member, IEEE*

***Abstract*—The recent advances in mobile technologies have resulted in IoT-enabled devices becoming more pervasive and integrated into our daily lives. The security challenges that need to be overcome mainly stem from the open nature of a wireless medium such as a Wi-Fi network. An impersonation attack is an attack in which an adversary is disguised as a legitimate party in a system or communications protocol. The connected devices are pervasive, generating high-dimensional data on a large scale, which complicates simultaneous detections. Feature learning, however, can circumvent the potential problems that could be caused by the large-volume nature of network data. This study thus proposes a novel Deep-Feature Extraction and Selection (D-FES), which combines stacked feature extraction and weighted feature selection. The stacked autoencoding is capable of providing representations that are more meaningful by reconstructing the relevant information from its raw inputs. We then combine this with modified weighted feature selection inspired by an existing shallow-structured machine learner. We finally demonstrate the ability of the condensed set of features to reduce the bias of a machine learner model as well as the computational complexity. Our experimental results on a well-referenced Wi-Fi network benchmark dataset, namely, the Aegean Wi-Fi Intrusion Dataset (AWID), prove the usefulness and the utility of the proposed D-FES by achieving a detection accuracy of 99.918% and a false alarm rate of 0.012%, which is the most accurate detection of impersonation attacks reported in the literature.**

***Index Terms*—Intrusion detection system, impersonation attack, deep learning, feature extraction, stacked autoencoder, large-scale Wi-Fi networks.**

Manuscript received March 16, 2017; revised July 17, 2017 and September xx, 2017; accepted xxxxx xxxxxx; date of publication xxx xxxx; date of current version xxxx xxxxxx. The associate editor coordinating the review process of this manuscript and approving it for publication was Prof. Tansu Alpcan.

M. E. Aminanto, R. Choi, H. C. Tanuwidjaja and K. Kim are with the School of Computing (SoC), Korea Advanced Institute of Science and Technology (KAIST), Daejeon 34141, Republic of Korea. (e-mail: aminanto@kaist.ac.kr; thepride@kaist.ac.kr; elevantista@kaist.ac.kr; kkj@kaist.ac.kr)

P. D. Yoo is with Centre for Electronic Warfare, Information and Cyber (EWIC), Cranfield Defence and Security, Defence Academy of the United Kingdom, Shrivenham, SN6 8LA, United Kingdom. (email: p.yoo@cranfield.ac.uk)

The work of M. E. Aminanto, R. Choi and K. Kim was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT) (2013-0-00396, Research on Communication Technology using Bio-Inspired Algorithm and 2017-0-00555, Towards Provable-secure Multi-party Authenticated Key Exchange Protocol based on Lattices in a Quantum World) and by National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2015R1A-2A2A01006812)

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.xxxx/TIFS.2017.xxxxxxxx

I. Introduction

The rapid growth of the Internet has led to a significant increase in wireless network traffic in recent years. According to a worldwide telecommunication consortium [1], proliferation of 5G and Wi-Fi networks is expected to occur in the next decades. By 2020 [1] wireless network traffic is anticipated to account for two thirds of total Internet traffic — with 66% of IP traffic expected to be generated by Wi-Fi and cellular devices only. Although wireless networks such as IEEE 802.11 have been widely deployed to provide users with mobility and flexibility in the form of high-speed local area connectivity, other issues such as privacy and security have raised. The rapid spread of Internet of Things (IoT)-enabled devices has resulted in wireless networks becoming to both passive and active attacks, the number of which has grown dramatically [2]. Examples of these attacks are impersonation, flooding, and injection attacks.

An Intrusion Detection System (IDS) is one of the most common components of every network security infrastructure [3] including wireless networks [4]. Machine-learning techniques have been well adopted as the main detection algorithm in IDS owing to their model-free properties and learnability [5]. Leveraging the recent development of machine-learning techniques such as deep learning [6] can be expected to bring significant benefits in terms of improving existing IDSs particularly for detecting impersonation attacks in large-scale networks. Based on the detection method, IDS can be classified into three types; misuse, anomaly, and specification-based IDS. A misuse-based IDS also known as a signature-based IDS [7] detects any attack by checking whether the attack characteristics match previously stored signatures or patterns of attacks. This type of IDS is suitable for detecting known attacks; however, new or unknown attacks are difficult to detect.

An anomaly-based IDS identifies malicious activities by profiling normal behavior and then measuring any deviation from it. The advantage of this detection type is its ability for detecting unknown attacks. However, misuse-based IDS achieved higher performance for detecting known attacks than anomaly-based IDS. An IDS that leverages machine-learning method is an example of an anomaly-based IDS [8], whereas

[1] Cisco Visual Networking Index: Forecast and Methodology 2015-2020, published at www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html

a specification-based IDS manually defines a set of rules and constraints to express the normal operations. Any violation of the rules and constraints during execution is flagged as an attack as discussed by Mitchell *et al.* [9]. In this study, we consider anomaly-based IDS rather than misuse-based IDS to enable us to include new attacks. We adopt anomaly-based IDS instead of specification-based IDS in order to avoid manual intervention by the network administrator when defining the set of new rules and constraints. From the perspective of human error, relying on human intelligence is undesirable [10], ($e.g.$, experts may misunderstand the alarm while a network is being monitored).

The wide and rapid spread of computing devices using Wi-Fi networks creates complex, large, and high-dimensional data, which cause confusion when capturing attack properties. Feature learning acts as an important tool for improving the learning process of a machine-learning model. It consists of feature construction, extraction, and selection. Feature construction expands the original features to enhance their expressiveness, whereas feature extraction transforms the original features into a new form and feature selection eliminates unnecessary features [11]. Feature learning is a key to improve the performance of existing machine-learning-based IDSs.

One of the key contributions of this study is the introduction of novel Deep-Feature Extraction and Selection (D-FES), which improves its feature learning process by combining stacked feature extraction with weighted feature selection. The feature extraction of Stacked Auto Encoder (SAE) is capable of transforming the original features into a more meaningful representation by reconstructing its input and providing a way to check that the relevant information in the data has been captured. SAE can be efficiently used for unsupervised learning on a complex dataset. Unlike supervised learning, unsupervised learning does not require a labeled dataset for training. Unsupervised learning capability is of critical importance as it allows a model to be built to detect new attacks without creating costly labels or dependent variables. SAE is built by stacking additional unsupervised feature learning layers and can be trained by using greedy methods for each additional layer. We train a new hidden layer by training a standard supervised neural network with one hidden layer. The output of the previously trained layer is taken as pre-training initialization for the next layer and as the extracted features.

We then propose a modified feature-selection-based method by considering the weights of each feature obtained from lightweight machine-learning models. Supervised machine-learning algorithms are capable of extracting the relevant necessary features from classification results. Support Vector Machine (SVM) was suggested to be embedded as a feature selection since it ranks the input features based on weights [12]. Weights computed by Artificial Neural Network (ANN) classification are able to determine the reliability of detecting a certain attack [13]. C4.5 also claimed to be inherently includes feature selection functions [14]. In those algorithms, the weights of nodes represent the strength of connections among nodes. The weight values from a trained model indicate the importance of the corresponding inputs. We select the most suitable features according to the weights provided SVM, ANN and C4.5. The small set of selected features obtained from the combination of the extracted features and the large number of original features is not only essential for real-time processing but also suitable for the large-scale nature of Wi-Fi networks. The final step of the proposed approach entails utilizing an ANN as a classifier to build an IDS using significantly condensed and extracted features only.

We evaluate the proposed approach on the well-referenced AWID dataset, a benchmark dataset built by Kolias *et al.* [15] for Wi-Fi networks. They tested a number of existing machine-learning models on the dataset in a heuristic manner. The lowest detection rate with an accuracy of 22% was observed specifically for an impersonation attack. The proposed approach outperforms their model in that particular category to achieve an accuracy of 99.91%. Clearly, the novelty of combining a deep-learning feature extractor and weighted feature selection with an ANN classifier improves the ability to detect impersonation attacks. Furthermore, the proposed approach can be further generalized for different attack types, both known and unknown large-scale attacks on Wi-Fi networks. In summary, the main contributions of this study are three-fold:

- Design of an impersonation attack detector using a condensed set of features without modifying any protocol and without using any additional measurement.
- Abstraction of raw features using a deep learning technique. The extracted features are measured once more using weighted feature selection techniques such that a measure-in-depth technique is proposed.
- Design of the proposed D-FES, which can be implemented in a real wireless network setup because an unbalanced dataset is used for testing purposes.

The remainder of the paper is organized as follows: Section II introduces previous work in which feature learning was adopted. A review of studies relating to an impersonation attack is provided in Section III. Section IV describes the proposed methodology. Section V discusses the experimental results. Section VI compares the experimental results of the proposed methodology with those of other well-referenced models. Section VII concludes the paper with an overview of future work.

II. Related Work

An IDS has been studied for decades especially on anomaly-based IDSs. Fragkiadakis *et al.* [4] proposed an anomaly-based IDS using *Dempster-Shafer's rule* for measurement. Shah *et al.* [16] also developed an evidence theory for combining anomaly-based and misuse-based IDSs. Bostani *et al.* [17] proposed an anomaly-based IDS by modifying an optimum path forest combined with k-means clustering. A distributed anomaly-based IDS, called $TermID$, proposed by Kolias *et al.* [18] incorporating ant colony optimization and rule induction in a reduced data operation to achieve data parallelism and reduce privacy risks.

Feature selection techniques are useful for reducing model complexity, which leads to faster learning and real-time processes. Kayacik *et al.* [19] investigated the relevance of each feature in the KDD'99 Dataset with a list of the most relevant features for each class label and provided useful discussions on the roles of information gain theories. Their work confirmed the importance and the role of feature selection for building an accurate IDS model. Puthran *et al.* [20] also worked on relevant features in the KDD'99 Dataset and improved the decision tree by using binary and quad splits. Almusallam *et al.* [21] leveraged a filter-based feature selection method. Zaman and Karray [22] categorized IDSs based on the Transmission Control Protocol/Internet Protocol (TCP/IP) network model using a feature selection method known as the Enhanced Support Vector Decision Function (ESVDF). Louvieris *et al.* [23] proposed an effect-based feature identification IDS using naïve Bayes as a feature selection method. Zhu *et al.* [24] also proposed a feature selection method using a multi-objective approach.

On the other hand, Manekar and Waghmare [25] leveraged Particle Swarm Optimization (PSO) and SVM. PSO performs feature optimization to obtain an optimized feature, after which SVM performs the classification task. A similar approach was introduced by Saxena and Richariya [26], although the concept of weighted feature selection was introduced by Schaffernicht and Gross [27]. Exploiting SVM-based algorithms as a feature selection method was introduced by Guyon *et al.* [28]. This method leveraged the weights adjusted during support vector learning and resulted in ranking the importance of input features. Another related approach was proposed by Wang [29] who ranked input features based on weights learned by an ANN. This method showed the ability of deep neural networks to find useful features among the raw data. Aljawarneh *et al.* [30] proposed a hybrid model of feature selection and an ensemble of classifiers which are tend to be computationally demanding.

We have examined several feature selection methods for IDS. Huseynov *et al.* [31] inspected ant colony clustering method to find feature clusters of botnet traffic. The selected features in [31] are independent from traffic payload and represent the communication patterns of botnet traffic. However, this botnet detection does not scale for enormous and noisy dataset due to the absence of control mechanism for clustering threshold. Kim *et al.* [32] tested artificial immune system and swarm intelligence-based clustering to detect unknown attacks. Furthermore, Aminanto *et al.* [33] discussed the utility of ant clustering algorithm and fuzzy inference system for IDS. We can claim that their bio-inspired clustering methods need to be scrutinized further.

Not only feature selection, but also feature extraction has been proposed to improve classification performance. Shin *et al.* [34] leveraged SAE for unsupervised feature learning in the field of medical imaging. This method showed that SAE, which is a type of deep learning techniques, can be effectively used for unsupervised feature learning on a complex dataset. Unsupervised learning by SAE can be used to learn hierarchical features that are useful for limited instances on a dataset.

Owing to the scale and complexity of recent data, building a machine-learning-based IDS has become a daunting task. As we aim to detect impersonation attacks in large-scale Wi-Fi networks, a large AWID dataset was chosen for this study. Kolias *et al.* [15] created a comprehensive 802.11 networks dataset that is publicly available and evaluated various machine-learning algorithms to validate their dataset in a heuristic manner. The performance of the IDS was unsatisfactory, indicating that conventional machine-learning methods are incapable of detecting attacks on large-scale Wi-Fi networks. Moreover, among all the classification results obtained, an impersonation attack detection was the most unsatisfactory. One of the main goals of our study thus is to improve the detection of impersonation attacks by leveraging the advancement of recent machine-learning techniques. Recently, Usha and Kavitha [35] proposed a combination of modified normalized gain and PSO to select optimal features and successfully improved the attack detection rate tested on the AWID dataset. However, they focus on detecting all attack classes rather than impersonation attacks only, which is the problem raised by Kolias *et al.* in [15]. We leveraged SAE for classification purposes and then improved our impersonation detector using weighted feature learning from shallow machine learners [36][37]. Thus, this study extends our previous work [37] to a novel IDS, which combines deep learning abstraction and a weighted feature selection technique.

III. IMPERSONATION ATTACK

Impersonation attacks are one of the most common Wi-Fi network attacks, where an adversary impersonates a legitimate object on the network, including users or base stations, to obtain unauthorized access to a system or a wireless network. Impersonation attacks are performed in different ways such as gaining unauthorized access, cloning device, creating a rogue Access Point (AP), spoofing address, and performing a replay attack [38]. Based on the aim of the attackers, an impersonation attack can be categorized as an impersonation of any device in the network, *i.e.*, the AP, either case (i) to crack the key or case (ii) to act as a man-in-the-middle (MITM) to attract a client to connect to the disguised AP [15].

In case (i), adversaries retrieve the keystream of a network protected by Wired Equivalent Privacy (WEP) and take an attack to the next level. Ahmad and Ramachandran [39] demonstrated that WEP keys can be cracked remotely, without requiring the attacker to be in the coverage area of the target AP. As its name, *Caffe-Latte*, suggests the attack could be completed from a remote location and within a considerably short amount of time. This attack captures a client's probe-requests of previously associated APs, which might be out of the client's range. It then disguises itself as one of those APs in which case the client would connect to the disguised AP. Similarly, a *Hirte* attack also takes advantage of retrieving the WEP key by forging Address Resolution Protocol (ARP) requests. It, however, uses a different approach by leveraging a fragmentation attack. Both *Caffe-Latte* and *Hirte* attacks could be executed using *aircrack-ng*, a Wi-Fi network security assessment tool [40].

In case (ii), adversaries adopt an MITM approach to perform an impersonation attack such as malicious honeypot, *Evil Twin*, and rogue APs [15]. A malicious honeypot attack is achieved

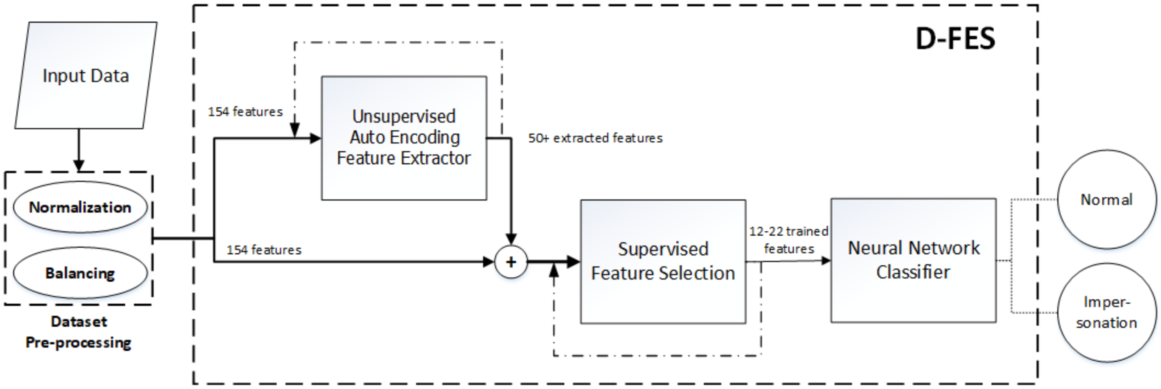


Fig. 1. Stepwise procedure of D-FES with two target classes: normal and impersonation attack

by providing an AP with an attractive name such as *Free Wi-Fi* or *Public Wi-Fi*. Once victims are connected to the malicious honeypot, adversaries can easily gain access to the victim's system. An *Evil Twin* AP uses a Service Set Identifier (SSID) identical to that of a legitimate AP. The victim would thus connect to the *Evil Twin* AP instead of the legitimate one. This attack requires two important assumptions. First, one area may have two identical SSIDs. Second, any legitimate client prefers an AP with higher signal strength. A rogue AP attack using a bait AP, is achieved by the illegitimate installation of an AP on a secure network by an adversary. The rogue AP confuses the legitimate client who sends or receives packets through what they believe to be a legitimate base station or an AP.

An impersonation attack might cause a serious breach of network security as it allows unauthorized or malicious users to access the network [41]. Some researchers proposed new detectors particularly for an impersonation attack [42], [43], [41], [44], [45] and [46]. Malisa *et al.* [42] proposed a mobile application, which can detect both whole and partial user interface impersonations. We agree that the concept of extracting features captures important information to detect an attack. However, our method builds one general method that can be generalized for other attack models. Goga *et al.* [43] released a social network impersonation dataset. They focused on identifying doppelganger attacks by defining a set of features manually. We leverage a deep learning technique to extract and select a set of features automatically without human intervention. The model of Beyah *et al.* [41] is designed to only detect rogue APs. Shang and Gui [44] proposed a novel strategy considering a Differential Flag Byte (DFB) to detect impersonation attacks at the bottom of a protocol stack with low computational complexity. Yilmaz and Arslan [45] developed an impersonation attack detector by measuring power delay profile differences of transmitters located in different places. Lakshmi *et al.* [46] demonstrated a novel way of detecting impersonation attacks by leveraging the properties of each node and operating independently on any cryptographic protocol. This method leveraged a special correlation of Received Signal Strength (RSS) transmitted from wireless nodes and Efficient Probabilistic Packet Marking (EPPM) to detect the impersonation attacks. It then used a cluster-based measurement developed to count the number of attackers and an SVM learner to improve the accuracy of counting the attackers. This method ends with an integrated detection and localization system that would localize the positions of the attackers. However, unlike the above-mentioned detectors, we need a general model that can specifically detect an impersonation attack. A machine-learning-based IDS is believed to be a general countermeasure for monitoring network traffic.

IV. Methodology

Feature learning could be defined as a technique that models the behavior of data from a subset of attributes only. It could also show the correlation between detection performance and traffic model quality effectively [47]. However, feature extraction and feature selection are different. Feature extraction algorithms derive new features from the original features to: (i) reduce the cost of feature measurement, (ii) increase classifier efficiency, and (iii) improve classification accuracy, whereas feature selection algorithms select no more than m features from a total of M input features, where m is smaller than M. Thus, the newly generated features are simply selected from the original features without any transformation. However, their goal is to derive or select a characteristic feature vector with a lower dimensionality which is used for the classification task.

We adopt both feature extraction and selection techniques in D-FES. Fig.1 shows the stepwise procedure of D-FES with two target classes. A pre-processing procedure, which comprises the normalization and balancing steps, is necessary. The procedure is explained in Section V in detail. As illustrated in Algorithm 1, we start D-FES by constructing SAE-based feature extractor with two consecutive hidden layers in order to optimize the learning capabality and the execution time [48]. The SAE outputs 50 extracted features, which are then combined with the 154 original features existing in the AWID dataset [15]. Weighted feature selection methods are then utilized using well-referenced machine learners including SVM, ANN, and C4.5 in order to construct the candidate models, namely D-FES-SVM, D-FES-ANN, and D-FES-C4.5, respectively. SVM separates the classes using a support vector (hyperplane). Then, ANN optimizes the parameters related to hidden layers that minimize the classifying error with

Algorithm 1 Pseudocode of D-FES

```
procedure D-FES
    function DATASET PRE-PROCESSING(Raw Dataset)
        function (Dataset Normalization)Raw Dataset
            return $NormalizedDataset$
        end function
        function (Dataset Balancing)Normalized Dataset
            return $BalancedDataset$
        end function
        return $InputDataset$
    end function
    function DEEP ABSTRACTION($InputDataset$)
        for $i$=1 to $h$ do    ▷ $h$=2; number of hidden layers
            for each data instance do
                Compute $y_i$ (Eq. (1))
                Compute $z_i$ (Eq. (2))
                Minimize $E_i$ (Eq. (6))
                $\theta_i = \{W_i, V_i, b_{f_i}, b_{g_i}\}$
            end for
            $W \leftarrow W_2$    ▷ 2nd layer, 50 extracted features
        end for
        $InputFeatures \leftarrow W + InputDataset$
        return $InputFeatures$
    end function
    function FEATURE SELECTION($InputFeatures$)
        switch D-FES do
            case D-FES-ANN($InputFeatures$)
                return $SelectedFeatures$
            case D-FES-SVM($InputFeatures$)
                return $SelectedFeatures$
            case D-FES-C4.5($InputFeatures$)
                return $SelectedFeatures$
    end function
    procedure CLASSIFICATION($SelectedFeatures$)
        Training ANN
        Minimize $E$ (Eq. (7))
    end procedure
end procedure
```

respect to the training data, whereas C4.5 adopts a hierarchical decision scheme such as a tree to distinguish each feature [49]. The final step of the detection task involves learning an ANN classifier with 12–22 trained features only.

A. *Feature Extraction*

1) (Sparse) Auto Encoder: An Auto Encoder (AE) is a symmetric neural network model, which uses an unsupervised approach to build a model with non-labeled data, as shown in Fig. 2. AE extracts new features by using an encoder-decoder paradigm by running from inputs through the hidden layer only. This paradigm enhances its computational performance and validates that the code has captured the relevant information from the data. The encoder is a function that maps an input x to a hidden representation as expressed by Eq. (1).

$$y = s_f\left(W \cdot x + b_f\right), \tag{1}$$

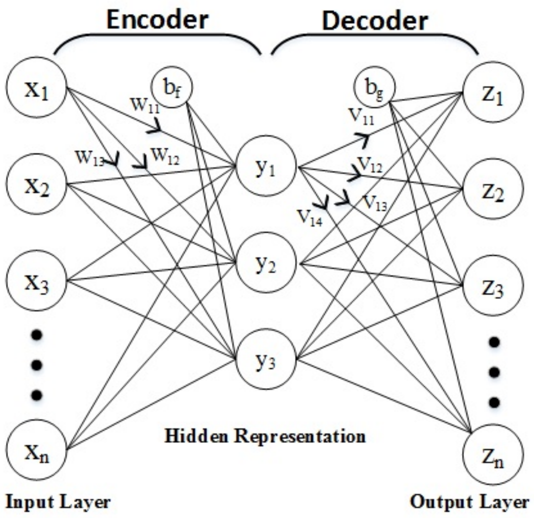


Fig. 2. AE network with symmetric input-output layers and three neurons in one hidden layer

where s_f is a nonlinear activation function which is a decision-making function to determine the necessity of any feature. Mostly, a logistic sigmoid, $sig(t) = \dfrac{1}{1+e^{-t}}$ is used as an activation function because of its continuity and differentiability properties [50]. The decoder function expressed in Eq. (2) maps hidden representation y back to a reconstruction.

$$z = s_g\left(V \cdot y + b_g\right), \tag{2}$$

where s_g is the activation function of the decoder which commonly uses either the identity function, $s_g(t) = t$, or a sigmoid function such as an encoder. We use W and V acts as a weight matrix for the features. b_f and b_g acts as a bias vector for encoding and decoding, respectively. Its training phase finds optimal parameters $\theta = \{W, V, b_f, b_g\}$ which minimize the reconstruction error between the input data and its reconstruction output on a training set.

This study uses a modified form of AE, $i.e.$ sparse AE [51]. This is based on the experiments of Eskin *et al.* [52], in which anomalies usually form small clusters in sparse areas of a feature space. Moreover, dense and large clusters usually contain benign data [21]. For the sparsity of AE, we first observe the average output activation value of a neuron i, as expressed by Eq. (3).

$$\hat{\rho_i} = \frac{1}{N}\sum_{j=1}^{N} s_f\left(w_i^T x_j + b_{f,i}\right), \tag{3}$$

where N is the total number of training data, x_j is the j-th training data, w_i^T is the i-th row of a weight matrix W, and $b_{f,i}$ is the i-th row of a bias vector for encoding b_f. By lowering the value of $\hat{\rho_i}$, a neuron i in the hidden layer shows the specific feature presented in a smaller number of training data.

The task of machine-learning is to fit a model to the given training data. However, the model often fits the particular training data but is incapable of classifying other data and this

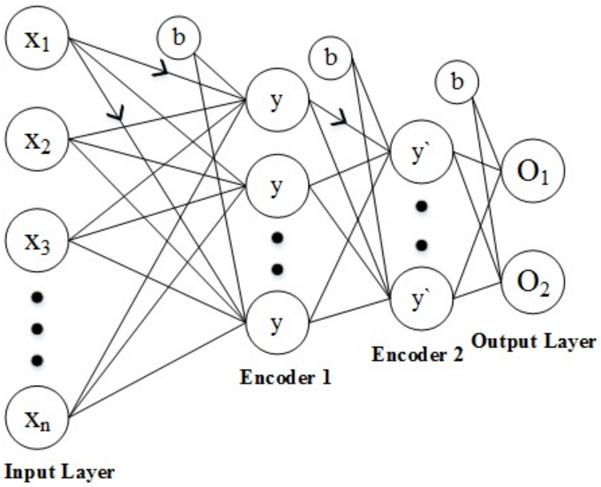


Fig. 3. SAE network with two hidden layers and two target classes

is known as the overfitting problem. In this case, we can use a regularization technique to reduce the overfitting problem. The sparsity regularization $\Omega_{sparsity}$ evaluates how close the average output activation value $\hat{\rho}_i$ and the desired value ρ are, typically with *Kullback-Leibler* (KL) divergence to determine the difference between the two distributions, as expressed in Eq. (4).

$$\begin{aligned}\Omega_{sparsity} &= \sum_{i=1}^{h} KL(\rho \| \hat{\rho}_i) \\ &= \sum_{i=1}^{h} \left[\rho \log\left(\frac{\rho}{\hat{\rho}_i}\right) + (1-\rho)\log\left(\frac{1-\rho}{1-\hat{\rho}_i}\right)\right],\end{aligned} \tag{4}$$

where h is the number of neurons in the hidden layer.

We may increase the value of the entries of the weight matrix W to reduce the value of sparsity regularization. To avoid this situation, we also add regularization for the weight matrix, known as L_2 regularization as stated in Eq. (5).

$$\Omega_{weights} = \frac{1}{2}\sum_{i=1}^{h}\sum_{j=1}^{N}\sum_{k=1}^{K}(w_{ji})^2, \tag{5}$$

where N and K are the number of training data and the number of variables for each data, respectively.

The goal of training sparse AE is to find the optimal parameters, $\theta = \{W, V, b_f, b_g\}$, to minimize the cost function shown in Eq. (6).

$$E = \frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{K}(z_{kn} - x_{kn})^2 + \lambda \cdot \Omega_{weights} + \beta \cdot \Omega_{sparsity}, \tag{6}$$

which is a regulated mean square error with L_2 regularization and sparsity regularization. The coefficient of the L_2 regularization term λ and the coefficient of sparsity regularization term β are specified while training the AE.

2) Stacked (Sparse) AE: (Sparse) AE can be used as deep learning technique by an unsupervised greedy layer-wise pre-training algorithm known as Stacked (Sparse) Auto Encoder (SAE). Here, pre-training refers to the training of a single AE using a single hidden layer. Each AE is trained separately before being cascaded afterwards. This pre-training phase is required to construct a stacked AE. In this algorithm, all layers except the output layer are initialized in a multi-layer neural network. Each layer is then trained in an unsupervised manner as an AE, which constructs new representations of the input.

The performance of the unsupervised greedy layer-wise pre-training algorithm can be significantly more accurate than the supervised one. This is because the greedy supervised procedure may behave too greedy as it extracts less information and considers one layer only [53] [54]. A neural network containing only one hidden layer, it may discard some of the information about the input data as more information could be exploited by composing additional hidden layers.

Fig. 3 shows the SAE network with two hidden layers and two target classes. The final layer implements the softmax function for the classification in the deep neural network. Softmax function is a generalized term of the logistic function that suppresses the K-dimensional vector $\mathbf{v} \in \mathbb{R}^K$ into K-dimensional vector $\mathbf{v}^* \in (0,1)^K$, which adds up to 1. In this function, it is defined T and C as the number of training instances and the number of classes, respectively. The softmax layer minimizes the loss function, which is either the cross-entropy function as in Eq. (7),

$$E = \frac{1}{T}\sum_{j=1}^{T}\sum_{i=1}^{C}\left[z_{ij}\log y_{ij} + (1-z_{ij})\log(1-y_{ij})\right], \tag{7}$$

or the mean-squared error. However, the cross-entropy function is used in this study.

The features from the pre-training phase, which is greedy layer wise, can be used either as an input to a standard supervised machine-learning algorithm or as an initialization for a deep supervised neural network.

B. Feature Selection

The supervised feature selection block in Fig.1 consists of three different feature selection techniques. These techniques are similar in that they consider their resulting weights to select the subset of important features. The following subsections contain further details of each feature selection technique.

1) D-FES-ANN: ANN is used as a weighted feature selection method. The ANN is trained with two target classes only (normal and impersonation attack classes). Fig. 4 shows an ANN network with one hidden layer only where b_1 and b_2 represent the bias values for the corresponding hidden and output layer, respectively.

In order to select the important features, we consider the weight values between the first two layers. The weight represents the contribution from the input features to the first hidden layer. A w_{ij} value close to zero means that the corresponding input feature x_j is meaningless for further propagation, thus having one hidden layer is sufficient for this particular task. We

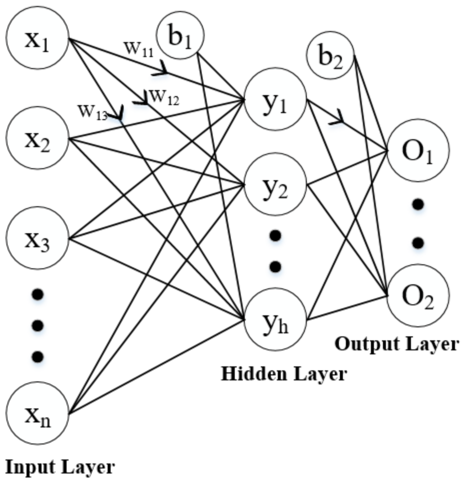


Fig. 4. ANN network with one hidden layer only

Algorithm 2 D-FES-ANN Function

```
function D-FES-ANN($InputFeatures$)
    Training ANN
    $W_{ij}$
    for each input feature do
        Compute $V_j$ (Eq. (8))
    end for
    Sort descending
    $SelectedFeatures \leftarrow V_j > threshold$
    return $SelectedFeatures$
end function
```

define the important value of each input feature as expressed by Eq. (8).

$$V_j = \sum_{i=1}^{h} |w_{ij}|, \tag{8}$$

where h is the number of neurons in the first hidden layer. As described in Algorithm 2, the feature selection process involves selecting the features of which the V_j values are greater than the threshold value after the input features are sorted according to their V_j values in a descending order.

Following the weighted feature selection, ANN is also used as a classifier. When learning with ANN, a minimum global error function is executed. It has two learning approaches, supervised and unsupervised. This study uses a supervised approach since knowing the class label may increase the classifier performance [55]. In addition, a scaled conjugate gradient optimizer, which is suitable for a large scale problem, is used [56].

2) D-FES-SVM: A supervised SVM is usually used for classification or regression tasks. If n is the number of input features, the SVM plots each feature value as a coordinate point in n-dimensional space. Subsequently, a classification process is executed by finding the hyperplane that distinguishes two classes. Although SVM can handle a nonlinear decision border of arbitrary complexity, we use a linear SVM since the nature of the dataset can be investigated by linear discriminant classifiers. The decision boundary for linear SVM is a straight line in two-dimensional space. The main computational property of SVM is the support vectors which are the data points that lie closest to the decision boundary. The decision function of input vector x as expressed by Eq. (9), heavily depends on the support vectors.

$$D(x) = w\vec{x} + b \tag{9}$$

$$w = \sum_{k} \alpha_k y_k x_k \tag{10}$$

$$b = (y_k - w x_k) \tag{11}$$

Eqs. (10) and (11) show the corresponding value of w and b, respectively. From Eq. (9), we can see that decision function $D(x)$ of input vector $\vec{x}$ is defined as the sum of the multiplication of a weight vector and input vector $\vec{x}$ and a bias value. A weight vector w is a linear combination of training patterns. The training patterns with non-zero weights are support vectors. The bias value is the average of the marginal support vectors.

SVM-Recursive Feature Elimination (SVM-RFE) is an application of RFE using the magnitude of the weight to perform rank clustering [28]. The RFE ranks the feature set and eliminates the low-ranked features which contribute less than the other features for classification task [57]. We use the SVM-RFE by using the linear case [28] described in Algorithm 3. The inputs are training instances and class labels. First, we initialize a feature ranked list that is filled by a subset of important features that is used for selecting training instances. We then train the classifier and compute the weight vector of the dimension length. After the value of the weight vector is obtained, we compute the ranking criteria and find the feature with the smallest ranking criterion. Using that feature, the feature ranking list is updated and the feature with the smallest ranking criterion is eliminated. A feature ranked list is finally created as its output.

Algorithm 3 D-FES-SVM Function

```
function D-FES-SVM($InputFeatures$)
    Training SVM
    Compute $w$ (Eq. (10))
    Compute the ranking criteria
    $c_i = {w_i}^2$
    Find the smallest ranking criterion
    $f = argmin(c)$
    Update feature ranked list
    $r = [s(f), r]$
    Eliminate the smallest ranking criterion
    $s = s(1 : f - 1, f + 1 : length(s))$
    $SelectedFeatures \leftarrow s$
    return $SelectedFeatures$
end function
```

3) Decision Tree: C4.5 is robust to noise data and able to learn disjunctive expressions [58]. It has a k-ary tree structure, which can represent a test of attributes from the input data by each node. Every branch of the tree shows potentially selected important features as the values of nodes and different test results. C4.5 uses a greedy algorithm to construct a tree in a top-down recursive divide-and-conquer approach [58]. The algorithm begins by selecting the attributes that yield the best classification result. This is followed by generating a test node for the corresponding attributes. The data are then divided based on the information gain value of the nodes according to the test attributes that reside in the parent node. The algorithm terminates when all data are grouped in the same class, or the process of adding additional separations produces a similar classification result, based on its predefined threshold. The feature selection process begins by selecting the top-three level nodes as explained in Algorithm 4. It then removes the equal nodes and updates the list of selected features.

Algorithm 4 D-FES-C4.5 Function

```
function D-FES-C4.5($InputFeatures$)
    Training C4.5
    $SelectedFeatures$ ← top-three level nodes
    for $i$=1 to $n$ do        ▷ $n$ = size of $SelectedFeatures$
        for $j$=1 to $n$ do
            if $SelectedFeatures[i]$=$SelectedFeatures[j]$ then Remove $SelectedFeatures[j]$
            end if
        end for
    end for
    return $SelectedFeatures$
end function
```

V. Evaluation

A set of experiments was conducted to evaluate the performance of the proposed D-FES method in Wi-Fi impersonation detection. Choosing a proper dataset is an important step in the IDS research field [59]. We employed the AWID Dataset [15] which comprises the largest amount of Wi-Fi network data collected from real network environments. We achieved fair model comparison and evaluation by performing the experiments on the same testing sets as in [15]. We then implement the proposed methodology using MATLAB R2016a and Java code extracted and modified from WEKA packages [60] running on an Intel Xeon E-3-1230v3 CPU @3.30 GHz with 32 GB RAM.

A. Dataset Pre-processing

There are two types of AWID dataset. The first type, named "CLS", has four target classes, whereas the second, named "ATK", has 16 target classes. The 16 classes of the "ATK" dataset belong to the four attack categories in the "CLS" dataset. As an example, the *Caffe-Latte*, *Hirte*, *Honeypot* and *EvilTwin* attack types listed in the "ATK" dataset, are categorized as an impersonation attack in the "CLS" dataset. Based on the size of the data instances included, the AWID dataset comprises both full and reduced versions. In this study, we use the reduced "CLS" for simplicity.

The data contained in the AWID dataset are generally diverse in value, discrete, continuous, and symbolic, with a flexible value range. These data characteristics could make it difficult for the classifiers to learn the underlying patterns correctly [61]. The pre-processing phase thus includes mapping symbolic valued attributes to numeric values, according to the normalization steps and dataset-balancing process described in Algorithm 5. The target classes are mapped to one of these integer valued classes: 1 for normal instances, 2 for an impersonation, 3 for a flooding, and 4 for an injection attack. Meanwhile, symbolic attributes such as receiver, destination, transmitter, and source address are mapped to integer values with a minimum value of 1 and a maximum value, which is the number of all symbols. Some dataset attributes such as the WEP Initialization Vector (IV) and Integrity Check Value (ICV) are hexadecimal data, which need to be transformed into integer values as well. The continuous data such as the timestamps were also left for the normalization step. Some of the attributes have question marks, ?, to indicate unavailable values. We use one alternative in which the question mark is assigned to a constant zero value [62]. After all data are transformed into numerical values, attribute normalization is needed [17]. Data normalization is a process; hence, all value ranges of each attribute are equal. We adopt the mean range method [63] in which each data item is linearly normalized between zero and one in order to avoid the undue influence of

TABLE I
Distribution of Each Class for Both Balanced and Unbalanced Dataset

	Class	Training	Test
Normal	Unbalanced	1,633,190	530,785
	Balanced	163,319	53,078
Attack	Impersonation	48,522	20,079
	Flooding	48,484	8,097
	Injection	65,379	16,682
	Total	162,385	44,858

AWID dataset mimics the natural unbalanced network distribution between normal and attack instances. "Balanced" means to make equal distribution between the number of normal instances (163,319) and thereof total attack instances (162,385). 15% of training data were withdrawn for validation data.

Algorithm 5 Dataset Pre-processing Function

```
function Dataset Pre-processing(Raw Dataset)
    function Dataset Normalization(Raw Dataset)
        for each data instance do
            cast into integer value
            normalize (Eq. (12))
            $NormalizedDataset$
        end for
    end function
    function Dataset Balancing($NormalizedDataset$)
        Pick 10% of normal instances randomly
        $BalancedDataset$
    end function
    $InputDataset$ ← $BalancedDataset$
    return $InputDataset$
end function
```

different scales [62]. Eq. (12) shows the normalizing formula.

$$z_i = \frac{x_i - min(x)}{max(x) - min(x)}, \tag{12}$$

where, z_i denotes the normalized value, x_i refers to the corresponding attribute value and $min(x)$ and $max(x)$ are the minimum and maximum values of the attribute, respectively.

The reduced "CLS" data are a good representation of a real network, in which normal instances significantly outnumber attack instances. The ratio between the normal and attack instances is 10:1 for both unbalanced training and the test dataset as shown in Table I. This property might be biased to the training model and affect the model performance [21] [64]. To alleviate this, we balance the dataset by selecting 10% of the normal instances randomly. However, we set a certain value as the seed of the random number generator for reproducibility purposes. The ratio between normal and attack instances became 1:1, which is an appropriate proportion for the training phase [64]. D-FES is trained using the balanced dataset and then verified on the unbalanced dataset.

B. Evaluation Metrics

We ensured that the evaluation of the performance of D-FES was fair by adopting the most well-referenced model-performance measures [65]: accuracy (Acc), Detection Rate (DR), False Alarm Rate (FAR), Mcc, $Precision$, F_1 score, CPU time to build model (TBM), and CPU time to test the model (TT). Acc shows the overall effectiveness of an algorithm [66]. DR, also known as $Recall$, refers to the number of impersonation attacks detected divided by the total number of impersonation attack instances in the test dataset. Unlike $Recall$, $Precision$ counts the number of impersonation attacks detected among the total number of instances classified as an attack. The F_1 score measures the harmonic mean of $Precision$ and $Recall$. FAR is the number of normal instances classified as an attack divided by the total number of normal instances in the test dataset while FNR shows the number of attack instances that are unable to be detected. Mcc represents the correlation coefficient between the detected and observed data [67]. Intuitively, our goal is to achieve a high Acc, DR, $Precision$, Mcc, and F_1 score and at the same time, maintaining low FAR, TBM, and TT. The above measures can be defined by Eqs. (13), (14), (15), (16), (17), (18), and (19):

$$Acc = \frac{TP + TN}{TP + TN + FP + FN}, \tag{13}$$

$$DR(Recall) = \frac{TP}{TP + FN}, \tag{14}$$

$$Precision = \frac{TP}{TP + FP}, \tag{15}$$

$$FAR = \frac{FP}{TN + FP}, \tag{16}$$

$$FNR = \frac{FN}{FN + TP}, \tag{17}$$

$$F_1 = \frac{2TP}{2TP + FP + FN}, \tag{18}$$

TABLE II
THE EVALUATION OF SAE'S SCHEMES

SAE Scheme	DR (%)	FAR (%)	Acc (%)	F_1 (%)
Imbalance_40 (154:40:10:4)	64.65	**1.03**	96.30	73.13
Imbalance_100 (154:100:50:4)	85.30	1.98	**97.03**	81.75
Balance_40 (154:40:10:4)	72.58	18.87	77.21	74.48
Balance_100 (154:100:50:4)	**95.35**	18.90	87.63	**87.59**

Each model uses either balanced or unbalanced data for the SAE algorithm with following parameters: (input features: number of features in 1^{st} hidden layer: number of features in 2^{nd} hidden layer: target classes).

$$Mcc = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}, \tag{19}$$

where True Positive (TP) is the number of intrusions correctly classified as an attack, True Negative (TN) is the number of normal instances correctly classified as a benign packet, False Negative (FN) is the number of intrusions incorrectly classified as a benign packet, and False Positive (FP) is the number of normal instances incorrectly classified as an attack.

C. Experimental Result

The proposed D-FES is evaluated on a set of experiments. First, we implement and verify different architectures of the feature extractor, SAE. Second, we verify two feature selection approaches: filter-based and wrapper-based methods. We finally validate the usefulness and the utility of D-FES on a realistic unbalanced test dataset.

1) Feature Extraction: We vary the SAE architectures in order to optimize the SAEs implementation with two hidden layers. The features generated from the first encoder layer are employed as the training data in the second encoder layer. Meanwhile, the size of each hidden layer is decreased accordingly such that the encoder in the second encoder layer learns an even smaller representation of the input data. The regression layer with the softmax activation function is then implemented in the final step. The four schemes are examined to determine the SAE learning characteristics. The first scheme, Imbalance_40, has two hidden layers with 40 and 10 hidden neurons in each layer. The second scheme, Imbalance_100, also has two hidden layers; however, it employs 100 and 50 hidden neurons in each layer. Although there is no strict rule for determining the number of hidden neurons, we consider a common rule of thumb [68], which ranges from 70% to 90% from inputs. The third and fourth schemes, named Balance_40 and Balance_100, have the same hidden layer architecture with the first and second schemes, respectively; however, in this case we use the balanced dataset, because the common assumption is that a classifier model built by a highly unbalanced data distribution performs poorly on minority class detection [69]. For our testing purpose, we leverage all four classes contained in the AWID dataset.

Table II shows the evaluation of the SAE schemes. The SAE architectures with 100 hidden neurons have higher *DR* than those with 40 hidden neurons. On the other hand, the SAE architectures with 40 hidden neurons have lower *FAR* than those with 100 hidden neurons. In order to draw a proper conclusion, other performance metrics that consider whole

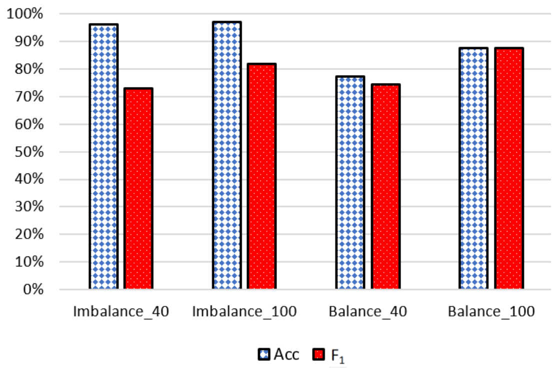


Fig. 5. Evaluation of SAE's Scheme on Acc and F_1 Score. The red bar represents F_1 score while the blue bar represents Acc rate.

classes are needed as the *DR* checks for the attack class only and the *FAR* measures for the normal class only. The *Acc* metric would be affected by the distribution of data, for which different balanced and unbalanced distributions may result in an incorrect conclusion. If we consider the *Acc* metric only as in Fig. 5, we may incorrectly select the Imbalance_100 with 97.03% accuracy, whereas the Balance_100 only achieved 87.63% accuracy. In fact, the Imbalance_100 achieved the highest accuracy rate because of the unbalanced proportion of normal class to attack class. We obtain the best performance by checking F_1 score, for which the Balance_100 has achieved the highest F_1 score among all schemes with 87.59%. Therefore, we choose the SAE architecture with 154:100:50:4 topology.

2) Feature Selection: In order to show the effectiveness of D-FES, we compare the following feature selection methods:

- CfsSubsetEval [70] (CFS) considers the predictive ability of each feature individually and the degree of redundancy between them, in order to evaluate the importance of a subset of features. This approach selects subsets of features that are highly correlated with the class, while having low inter-correlation.
- Correlation (Corr) measures the correlation between the feature and the class in order to evaluate the importance of a subset of features.
- The weight from a trained ANN model mimics the importance of the correspondence input. By selecting the important features only, the training process becomes lighter and faster than before [29].
- SVM measures the importance of each feature based on the weight of the SVM classification results.
- C4.5 is one of the decision tree approaches. It can select a subset of features that is not highly correlated. Correlated features should be in the same split; hence, features that belong to different splits are not highly correlated [58].

A filter-based method usually measures the correlation and redundancy of each attribute without executing a learning algorithm. Therefore, the filter-based method is usually lightweight and fast. On the other hand, the wrapper-based method examines the results of any learning algorithm that outputs a subset of features [71]. CFS and Corr belong to the filter-based methods, whereas ANN, SVM, and C4.5 are wrapper-based methods.

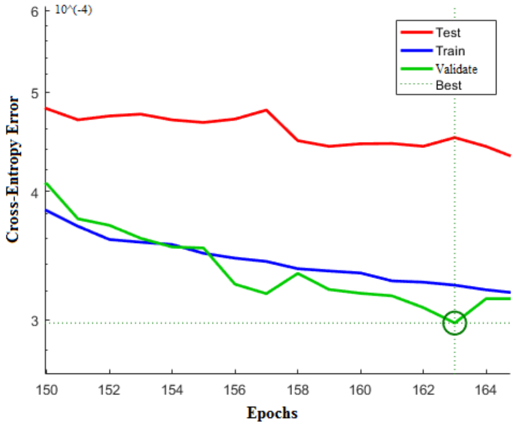


Fig. 6. Cross entropy error of ANN. The best validation performance is achieved at the epoch of 163.

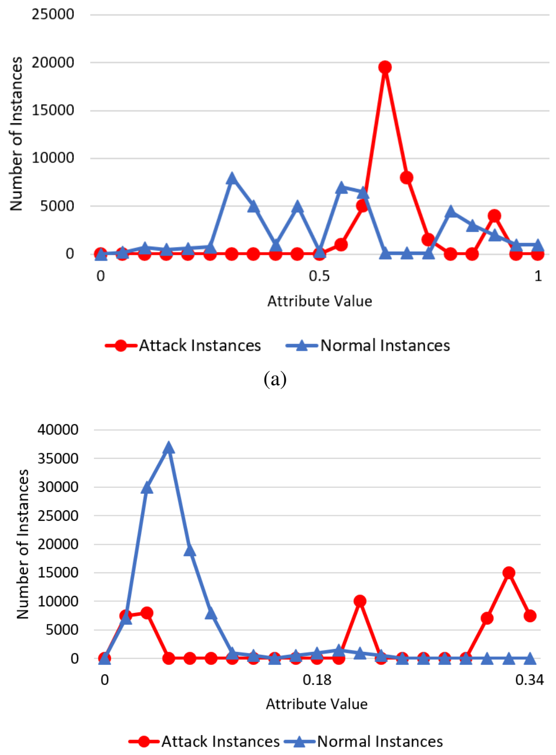
We select a subset of features using the wrapper-based method for considering each feature weight. For ANN, we first set a threshold weight value and if the weight of a feature is greater than the threshold then the feature is selected. The SVM attribute selection function ranks the features based on their weight values. The subset of features with a higher weight value than the predefined threshold value is then selected. Similarly, C4.5 produces a deep binary tree. We select the features that belong to the top-three levels in the tree. CFS produces a fixed number of selected features and Corr provides a correlated feature list.

During ANN training for both feature selection and classification, we optimize the trained model using a separate validation dataset; that is, we separate the dataset into three parts: training data, validation data and testing data in the following proportions: 70%, 15% and 15%, respectively. The training data are used as input into the ANN during training and the weights of neurons are adjusted during the training according to its classification error. The validation data are used to measure model generalization providing useful information on when to terminate the training process. The testing data is used for an independent measure of the model performance after training. The model is said to be optimized when it reaches the smallest average square error on the validation dataset. Fig. 6 shows an example of ANN performance with respect to the cross-entropy error during ANN training. At the epoch of 163, the cross-entropy error, a logarithmic-based error measurement comparing the output values and desired values, starts increasing, meaning that at the epoch of 163, the model is optimized. Although the training data outputs decreasing error values after the epoch point of 163, the performance of the model no longer continues to improve, as the decreasing cross-entropy error may indicate the possibility of overfitting.

Table III contains all the feature lists selected from the various feature selection methods. Some features are essential for detecting an impersonation attack. These are the 4^{th} and the 7^{th}, which are selected by the ANN and SVM and the 71^{st},

TABLE III
FEATURE SET COMPARISONS BETWEEN FEATURE SELECTION AND D-FES

Method	Selected Features	D-FES
CFS	5, 38, 70, 71, 154	38, 71, 154, 197
Corr	47, 50, 51, 67, 68, 71, 73, 82	71, 155, 156, 159, 161, 165, 166, 179, 181, 191, 193, 197
ANN	4, 7, 38, 77, 82, 94, 107, 118	4, 7, 38, 67, 73, 82, 94, 107, 108, 111, 112, 122, 138, 140, 142, 154, 161, 166, 192, 193, 201, 204
SVM	47, 64, 82, 94, 107, 108, 122, 154	4, 7, 47, 64, 68, 70, 73, 78, 82, 90, 94, 98, 107, 108, 111, 112, 122, 130, 141, 154, 159
C4.5	11, 38, 61, 66, 68, 71, 76, 77, 107, 119, 140	61, 76, 77, 82, 107, 108, 109, 111, 112, 119, 158, 160



(a)

(b)

Fig. 7. Characteristics of (a) 38^{th} and (b) 166^{th} features. The blue line represents normal instances while the red line represents attack instances.

TABLE IV
MODEL COMPARISONS ON SELECTED FEATURES

Model	DR (%)	FAR (%)	Acc (%)	F_1 (%)	Mcc (%)	TBM (s)
CFS	94.85	3.31	96.27	92.04	89.67	80
Corr	92.08	0.39	97.88	95.22	93.96	**2**
ANN	99.79	0.47	97.88	99.10	98.84	150
SVM	**99.86**	0.39	**99.67**	99.28	99.07	10,789
C4.5	99.43	**0.23**	99.61	**99.33**	**99.13**	1,294

which is selected by CFS and Corr. The characteristics of the selected features are shown in Figs. 7(a) and 7(b). The blue line indicates normal instances and at the same time, the red line depicts the characteristics of an impersonation attack. We can distinguish between normal and attack instances based on the attribute value of data instances. For example, once a data instance has an attribute value of 0.33 in the 166^{th} feature, the data instance has high probability of being classified as an attack. This could be applied to the 38^{th} and other features as well.

TABLE V
MODEL COMPARISONS ON D-FES FEATURE SET

Model	DR (%)	FAR (%)	Acc (%)	F_1 (%)	Mcc (%)	TBM (s)
CFS	96.34	0.46	98.80	97.37	96.61	1,343
Corr	95.91	1.04	98.26	96.17	95.05	**1,264**
ANN	99.88	0.02	99.95	99.90	99.87	1,444
SVM	**99.92**	**0.01**	**99.97**	**99.94**	**99.92**	12,073
C4.5	99.55	0.38	99.60	99.12	98.86	2,595

TABLE VI
FEATURE SET SELECTED BY D-FES-SVM

Index	Feature Name	Description
47	radiotap.datarate	Data rate (Mb/s)
64	wlan.fc.type_subtype	Type or Subtype
82	wlan.seq	Sequence number
94	wlan_mgt.fixed.capabilities.preamble	Short Preamble
107	wlan_mgt.fixed.timestamp	Timestamp
108	wlan_mgt.fixed.beacon	Beacon Interval
122	wlan_mgt.tim.dtim_period	DTIM period
154	data.len	Length

Table IV lists the performance of each algorithm on the selected feature set only. SVM achieved the highest DR (99.86%) and Mcc (99.07%). However, it requires CPU time of 10,789s to build a model, the longest time among the models observed. As expected, the filter-based methods (CFS and Corr) built their models quickly; however, they attained the lowest Mcc for CFS (89.67%).

Table V compares the performances of the candidate models on the feature sets that are produced by D-FES. SVM again achieved the highest DR (99.92%) and Mcc (99.92%). It also achieved the highest FAR with a value of only 0.01%. Similarly, the lowest Mcc is achieved by Corr (95.05%). This enables us to draw the conclusion that wrapper-based feature selections outperform filter-based feature selections. As SVM showed the best performance, we may consider the properties of selected features by SVM as described in Table VI.

We observe the following patterns from Tables IV and V: Only two out of five methods (Corr anf C4.5) showed lower FAR without D-FES, which we expect to minimize the FAR value of the proposed IDS. This phenomenon might exist because the original and extracted features are not correlated because Corr and C4.5 measure the correlation between each feature. Filter-based feature selection methods require much shorter CPU time compared to the CPU time taken by D-FES. However, D-FES improves the filter-based feature selections performance significantly.

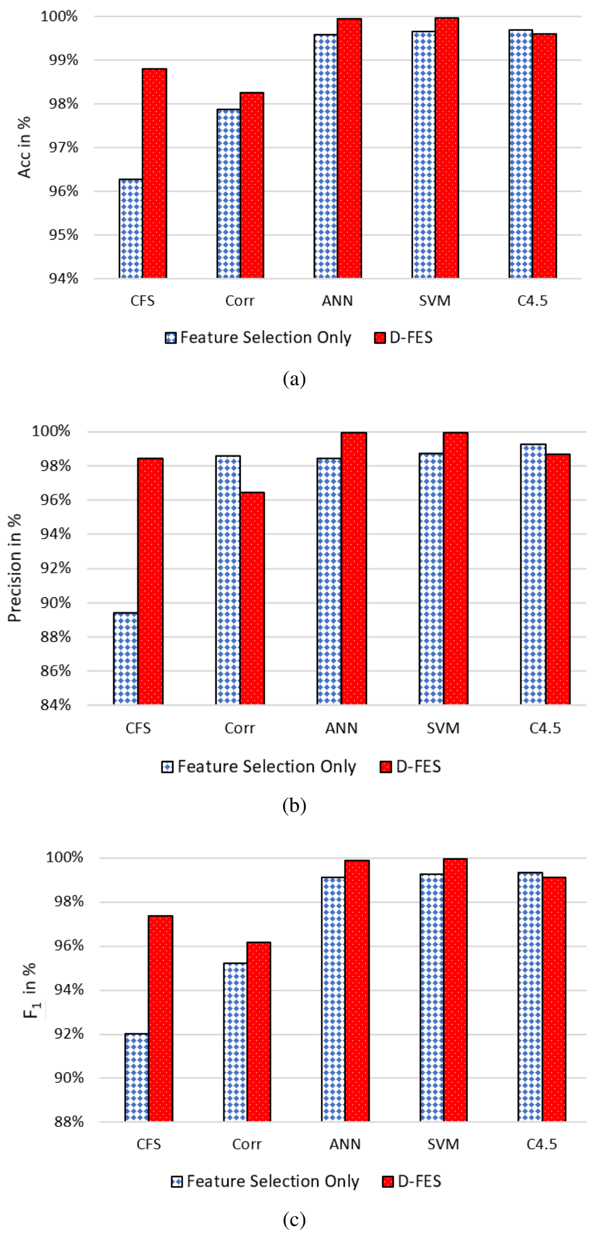


Fig. 8. Models performance comparisons in terms of (a) Acc, (b) $Precision$ and (c) F_1 score. The blue bar represents performances by feature selection only while the red bar represents performances by D-FES.

Similar patterns are captured by Figs. 8(a), 8(b) and 8(c), which depict the performance of different models in terms of Acc, $Precision$ and F_1 score, respectively. D-FES-SVM achieved the highest Acc, $Precision$ and F_1 score of 99.97%, 99.96% and 99.94%, respectively. By D-FES, all methods achieve $Precision$ of more than 96%, shows that D-FES can reduce number of incorrect classification of normal instances as an attack. We can also observe that D-FES improves the Acc of filter-based feature selections significantly. Except the C4.5, all feature selection methods are improved both the Acc and F_1 score by using D-FES. This makes the proposed D-FES a good candidate for an intrusion detector.

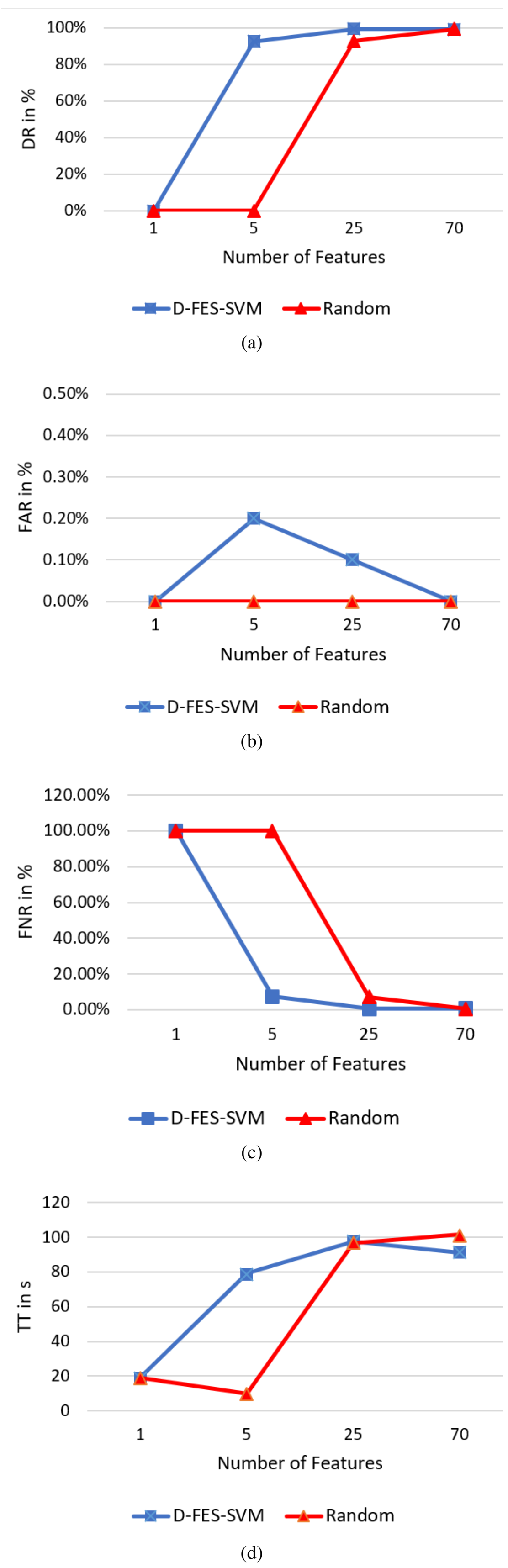


Fig. 9. Model performance comparisons between D-FES and random method in terms of: (a) DR (b) FAR (c) FNR (d) TT

VI. Comparisons with State-of-the-Art Methods

For a fair testing and comparison of the candidate models, we leverage the unbalanced test dataset, as it mimics the real situation of Wi-Fi networks. The unbalanced dataset contains 530,785 normal instances and 20,079 impersonation attack instances as shown in Table I. We compare D-FES-SVM, as the highest F_1 score and randomly selected features with respect to the number of features involved during training as depicted in Figs. 9(a), 9(b), 9(c) and 9(d). The blue line with blue squares shows the performance of the proposed D-FES-SVM method, whereas the red line with red triangles represents selected features randomly. Fig. 9(a) shows that the model learned by D-FES-SVM has the ability to classify the impersonation attack instances with selected features only. The DR of D-FES-SVM has increased insignificantly as the number of features added since the previously selected features are more informative. D-FES-SVM also maintains low *FAR* on five features only; even more, *FAR* of D-FES-SVM further decreased as the number of features added as shown in Fig. 9(b). Although the random method achieved almost perfect *FAR* of 0%, this result does not necessarily mean it is good because the random method classified all data instances as normal instances, thus none of the normal instances are misclassified as an attack and all attack instances are incorrectly detected as a normal instance as shown in 100% of *FNR* in Fig. 9(c). We can see that both D-FES and random method are taking the same amount of time during the testing with 25 features as shown in Fig. 9(d). However, the random method took longer time at 70 features because the 5 initial features selected were inappropriate to distinguish between normal and attack instances as indicated as 0% of DR. Thus, the random method requires less amount of time as it is not capable enough to distinguish attack instances. The CPU time increases as the number of features increases. D-FES-SVM takes longer time than the random method and it increases DR significantly. However, the random method cannot even classify a single impersonation attack.

We also compare the performance of D-FES against our previous work [36] and that of Kolias *et al.* [15]. The experimental results are provided in Table VII. D-FES-SVM classifies impersonation attack instances with a detection rate of 99.918%, whereas it maintains a low *FAR* of only 0.012%. However, as a trade-off, D-FES takes longer time to build the model as seen in Tables IV and V. This computational burden may affect the feasibility to implement D-FES in resource-constrained devices. Nevertheless, we may minimize this by implementing over the clouds or adopting distributed approach [18] using parallel computation for SAE and ANN. The other two methods, D-FES-ANN and D-FES-C4.5 performed comparably to D-FES-SVM, demonstrating that weighted feature selection is useful for improving feature-based learning. Kolias *et al.* [15] tested various classification algorithms such as random tree, random forest, J48, and naïve Bayes, on their own dataset. In terms of an impersonation attack, the naïve Bayes algorithm showed the best performance by correctly classifying 4,419 out of 20,079 impersonation instances. However, it only achieved approximately 22% *DR*, which is unsatisfactory. Our previous approach, combining ANN with SAE, successfully improved the model performance on the impersonation attack detection task [36]. We achieved a *DR* of 84.829% and a FAR of 2.364%. Unlike our previous work [36], this study leverages SAE as a feature extractor and considers the extracted features in the feature selection process in order to achieve a condensed subset of features. We observe the advantage of SAE for abstracting complex and high-dimensional data, as shown by the near-perfect *DR* and *FAR* achieved by D-FES.

TABLE VII
Comparisons with Other Work

Approach	DR (%)	FAR (%)
D-FES-SVM	**99.918**	**0.012**
D-FES-ANN	99.877	0.024
D-FES-C4.5	99.549	0.381
ANN+SAE [36]	84.829	2.364
Kolias *et al.* [15]	22.008	0.021

VII. Conclusion

In this study, we presented a novel method, D-FES, which combines stacked feature extraction and weighted feature selection techniques in order to detect impersonation attacks in Wi-Fi networks. SAE is implemented to achieve high-level abstraction of complex and large amounts of Wi-Fi network data. The model-free properties in SAE and its learnability on complex and large-scale data take into account the open nature of Wi-Fi networks, where an adversary can easily inject false data or modify data forwarded in the network. Extracted features combined with original features were examined using weighted feature selection in order to eliminate redundant and unimportant features. We tested ANN, SVM, and C4.5 for weighted feature selection and observed that a few important features are sufficient to detect impersonation attack instances in large-scale Wi-Fi networks. The proposed methodology achieved a detection rate of 99.918% and a false alarm rate of 0.012%. Clearly, these are the best results on impersonation attacks reported in the literature. In future, we plan to extend D-FES to i) detect any attack classes not limited to impersonation attack only, ii) have the capability to identify an unknown attack that exploits zero-day vulnerability and iii) fit the distributed nature of the IoT environment, which is characterized by limited computing power, memory, and power supply.

Appendix

Table VIII summarizes the abbreviations used in this study.

Acknowledgment

The authors would like to thank the anonymous reviewers for their helpful and constructive comments that greatly contributed to improving the final version of this study.

TABLE VIII
LIST OF ABBREVIATIONS

Term	Abbreviation	Term	Abbreviation
Acc	Accuracy	IDS	Intrusion Detection System
AE	Auto Encoder	IoT	Internet of Things
ANN	Artificial Neural Network	IV	Initialization Vector
AP	Access Point	Mcc	Matthews correlation coefficient
ARP	Address Resolution Protocol	KL	*Kullback-Leibler*
AWID	Aegean Wi-Fi Intrusion Dataset	MITM	Man in the Middle
CFS	CfsSubsetEval [70]	PSO	Particle Swarm Optimization
Corr	Correlation	RSS	Received Signal Strength
CPU	Central Processing Unit	SAE	Stacked Auto Encoder
DFB	Differential Flag Byte	SSID	Service Set Identifier
D−FES	Deep Feature Extraction and Selection	SVM	Support Vector Machine
DR	Detection Rate	SVM −RFE	SVM-Recursive Feature Elimination
FAR	False Alarm Rate	TBM	Time to Build Model
FN	False Negative	TCP/IP	Transmission Control Protocol/Internet Protocol
FNR	False Negative Rate	TN	True Negative
FP	False Positive	TT	Time to Test Model
EPPM	Efficient Probabilistic Packet Marking	TP	True Positive
ESVDF	Enhanced Support Vector Decision Function	WEP	Wired Equivalent Privacy
ICV	Integrity Check Value		

REFERENCES

[1] A. Osseiran, F. Boccardi, V. Braun, K. Kusume, P. Marsch, M. Maternia, O. Queseth, M. Schellmann, H. Schotten, H. Taoka, H. Tullberg, M. A. Uusitalo, B. Timus, and M. Fallgren, "Scenarios for 5G mobile and wireless communications: The vision of the metis project," *IEEE Commun. Mag.*, vol. 52, no. 5, pp. 26–35, May 2014.

[2] C. Kolias, A. Stavrou, J. Voas, I. Bojanova, and R. Kuhn, "Learning internet-of-things security" hands-on"," *IEEE Security Privacy*, vol. 14, no. 1, pp. 37–46, 2016.

[3] C. Kolias, G. Kambourakis, and M. Maragoudakis, "Swarm intelligence in intrusion detection: A survey," *Computers & Security*, vol. 30, no. 8, pp. 625–642, 2011.

[4] A. G. Fragkiadakis, V. A. Siris, N. E. Petroulakis, and A. P. Traganitis, "Anomaly-based intrusion detection of jamming attacks, local versus collaborative detection," *Wireless Communications and Mobile Computing*, vol. 15, no. 2, pp. 276–294, 2015.

[5] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *Proc. Symp. Security and Privacy, Berkeley, California*. IEEE, 2010, pp. 305–316.

[6] G. Anthes, "Deep learning comes of age," *Communications of the ACM*, vol. 56, no. 6, pp. 13–15, 2013.

[7] A. H. Farooqi and F. A. Khan, "Intrusion detection systems for wireless sensor networks: A survey," in *Proc. Future Generation Information Technology Conference, Jeju Island, Korea*. Springer, 2009, pp. 234–241.

[8] I. Butun, S. D. Morgera, and R. Sankar, "A survey of intrusion detection systems in wireless sensor networks," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 266–282, 2014.

[9] R. Mitchell and I. R. Chen, "Behavior rule specification-based intrusion detection for safety critical medical cyber physical systems," *IEEE Trans. Dependable Secure Comput.*, vol. 12, no. 1, pp. 16–30, Jan 2015.

[10] A. Meshram and C. Haas, "Anomaly detection in industrial networks using machine learning: a roadmap," in *Proc. Machine Learning for Cyber Physical Systems, Karlsruhe, Germany*. Springer, 2017, pp. 65–72.

[11] H. Motoda and H. Liu, "Feature selection, extraction and construction," *Communication of IICM (Institute of Information and Computing Machinery), Taiwan*, vol. 5, pp. 67–72, 2002.

[12] J. Zhang, X. Hu, P. Li, W. He, Y. Zhang, and H. Li, "A hybrid feature selection approach by correlation-based filters and svm-rfe," in *Proc. Pattern Recognition (ICPR), Stockholm, Sweden*. IEEE, 2014, pp. 3684–3689.

[13] C. Thomas and N. Balakrishnan, "Improvement in intrusion detection with advances in sensor fusion," *IEEE Trans. Inf. Forensics Security*, vol. 4, no. 3, pp. 542–551, 2009.

[14] W. Wang, X. Wang, D. Feng, J. Liu, Z. Han, and X. Zhang, "Exploring permission-induced risk in android applications for malicious application detection," *IEEE Trans. Inf. Forensics Security*, vol. 9, no. 11, pp. 1869–1882, 2014.

[15] C. Kolias, G. Kambourakis, A. Stavrou, and S. Gritzalis, "Intrusion detection in 802.11 networks: empirical evaluation of threats and a public dataset," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 184–208, 2015.

[16] V. Shah and A. Aggarwal, "Enhancing performance of intrusion detection system against kdd99 dataset using evidence theory," *Int. Journal of Cyber-Security and Digital Forensics*, vol. 5(2), pp. 106–114, 2016.

[17] H. Bostani and M. Sheikhan, "Modification of supervised OPF-based intrusion detection systems using unsupervised learning and social network concept," *Pattern Recognition*, vol. 62, pp. 56–72, 2017.

[18] C. Kolias, V. Kolias, and G. Kambourakis, "Termid: a distributed swarm intelligence-based approach for wireless intrusion detection," *International Journal of Information Security*, vol. 16, no. 4, pp. 401–416, 2017.

[19] H. G. Kayacik, A. N. Zincir-Heywood, and M. I. Heywood, "Selecting features for intrusion detection: A feature relevance analysis on KDD 99 intrusion detection datasets," in *Proc. Privacy, Security and Trust, Canada*. Citeseer, 2005.

[20] S. Puthran and K. Shah, "Intrusion detection using improved decision tree algorithm with binary and quad split," in *Proc. Security in Computing and Communication*. Springer, 2016, pp. 427–438.

[21] N. Y. Almusallam, Z. Tari, P. Bertok, and A. Y. Zomaya, "Dimensionality reduction for intrusion detection systems in multi-data streamsa review and proposal of unsupervised feature selection scheme," *Emergent Computation*, vol. 24, pp. 467–487, 2017.

[22] S. Zaman and F. Karray, "Lightweight IDS based on features selection and IDS classification scheme," in *Proc. Computational Science and Engineering (CSE)*. IEEE, 2009, pp. 365–370.

[23] P. Louvieris, N. Clewley, and X. Liu, "Effects-based feature identification for network intrusion detection," *Neurocomputing*, vol. 121, pp. 265–273, 2013.

[24] Y. Zhu, J. Liang, J. Chen, and Z. Ming, "An improved NSGA-iii algorithm for feature selection used in intrusion detection," *Knowledge-Based Systems*, vol. 116, pp. 74–85, 2017.

[25] V. Manekar and K. Waghmare, "Intrusion detection system using support vector machine (SVM) and particle swarm optimization (PSO)," *Int. Journal of Advanced Computer Research*, vol. 4, no. 3, pp. 808–812, 2014.

[26] H. Saxena and V. Richariya, "Intrusion detection in KDD99 dataset using SVM-PSO and feature reduction with information gain," *Int. Journal of Computer Applications*, vol. 98, no. 6, 2014.

[27] E. Schaffernicht and H.-M. Gross, "Weighted mutual information for feature selection," in *Proc. Artificial Neural Networks, Espoo, Finland*. Springer, 2011, pp. 181–188.

[28] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Machine Learning*, vol. 46, no. 1-3, pp. 389–422, 2002.

[29] Z. Wang, "The applications of deep learning on traffic identification," in *Conf. BlackHat, Las Vegas, USA*. UBM, 2015.

[30] S. Aljawarneh, M. Aldwairi, and M. B. Yassein, "Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model," *Journal of Computational Science*, 2017.

[31] K. Huseynov, K. Kim, and P. Yoo, "Semi-supervised botnet detection using ant colony clustering," in *Symp. Cryptography and Information Security (SCIS), Kagoshima, Japan*, 2014.

[32] K. M. Kim, H. Kim, and K. Kim, "Design of an intrusion detection system for unknown-attacks based on bio-inspired algorithms," in *Computer Security Symposium (CSS), Nagasaki, Japan*, 2015.

[33] M. E. Aminanto, H. Kim, K.-M. Kim, and K. Kim, "Another fuzzy anomaly detection system based on ant clustering algorithm," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 100, no. 1, pp. 176–183, 2017.

[34] H.-C. Shin, M. R. Orton, D. J. Collins, S. J. Doran, and M. O. Leach, "Stacked autoencoders for unsupervised feature learning and multiple organ detection in a pilot study using 4D patient data," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1930–1943, 2013.

[35] M. Usha and P. Kavitha, "Anomaly based intrusion detection for 802.11 networks with optimal features using SVM classifier," *Wireless Networks*, vol. 22, pp. 1–16, 2016.

[36] M. E. Aminanto and K. Kim, "Detecting impersonation attack in Wi-Fi networks using deep learning approach," in *Proc. Workshop of Information Security Applications (WISA), Jeju Island, Korea*. Springer, 2016, pp. 136–147.

[37] M. E. Aminanto, H. C. Tanuwidjaja, P. D. Yoo, and K. Kim, "Weighted feature selection techniques for detecting impersonation attack in Wi-Fi networks," in *Symp. Cryptography and Information Security (SCIS), Naha, Japan*, 2017.

[38] M. Barbeau, J. Hall, and E. Kranakis, "Detecting impersonation attacks in future wireless and mobile networks," in *Proc. Secure Mobile Ad-hoc Networks and Sensors, Singapore*. Springer, 2006, pp. 80–95.

[39] M. S. Ahmad and V. Ramachandran, "Cafe latte with a free topping of cracked WEP retrieving WEP keys from road warriors," in *Conf. ToorCon, San Diego, USA*. TOORCON, 2007.

[40] Aircrack-ng, "Airbase-ng," http://www.aircrack-ng.org/doku.php?id=airbase-ng#hirte_attack_in_access_point_mode, 2010, [Online; accessed 12-December-2016].

[41] R. Beyah, S. Kangude, G. Yu, B. Strickland, and J. Copeland, "Rogue access point detection using temporal traffic characteristics," in *Proc. Conf. Global Telecommunications (GLOBECOM), Dallas, Texas*, vol. 4. IEEE, 2004, pp. 2271–2275.

[42] L. Malisa, K. Kostiainen, M. Och, and S. Capkun, "Mobile application impersonation detection using dynamic user interface extraction," in *Proc. European Symp. on Research in Computer Security(ESORICS), Heraklion, Greece*. Springer, 2016, pp. 217–237.

[43] O. Goga, G. Venkatadri, and K. P. Gummadi, "The doppelgänger bot attack: Exploring identity impersonation in online social networks," in *Proc. Internet Measurement Conf. (IMC), Tokyo, Japan*. ACM, 2015, pp. 141–153.

[44] T. Shang and L. Y. Gui, "Identification and prevention of impersonation attack based on a new flag byte," in *Proc. Int. Conf. Computer Science and Network Technology (ICCSNT), Harbin, China*, vol. 1. IEEE, 2015, pp. 972–976.

[45] M. H. Yılmaz and H. Arslan, "Impersonation attack identification for secure communication," in *Proc. Globecom Workshops (GC Wkshps), Atlanta, USA*. IEEE, 2013, pp. 1275–1279.

[46] I. B. Lakshmi, B. S. Lakshmi, and R. Karthikeyan, "Detection and prevention of impersonation attack in wireless networks," *Int. Journal of Advanced Research in Computer Science and Technology (IJARCST)*, 2014.

[47] F. Palmieri, U. Fiore, and A. Castiglione, "A distributed approach to network anomaly detection based on independent component analysis," *Concurrency and Computation: Practice and Experience*, vol. 26, no. 5, pp. 1113–1129, 2014.

[48] Q. Xu, C. Zhang, L. Zhang, and Y. Song, "The learning effect of different hidden layers stacked autoencoder," in *Proc. Int. Con. Intelligent Human-Machine Systems and Cybernetics (IHMSC), Zhejiang, China*, vol. 02. IEEE, Aug 2016, pp. 148–151.

[49] H. Shafri and F. Ramle, "A comparison of support vector machine and decision tree classifications using satellite data of langkawi island," *Information Technology Journal*, vol. 8, no. 1, pp. 64–70, 2009.

[50] R. Rojas, "The backpropagation algorithm," in *Neural networks*. Springer, 1996, pp. 149–182.

[51] B. A. Olshausen and D. J. Field, "Sparse coding with an overcomplete basis set: A strategy employed by v1?" *Vision Research*, vol. 37, no. 23, pp. 3311–3325, 1997.

[52] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo, "A geometric framework for unsupervised anomaly detection," *Applications of Data Mining in Computer Security*, vol. 6, pp. 77–101, 2002.

[53] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle *et al.*, "Greedy layer-wise training of deep networks," *Advances in neural information processing systems*, vol. 19, pp. 153–160, 2007.

[54] Y. Bengio, Y. LeCun *et al.*, "Scaling learning algorithms towards AI," *Large-scale kernel machines*, vol. 34, no. 5, pp. 1–41, 2007.

[55] L. Guerra, L. M. McGarry, V. Robles, C. Bielza, P. Larrañaga, and R. Yuste, "Comparison between supervised and unsupervised classifications of neuronal cell types: a case study," *Developmental neurobiology*, vol. 71, no. 1, pp. 71–82, 2011.

[56] M. F. Møller, "A scaled conjugate gradient algorithm for fast supervised learning," *Neural Networks*, vol. 6, no. 4, pp. 525–533, 1993.

[57] X. Zeng, Y.-W. Chen, C. Tao, and D. van Alphen, "Feature selection using recursive feature elimination for handwritten digit recognition," in *Proc. Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), Kyoto, Japan*. IEEE, 2009, pp. 1205–1208.

[58] C. A. Ratanamahatana and D. Gunopulos, "Scaling up the naive Bayesian classifier: Using decision trees for feature selection," in *Workshop on Data Cleaning and Preprocessing (DCAP) at IEEE Int. Conf. Data Mining (ICDM), Maebashi, Japan*. IEEE, 2002.

[59] A. Özgür and H. Erdem, "A review of KDD99 dataset usage in intrusion detection and machine learning between 2010 and 2015," *PeerJ PrePrints*, vol. 4, p. e1954v1, 2016.

[60] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *ACM SIGKDD Explorations Newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[61] M. Sabhnani and G. Serpen, "Application of machine learning algorithms to KDD intrusion detection dataset within misuse detection context." in *Proc. Int. Conf. Machine Learning; Models, Technologies and Applications (MLMTA), Lax Vegas, USA*, 2003, pp. 209–215.

[62] D. T. Larose, *Discovering knowledge in data: an introduction to data mining*. John Wiley & Sons, 2014.

[63] W. Wang, X. Zhang, S. Gombault, and S. J. Knapskog, "Attribute normalization in network intrusion detection," in *Proc. Int. Symp. Pervasive Systems, Algorithms, and Networks (ISPAN), Kaohsiung, Taiwan*. IEEE, Dec. 2009, pp. 448–453.

[64] Q. Wei and R. L. Dunbrack Jr, "The role of balanced training and testing data sets for binary classifiers in bioinformatics," *Public Library of Science (PloS) one*, vol. 8, no. 7, pp. 1–12, 2013.

[65] O. Y. Al-Jarrah, O. Alhussein, P. D. Yoo, S. Muhaidat, K. Taha, and K. Kim, "Data randomization and cluster-based partitioning for botnet intrusion detection," *IEEE Trans. Cybern.*, vol. 46, no. 8, pp. 1796–1806, 2015.

[66] M. Sokolova, N. Japkowicz, and S. Szpakowicz, "Beyond accuracy, F-score and ROC: a family of discriminant measures for performance evaluation," in *Australasian Joint Conference on Artificial Intelligence, Hobart, Australia*. Springer, 2006, pp. 1015–1021.

[67] P. D. Schloss and S. L. Westcott, "Assessing and improving methods used in operational taxonomic unit-based approaches for 16s rRNA gene sequence analysis," *Applied and environmental microbiology*, vol. 77, no. 10, pp. 3219–3226, 2011.

[68] Z. Boger and H. Guterman, "Knowledge extraction from artificial neural network models," in *Proc. Int. Conf. Systems, Man, and Cybernetics, Orlando, USA*, vol. 4. IEEE, 1997, pp. 3030–3035.

[69] G. M. Weiss and F. Provost, *The effect of class distribution on classifier learning: an empirical study*. Rutgers University, Tech. Rep. ML-TR-44, 2001.

[70] M. A. Hall and L. A. Smith, "Practical feature subset selection for machine learning," in *Proc. Australasian Computer Science Conference (ACSC), Perth, Australia*. Springer, 1998, pp. 181–191.

[71] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artificial Intelligence*, vol. 97, no. 1, pp. 273–324, 1997.

Muhamad Erza Aminanto received the B.S. and M.S. degrees in Electrical Engineering from Bandung Institute of Technology (ITB), Indonesia in 2013 and 2014, respectively. He is pursuing his Ph.D in the School of Computing at Korea Advanced Institute of Science and Technology (KAIST), South Korea.

His current research interests include machine-learning, intrusion detection systems and big data analytics.

Rakyong Choi received the B.S. and M.S. degrees in the Department of Mathematical Sciences in the Korea Advanced Institute of Science and Technology (KAIST), South Korea in 2011 and 2013, respectively. He is pursuing his Ph.D. in the School of Computing at KAIST, South Korea.

His current research interests include post-quantum Cryptography, fully homomorphic signatures, fully homomorphic encryption and lattice-based cryptography.

Harry Chandra Tanuwidjaja received the B.S. and M.S. degrees in Electrical Engineering from the Bandung Institute of Technology (ITB), Indonesia in 2013 and 2015, respectively. He is pursuing his Ph.D in the School of Computing at the Korea Advanced Institute of Science and Technology (KAIST), South Korea.

His current research interests include malware detection, machine-learning, and intrusion detection systems.

Paul D. Yoo (M'11−SM'13) is currently with the Cranfield Defence and Security, based at the United Kingdom Ministry of Defence establishment on the Oxfordshire/Wiltshire borders. Prior to this, Dr Yoo held academic/research posts in Sydney, Bournemouth, and the UAE. Dr Yoo serves as Editor of IEEE Communications Letters and IEEE Transactions of Sustainable Computing journals and holds over 60 prestigious journal and conference publications. Dr Yoo is affiliated with University of Sydney and Korea Advanced Institute of Science and Technology (KAIST) as Visiting Professor. He is a Senior Member of IEEE and a Member of BCS. His research focuses on large-scale data analytics including design and development of computational models and algorithms inspired by intelligence found in physical, chemical and biological systems and to solve practical problems in security and forensic computing.

Kwangjo Kim (M'12) received the B.Sc. and M.Sc. degrees in electronic engineering from Yonsei University, Seoul, Korea, in 1980 and 1983, respectively, and the Ph.D. degree from the Division of Electrical and Computer Engineering, Yokohama National University, Yokohama, Japan, in 1991.

He was a Visiting Professor with the Massachusetts Institute of Technology, Cambridge, MA, USA and the University of California at San Diego, La Jolla, CA, USA, in 2005 and the Khalifa University of Science, Technology and Research, Abu Dhabi, UAE, in 2012 and an Education Specialist with the Bandung Institute of Technology, Bandung, Indonesia, in 2013. He is currently a Full Professor with the School of Computing and Graduate School of Information Security, Korea Advanced Institute of Science and Technology, Daejeon, Korea, the Korean representative to IFIP TC-11 and the honorable President of the Korea Institute of Information Security and Cryptography (KIISC). His current research interests include the theory of cryptology, information security and its applications. Prof. Kim served as a Board Member of the International Association for Cryptologic Research (IACR) from 2000 to 2004, the Chairperson of the Asiacrypt Steering Committee from 2005 to 2008 and the President of KIISC in 2009. He is a Fellow of IACR and a member of IEICE, IEEE, and ACM.