TV-ADS: A Smarter Attack Detection Scheme Based on Traffic Visualization of Wireless Network Event Cell

Zhiwei Zhang[1,2], Guiyuan Tang[2], Baoquan Ren[1], Hongjun Li[1], Yulong Shen[2]*

[1] Academy of Systems Engineering, Academy of Military Sciences, China
[2] School of Computer Science and Technology, Xidian University, China
zwzhang@xidian.edu.cn, tanggy@stu.xidian.edu.cn, {renbq, lihongjun86}@126.com, ylshen@mail.xidian.edu.cn

Abstract

To protect the increasing cyberspace assets, attack detection systems (ADSs) as well as intrusion detection systems (IDSs) have been equipped in various network environments. Recently, with the development of big data, machine learning, deep learning, neural networks and other artificial intelligence (AI) technologies, more and more ADSs/IDSs based on Artificial Intelligence are presented in academia and industry. Particularly, depending on the outstanding performance and efficiency in recognizing and classifying images, computer vision algorithms have been employed to detect malicious software and malicious traffic. However, we found that in wireless networks, the results vary significantly depending on the mapping methods used to transform the original network traffic data into visual images. Therefore, in this paper, we propose an AI-based attack detection scheme (TV-ADS) by introducing a novel traffic-image mapping method, which segments the sequential network traffic into individual event cells and transforms variant images to a uniform standard size, and design a CNN model to recognize normal and malicious traffics with these visible network event images. Finally, the results of our experiments on the AWID3 dataset demonstrate that our TV-ADS outperforms the existing schemes in terms of accuracy, precision, recall, F1-score and efficiency.

Keywords: Wireless network, Attack detection, Image visualization, Convolutional neural network

1 Introduction

In today's digital era, wireless networks have pervaded all aspects of our daily routine. From basic home Wi-Fi connections to wireless LANs in major corporations, and even the widespread integration of IoT devices. Wireless advancements have granted us unparalleled convenience and flexibility, rendering wireless technology an essential component of modern existence and employment. According to the 2023 Global Networking Trends report by Cisco, the number of global Wi-Fi hotspots has quadrupled from 2018 to 2023, and wireless connectivity will account for 45% of all connected devices by 2023 [1]. Although wireless networks offer significant convenience in our daily lives, they are susceptible to various security threats, especially attacks targeting the physical and data link layers [2]. However, research on machine learning-based attack detection systems specialized in detecting Wi-Fi attacks is lagging behind, making attack detection in wireless networks an important and urgent research topic.

Compared with traditional wired networks, wireless networks have many inherent security risks. Firstly, due to the special propagation mode of wireless signals, attackers can carry out eavesdropping attacks without establishing a direct physical connection with the target network. This situation provides the possibility of "man-in-the-middle", sniffing and spoofing attacks. Secondly, since wireless networks are usually simple to access, it is easier for malicious users to join the network and launch internal attacks [3]. In view of the above security threats, wireless network attack detection has emerged as a crucial approach for ensuring the security of wireless networks. Its main purpose is to discover and respond to potential threats and attacks in real-time by monitoring and analyzing wireless network traffic. In the field of network attack detection, machine learning shows promising research and application value [4-5], which can effectively identify attacks by learning from a vast amount of network traffic datasets, extracting meaningful features and building models [6-7].

In the past few years, deep learning has made significant progress in several research and application areas [8-11], and it focuses on learning from large amounts of data using multi-layer neural network models for feature extraction and pattern recognition. One key architecture in deep learning is the Convolutional Neural Network (CNN) [12], which is particularly suited for processing images and audio data. CNNs can learn features directly from raw image data, eliminating the need for complex pre-processing and feature engineering. This method of learning from low-level to high-level features enables CNN to achieve exceptional performance in image recognition tasks. There are related studies that consider transforming other forms of data into images for analysis using image processing and computer vision techniques. For example, for time series data, Wang et al. [13] transformed time series data into various types of GAF and MTF images to classify them with computer vision techniques. For textual data, Saharia et al. [14] proposed a text-to-image model that can transform textual content into images. For malware detection, Wang et al. [15] introduced

*Corresponding Author: Yulong Shen; E-mail: ylshen@mail.xidian.edu.cn
DOI: 10.53106/160792642024032502012

traffic visualization for the classification of malware traffic in 2017. This data transformation method made full use of CNN's successful experience in the image field to discover patterns and associations in the data with the help of computer vision technology, which promoted research and application in related fields.

In the field of attack detection in wireless network, the incorporation of deep learning techniques in wireless network attack detection has opened up a novel perspective on conventional approaches. However, CNNs are primarily utilized for image processing, which diverges considerably from the data structure created within the realm of wireless networks. Therefore, this paper proposes a method to visualize wireless network traffic data as images. Each pixel in the image represents information from a specific segment of a network packet. This transformation method provides us with a new perspective that allows us to observe and analyze network traffic from an image perspective. To detect potential attack patterns from these images, we design and train a CNN model. The formed images are then classified and identified using the CNN, allowing for binary classification of images and identifying normal and attack traffic, which ultimately leads to efficient attack detection. During the experimental phase, we use the well-known 802.11 oriented AWID family of datasets, specifically the AWID3 wireless network traffic dataset released in 2021 [16], which significantly complements and extends the previous AWID2 dataset by capturing and investigating the traces of various attacks initiated in IEEE 802.1X Extensible Authentication Protocol (EAP) environments set.

In summary, our contributions in this paper are as follows:

- Packet to image transformation. To tackle the challenge of detecting attacks on wireless network traffic, we propose a framework for attack detection techniques that transform wireless network traffic into color images, providing high scalability and detection accuracy. To evaluate encrypted traffic, we conducted an evaluation consisting of 7 attacks.
- Avoiding manual feature extraction. To improve the efficiency and accuracy of the analysis, we use unprocessed PCAP network traffic packets, split for direct examination without requiring any complex feature engineering or domain expertise. This method not only simplifies the attack detection process, but also outperforms traditional attack detection methods based on feature extraction in terms of detection performance.
- Simple 2D-CNN structure for classification. Using simpler 2D-CNN greatly reduces model complexity. We design a 2D-CNN deep learning model to evaluate the effectiveness of visualization-based detection in differentiating between normal and attack images. This model significantly improves classification efficiency and accuracy, achieving a classification accuracy of 99.90% and precision of 99.88%.

The rest of this paper is organized as follows: Section 2 presents the related work on attack detection in wireless networks. In section 3, we analyze the PCAP file and introduce the wireless traffic data splitting method. Furthermore, section 4 describes the event cell-based image transformation method. We discuss our attack traffic image detection model and its implementation in section 5. Section 6 presents and discusses the experimental results of our model. Finally, Section 7 summarizes our work and analyzes the study's limitations and future research directions.

2 Related Work

With the rapid advancement of wireless network technology, wireless network attack detection has become a significant research area in the field of network security. Researchers have explored different approaches to combat these security threats in light of the continuous evolution of attack methods. In this section, we present an analysis of the historical and modern developments in wireless network attack detection methods across three categories: feature-based, behavior-based, and deep learning (DL)-based. Special attention is given to deep learning-based methods that use visualization for attack detection.

2.1 Feature-based Wireless Attack Detection Methods

Feature-based network attack detection methods heavily rely on predefined attack rules or feature libraries to identify malicious activities. When the detected activity matches a known attack signature, the system generates an alert. Although this type of method typically has a high level of accuracy, it often faces difficulties in recognizing novel or previously unknown attacks.

Regarding the feature selection challenge, Riyaz et al. [17] proposed a feature selection algorithm based on conditional random fields and linear correlation coefficients. This algorithm effectively identified features with significant contributions, resulting in an impressive classification accuracy of 98.88% using CNN. Ahmim et al. [18] circumvented manual analysis by employing rough set theory for feature selection through a data mining approach based on support vector machines. Wang et al. [19] presented an IDS based on sparse logistic regression for feature selection, unifying feature selection and classification in a single framework, ultimately achieving an experimental accuracy of 97.86%.

To address the issue of excessive features in the original dataset, Jiang et al. [20] employed Z-score normalization for data, applied a compressive sampling method to reduce the dimensionality of features, and combined this with SVM for classifying the compressed results, effectively detecting denial-of-service attacks, probe attacks, and other intrusion attempts. Chowdhury et al. [21] introduced a network intrusion detection system based on optimal features, achieving a real-time traffic detection rate of 87.43%. Yang et al. [22] proposed an improved convolutional neural network-based wireless network intrusion detection method, this yielded an accuracy improvement of 8.82% compared to the use of LeNet-5 [23] and 0.51% improvement with regard to the use of DBN [24].

In summary, feature-based attack detection methods for wireless networks offer high efficiency, accuracy, and ease

of implementation and management. However, they also suffer from drawbacks such as over-reliance on known attack patterns and the challenge of dealing with unknown attacks.

2.2 Behavior-based Wireless Attack Detection Methods

Behavior-based attack detection methods for wireless networks rely on the establishment of a baseline for normal network or system behavior. When detected behavior deviates from this baseline, the system identifies it as a potential attack and triggers an alert. The advantage of this approach is that it doesn't depend on known attack patterns and can detect unknown or novel attack types. However, it has a drawback of relatively high false alarm rates, as demonstrated by Hall et al. [25] in their study of an anomaly-based detection method, where the false alarm rate reached as high as 100%.

Butun et al. [26] implemented Min-Max normalization to the dataset and modeled normal behavior using a Gaussian mixture model to identify anomalous data. Regarding program system invocation behavior, Hoang et al. [27] proposed a fuzzy inference mechanism to establish a soft boundary between abnormal and normal behavior, leading to a significant reduction of false alarms by 48% in their empirical demonstration. Denning et al. [28] identified attacking behaviors by modeling parameters as independent Gaussian random variables. They defined a specific range of values for univariate variables, and identified data that deviates significantly from this range as attacking behavior. Ye et al. [29] introduced a multivariate quality control technique to establish a comprehensive normative record of normal activity in an information system and to detect intrusions using this record. García-Teodoro et al. [30] introduced a model based on time series analysis, employing interval timers and event counters. This model classifies traffic instances as anomalous if they manifest with an unexpectedly low probability during a specified timeframe.

In summary, behavior-based methods for detecting wireless network attacks have advantages in detecting unknown attacks, but also face challenges such as establishing a behavioral baseline, dealing with a high false alarm rate, and the need for significant computational resources.

2.3 DL-based Wireless Attack Detection Methods

The deep learning methods circumvent the need for manual design and feature engineering, which are traditionally labor-intensive and time-consuming. Moreover, the use of image-based data representation offers a unique perspective for network traffic analysis. By converting complex data into visual information, researchers attempt to detect network intrusions using image processing techniques [31-36].

Rong et al. [37] preprocessed the raw conversational network traffic by converting it into fixed-size RGB images and subsequently trained a deep transfer learning model based on ResNet-50 [38] for the detection of previously unseen malware samples. Ahmad et al. [39] utilized the GAF technique to convert Wi-Fi frames into images, which were subsequently fed into a deep-based intrusion detection system for intrusion detection. Similarly, to fully leverage CNNs for processing network traffic data. Feng et al. [40] introduced a visualization-based V-CNN model, which provided a visual representation of the data prior to the CNN model, making it suitable for image recognition. Lin et al. [41] employed Cycle-GAN to convert the dataset into images, and subsequently utilized it to learn normal data images for generating anomalous data images. The generated composite anomalous data was then combined with the original data for model training. Genge et al. [42] transformed one-dimensional data into two-dimensional image data through data padding to identify attacks by learning effective features.

Recent research has emphasized the substantial potential of machine learning and deep learning methods for identifying cyberattacks. However, several persistent challenges still need to be addressed. Firstly, the present feature-to-image transformation process is notably intricate, entailing complicated feature extraction and screening. The integration of this feature-to-image transformation with standardized raw data processing markedly increases the time required for detection. Consequently, this extended timeframe diminishes the adaptability of the system in environments where real-time performance is imperative. Secondly, many studies overlook the unique challenges posed by wireless networks. Unlike their wired counterparts, wireless networks introduce greater uncertainties in transmission mediums, speeds, and device connectivity. This complexity is compounded in encrypted wireless traffic, where obtaining high-level feature information becomes difficult. Such constraints complicate the deployment of detection methods reliant on application-layer features. Moreover, most current methods rely on complex classification algorithms that increase information processing time and require higher hardware configurations. In stark contrast, this paper introduces a pioneering approach: we present a wireless network traffic visualization technique that bypasses feature extraction and harnesses a simpler 2D-CNN neural network for classification. This significantly enhances attack detection efficiency and accuracy.

3 Traffic Data Splitting Method

In this section, we analyze the PCAP file, and introduce a packet splitting method that efficiently splits wireless network traffic based on event cell.

3.1 PCAP Analysis

PCAP (Packet Capture) is a network packet capture file format that is commonly used to store network traffic data. It can be employed by various network tools, such as Wireshark and TCPDump, for capturing and analyzing network traffic. This format is widely used in network traffic analysis, troubleshooting, and intrusion detection. PCAP file provides detailed information about network packets, including source/destination IP addresses, ports, protocols, and payload content. For attack detection, PCAP file analysis is invaluable for detecting anomalous network behavior. However, when dealing with large-scale network traffic data, handling a single large PCAP file can be challenging, necessitating the need for file splitting. Splitting simplifies the management and processing of PCAP files, making it easier to identify and

analyze security events that occur at specific times, such as intrusion attempts and malicious activities.

PCAP's feature engineering involves data preprocessing, feature extraction, and normalization, along with the integration of features for input classification. However, this process adds overhead to attack detection, and the feature extraction process can result in the loss of critical information. With the increasing complexity and diversity of network traffic, relying solely on predefined features may not be sufficient to capture all types of attack behaviors. In this paper, we depart entirely from the feature extraction process to develop a featureless deep learning method that is more conducive to attack detection. By splitting and visualizing PCAP files, we transform the binary data within these files into images, enabling the use of image analysis techniques for direct detection of anomalous patterns in network traffic.

3.2 Wireless Network Traffic Splitting

Advancements in deep learning technologies now permit us to train models by employing a direct and streamlined transformation of raw data. This enables the automatic learning of valuable feature representations from the raw data. The raw PCAP files collected are often large and challenging to analyze. There are various methods regarding the splitting of wireless network traffic, such as by packet, session, stream, and time. However, the methods of splitting by stream or by session are limited by whether the traffic is encrypted or not, and the method of splitting by time leads to too much difference in the size of transformed images.

In this paper, we propose an event cell-based splitting method, where an event cell is a specific network packet or frame captured at a specific time, and each event cell represents a single network communication event in wireless network communication. In a PCAP file, an event cell represents one line of data in the file, i.e., a single packet. This method of event cell splitting is both general and flexible enough to be independent of any particular network type or structure, and provides a fine-grained perspective for attack detection studies of wireless network traffic.

4 Transform Event Cell into Color Image

In this section, we propose a method for transforming wireless network traffic packets into images after splitting them based on event cell, and we introduce two methods for image scaling.

4.1 Image Generation Method

We use the event cell-based splitting strategy to split the individual packets in a PCAP document into separate PCAP documents. We transform raw traffic packet files from wireless network devices into colorful images. This process involves multiple steps designed to transform complex network data into a visual format that enhances our ability to understand and analyze it.

Each packet is initially stored as a sequence of bytes in the PCAP file. To visualize the data, we convert the bytes into hexadecimal strings. In this conversion process, each byte (comprising 8 bits) is represented by 2 hexadecimal characters. Consequently, each packet results in a lengthy string of hexadecimal characters. Subsequently, we divide these hexadecimal characters into 6 consecutive character segments. Each segment represents a 24-bit (3-byte) block of data, which corresponds to a color value. This color value comprises three channels: red (R), green (G), and blue (B). Each of the two hexadecimal characters is interpreted as a decimal integer ranging from 0 to 255, representing the value of the respective RGB color channel. This method allows us to represent the information from the original traffic packet within the image. Different regions of the color image correspond to different segments of the original traffic packet file structure. The pseudo-code for this transformation algorithm is shown in Algorithm 1.

Algorithm 1. Conversion from PCAP to color image

```
Input: Raw Network Traffic Packets
Output: Color Image
function HEX_TO_RGB(hex_str)
    R ← int (hex_str[0:2], 16)
    G ← int (hex_str[2:4], 16)
    B ← int (hex_str[4:6], 16)
    return (R, G, B)
end function
function PCAP_TO_COLOR_IMAGE(Input)
    packets ← readFromPCAPfile(Input)
    hex_str_value ← [ ]
    for each packet in packets do
      hex_str ← convert packetbytes to hexadecimal String
      hex_str_values.append(hex_str)
    end for
    rgb_values ← [ ]
    for each hex_str in hex_str_values do
      for i=0 to len(hex_str) step 6 do
          segment ← hex_str[i:i+6]
          if len(segment) == 6 then
              color ← HEX_TO_RGB(segment)
              rgb_values.append(color)
          end if
      end for
    end for
    image ← create color image using rgb_values
    return image
end function
```

4.2 Image Scaling Methods

To ensure consistency and validity of the produced color images, we employ a strategy of fixing the width of the image, which helps determine the extent of image scaling. Our method is to choose a fixed width, typically a power of 2 pixels, and to scale the height of the image adaptively based on the event cell size. This selection of width is critical in the subsequent processing and analysis of the images. This process facilitates the conversion of event cells into images of uniform dimensions, enabling effective data representation. The image height is used as an adaptive parameter to fit the original event cell size. The height is determined by dividing the total pixel count by an integer 2^n greater than the width. This method ensures that the image information remains consistent with the event cell size and adjusts to the distinct nature of various packets. It is crucial to preserve the integrity of the data and information content, as event cells of varying

sizes necessitate different image sizes for accurate depiction of their contents. The image is constructed using a systematic filling of pixels from left-to-right and top-to-bottom. This padding technique ensures the accuracy of the sequence of information in the event cell as it is presented in the image.

The generated image is analyzed, and the length and width of the image are resized to create a new image. We analyze and compare two image scaling methods, a crop and pad method which maintains a fixed number of pixels through crop or pad operations and uses pixel elimination for pixels that are outside a specified range. For pixel shortages, we use pixel repetition alignment to ensure that the height and width of the new image remain consistent. The reason for using this scaling method is that for the high-level information that is encrypted in the event cell, we consider it to be information that is irrelevant for attack detection. This information is cropped out when the generated image is cropped. Another method is to perform a bilinear interpolation on the generated image so that the size of the image remains consistent. Figure 1 shows the effect of using two image scaling methods for an event cell.

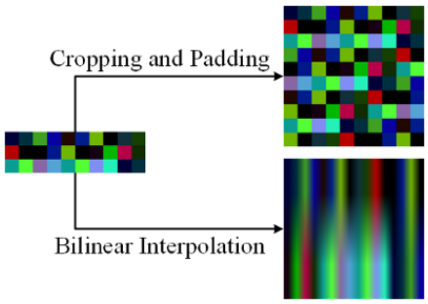


Figure 1. The effect of two types of image scaling

5 CNN Model Design

The attack traffic image detection model employs deep learning techniques to process the transformed wireless traffic images. To detect attacks on wireless network traffic, a binary classification model is developed. The labeled dataset is utilized to train the 2D-CNN model, enabling it to accurately classify input images into specific classes. Subsequently, new images of wireless network traffic are classified by this model when making predictions.

The described 2D-CNN structure mainly comprises two parts: a feature extraction component and a feature mapping component. The feature extraction part includes a convolutional layer and a pooling layer that collaborate to automatically extract meaningful features from the input image. The feature mapping part consists of a sequence of fully connected layers, aimed at utilizing the features obtained from the feature extraction part for practical classification tasks. The structure of the 2D-CNN is illustrated in Figure 2.

With this 2D-CNN model, image data undergoes a series of processing steps, starting with the input of a color image with three channels. The input layer receives this data stream, which then passes through a 2D convolutional layer. In this convolutional layer, 16 filters of size 3x3 are applied, and the Rectified Linear Unit (ReLU) activation function is employed. The primary objective of the convolutional layer is to extract essential features from the input image.

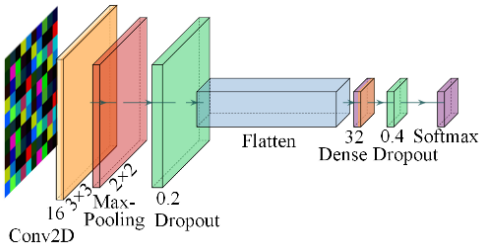


Figure 2. The structure of the 2D-CNN

Subsequently, the data passes through a max-pooling layer with a filter size of 2x2. Pooling layers are typically used to reduce the spatial dimensions of the feature maps, thereby reducing computational complexity and enhancing model efficiency. Following the pooling layer, there is a dropout layer with a dropout ratio of 0.2. This means that in each iteration, the model randomly deactivates 20% of the neurons, which aids in preventing overfitting of the training data.

The pooled feature representations are then passed to a flatten layer, which transforms the multidimensional feature maps into one-dimensional feature vectors, preparing them for subsequent processing by the dense layers. Subsequent to the flattening operation, the feature vectors are fed into a dense layer that consists of 32 neurons, and the ReLU activation function is applied. To improve the model's robustness and reduce the risk of overfitting, a dropout layer with a dropout rate of 0.4 is introduced.

Subsequently, the data passes through another fully connected layer consisting of 2 neurons and utilizes a softmax activation function. The softmax function is a commonly used activation function in classification tasks, which converts the outputs of neurons into predictive probabilities. The hyperparameters of the 2D-CNN model are detailed in Table 1.

Table 1. 2D-CNN model hyperparameters

Parameter	Value
Optimizer	Adam
Convolution layer activation function	ReLU
Classification layer activation function	Softmax
Early stopping monitor	Val_loss
Learning rate	0.001
Convolution layer filters	16
Convolution layer kernel size	3×3
Pooling layer filter size	2×2

The model is trained using the Adam optimizer and cross-entropy loss function. To prevent overfitting, we use an early stopping technique to monitor loss values on the validation set, allowing us to halt training prematurely if necessary. We set the model to show no progress on the validation set for 5 consecutive training epochs, then training ends early. Each epoch processes 32 samples.

6 Experimentation and Analysis

In this section, we perform color image transformation and generation experiments on the AWID3 dataset, and evaluate the performance of our proposed TV-ADS attack detection method. Additionally, we analyze the impact of dataset size on detection performance, as well as the required model training and testing time at optimal performance.

6.1 Dataset and Experimental Environment

The AWID3 dataset represents a significant expansion and enhancement of the previous AWID2 corpus. It captures and investigates traces of various attacks conducted within the IEEE 802.1X Extensible Authentication Protocol (EAP) environment, with a particular focus on analyzing attack activities within the IEEE 802.11 wireless network setting. Building upon the foundation of AWID2, this dataset offers more detailed and comprehensive network traffic data by further extending and enriching its contents.

The data collection process for the AWID3 dataset involves simulating a variety of attacks within wireless network environments, each with different security levels. These attacks encompass a variety of types, including malicious packet injection, identity spoofing, and denial of service. We chose seven attacks from the AWID3 dataset because they are specific to wireless networks. The seven attacks are Deauthentication, Disassociation, Rogue AP, Reassociation, Krack, Kr00k, and Evil Twin. We approximate the ratio of the number of samples of the normal traffic data to the attack traffic data used to be 1:1 to avoid data imbalance. In the total dataset, 20% is allocated as the test set, with the remaining 80% of the data being designated for training purposes. During the training process, 20% of the training data is further partitioned to serve as a validation set. Table 2 provides the number of normal and attack frames in each attack file and the number used in our experiments. Figure 3 shows a series of color images that exemplify each type of normal and attack data processed through the two scaling modes: cropping and padding and bilinear interpolation.

The construction process of the AWID3 dataset meticulously replicates a real wireless network environment, capturing the actual occurrences of various attacks. This approach enables us to rigorously test our attack detection methodology in authentic scenarios, thereby enhancing our ability to assess its performance and feasibility.

Our experiment is conducted on a Windows 10 system, with a CPU of 13th Gen Intel® Core™ i9-13900K running at 3.00 GHz, and equipped with 128GB of RAM and 2TB of storage capacity. The codebase for the experiment is developed in Python 3.8 and executed within the PyCharm 2023 Integrated Development Environment.

Table 2. AWID3 attack types

File name (PCAP)	Normal frames	Attack frames	Total frames
Deauth	1,587,527	38,945	1,626,472
Disass	1,938,588	75,131	2,013,719
(Re)Assoc	1,838,436	5,503	1,843,939
Rogue_AP	1,971,883	1,310	1,973,193
Krack	1,388,503	49,990	1,438,493
Kr00k	2,708,655	191,803	2,900,458
Evil_Twin	3,676,669	102,059	3,778,728
Total	15,110,261	464,741	15,575,002
Used	70,000	71,813	141,813
Training data	56,000	57,451	113,451
Testing data	14,000	14,362	28,362

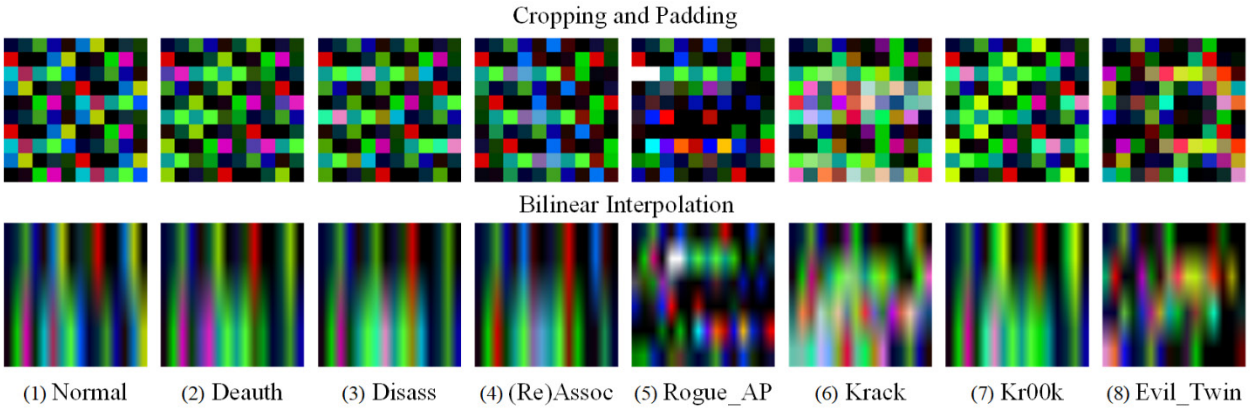


Figure 3. Several color images of each type of normal and attack data for both scaling modes

6.2 Comparative Analysis

We compare two methods that also use the AWID3 dataset for attack detection. We use four metrics, accuracy, precision, recall and F1 score, to evaluate and compare the performance of these methods.

Accuracy is the most intuitive performance evaluation metric, and it is also a basic metric for evaluating the classification performance of a model, quantifying the proportion of correctly classified samples out of the total number of samples. Accuracy is often used to evaluate the overall classification performance.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}. \quad (1)$$

Precision is a metric that measures the percentage of correctly predicted positive samples out of all samples that were predicted as positives by the model. It serves as a measure of the model's accuracy in identifying positive categories. High precision means fewer false positive results.

$$\text{Precision} = \frac{TP}{TP + FP}. \quad (2)$$

Recall is the proportion of all samples that the model correctly predicts as positive categories out of all samples that are actually positive categories. It measures the model's ability to recognize positive categories.

$$\text{Recall} = \frac{TP}{TP + FN}. \quad (3)$$

The F1 score is a harmonic mean that combines precision and recall, providing a comprehensive evaluation of a model's performance. It considers the model's trade-off between precision and recall and is particularly useful for assessing models in unbalanced classification problems.

$$F_1 = 2 \cdot \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \quad (4)$$

where, True Positives (TP): the number of positive category samples that the model correctly classifies as Positive.

False Positives (FP): the number of negative category samples that the model incorrectly classifies as positive.

True Negatives (TN): the number of negative category samples that the model correctly classifies as negative.

False Negatives (FN): the number of positive category samples that the model incorrectly classifies as negative.

As mentioned in Section 4.2, we use two methods to scale the generated images. The first method involves applying Cropping and Padding (C&P) as a scaling process, and subsequently training and testing the model after this pre-processing step, referred to as TV-ADS (C&P). The second method involves using Bilinear Interpolation (BI) as the scaling technique, and then training and testing the model after this pre-processing step, designated as TV-ADS (BI). Figure 4(a) shows the change in loss when the model is trained on the dataset processed using cropping and padding, and Figure 4(b) shows the change in loss when the model is trained on the dataset processed using bilinear interpolation. Figure 5(a) shows the confusion matrix for the TV-ADS (C&P) model on its test set. Figure 5(b) shows the confusion matrix for the TV-ADS (BI) model on its test set.

As can be seen from the detailed comparative analysis, our model performs well on all four metrics. It is worth noting that accuracy is an important metric for assessing the classification ability of a model. As shown in Table 3, our TV-ADS (C&P) method possesses an impressive accuracy rate of 99.90%. For practical applications, minimizing false alarms is essential, therefore, we emphasize precision. In this regard, TV-ADS (C&P) also performs well, with a precision rate of 99.88%. To provide a holistic performance assessment, we incorporate the F1 score, a combined metric of precision and recall. TV-ADS (C&P) achieves a 99.90% F1 score and 99.93% recall. Structurally, our model is elegantly simple. Our 2D-CNN model has just one convolutional layer, significantly trimming its complexity and computational demand. A visual comparison of various methods can be seen in the line graphs presented in Figure 6.

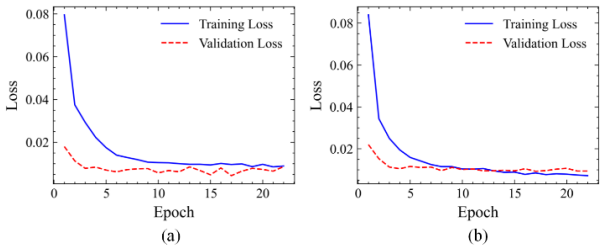


Figure 4. Plot of change in loss function

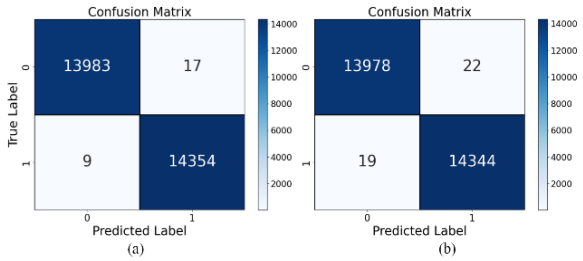


Figure 5. Confusion matrix on test data

Table 3. Comparison of model performance on AWID3 dataset, TV-ADS (C&P) denotes images scaled with cropping and padding for model training, TV-ADS (BI) denotes images scaled with bilinear interpolation for model training

Method	Accuracy	Precision	Recall	F1 score
1D-CNN-Binary [43]	94.60%	94.60%	95.10%	94.60%
A3C [44]	98.68%	98.40%	98.90%	98.68%
TV-ADS-(C&P)	99.90%	99.88%	99.93%	99.90%
TV-ADS-(BI)	99.85%	99.84%	99.86%	99.85%

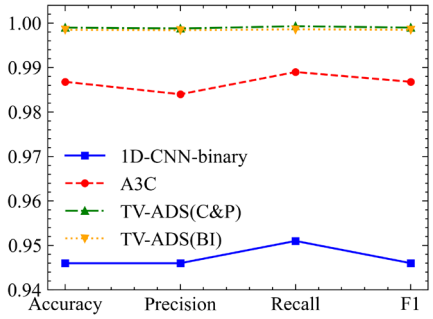


Figure 6. Performance comparison line chart

6.3 Incremental Analysis

To evaluate the model's performance over different sizes of datasets, we conduct an incremental analysis. Our objective is to investigate whether increasing the volume of training data could enhance the model's performance and to understand the precise influence of incremental data on its performance. Specifically, we test the performance of TV-ADS (C&P) and TV-ADS (BI) models on different datasets. In this case, the ratio of attack samples to normal samples is 1:1. We evaluate the model's performance using accuracy, precision, recall, and F1 score.

This experiment examines the performance of the model on the dataset processed by two scaling methods. The performance metrics include accuracy, precision, recall, and F1 score. The table displays datasets that include the training set, validation set, and test set, and they are divided in the same way as presented in Section 6.1. As shown in Table 4, The overall dataset size escalates from 1008 samples to 141813 samples.

From the results of TV-ADS (C&P), all the performance metrics show a positive growth trend as the dataset size increases. Accuracy increases from 98.25% to 99.90%, precision from 97.54% to 99.88%, recall from 99.00% to 99.93%, and F1 score from 98.27% to 99.90%. Particularly noteworthy is that all metrics are close to or exceed 99% by the time the dataset size reaches 8008 samples. The model's performance gains become slower when the dataset is larger. When the dataset size exceeds 100000, the model performance has reached its optimum, and thereafter the model performance does not improve anymore as the dataset size increases.

For TV-ADS (BI), its overall performance is not as good as TV-ADS (C&P), similarly, the performance metrics show the same improvement as the dataset increases. Its accuracy increased from 93.56% to 99.85%, precision from 96.80% to 99.84%, recall from 90.09% to 99.86%, and F1 score from 93.33% to 99.85%. Compared to TV-ADS (C&P), the performance of TV-ADS (BI) is lower for small datasets but improves significantly as the dataset increases. Similarly, the model performance has been optimized when the dataset size reaches 100000.

In general, regardless of the image scaling method used, our model exhibits excellent performance beyond traditional methods, reaffirming the effectiveness and potential of visualizing traffic data for attack detection.

Table 4. Performance metrics for incremental analysis

	TV-ADS (C&P)				TV-ADS (BI)			
Metrics / Data size	Accuracy	Precision	Recall	F1 score	Accuracy	Precision	Recall	F1 score
1,008	98.25%	97.54%	99.00%	98.27%	93.56%	96.80%	90.09%	93.33%
2,002	98.75%	98.03%	99.50%	98.76%	97.25%	95.67%	99.00%	97.31%
4,004	98.87%	99.24%	98.50%	98.87%	97.62%	98.47%	96.75%	97.61%
8,008	99.25%	99.74%	98.75%	99.24%	98.25%	98.01%	98.50%	98.25%
10,010	99.40%	99.40%	99.40%	99.40%	99.20%	99.59%	98.80%	99.19%
29,998	99.51%	99.76%	99.26%	99.51%	99.70%	99.89%	99.50%	99.69%
71,314	99.85%	99.86%	99.86%	99.86%	99.73%	99.77%	99.69%	99.73%
100,000	99.92%	99.87%	99.97%	99.92%	99.87%	99.87%	99.88%	99.87%
120,002	99.87%	99.80%	99.95%	99.87%	99.85%	99.82%	99.88%	99.85%
141,813	99.90%	99.88%	99.93%	99.90%	99.85%	99.84%	99.86%	99.85%

6.4 Evaluation of Training Time and Testing Time

In addition to the accuracy and robustness of the model, time efficiency is also a key evaluation criterion. Especially in real-time or near real-time applications, the training and testing speed of a model directly determines its usability, we measured the training and testing time of the model to verify whether its time loss is acceptable in real scenarios. To ensure the best performance in this test section, As shown in Table 5, we use 141813 total data, of which the test set accounts for 20%, i.e., 28362 color images.

Table 5. Training time and testing time

Training stage	Testing stage
2946.93 seconds	10.86 seconds

During the training process on the dataset processed with the Cropping and Padding method, the model take 2946.93 seconds. This duration isn't arbitrary and it is influenced by three primary factors. Firstly, the size of the dataset plays a role, with larger datasets typically demanding longer training periods. Secondly, the model's structural complexity is pivotal, more intricate models tend to consume more computational time. Lastly, the efficiency of the computational hardware also substantially impacts the training speed. In the testing phase, we evaluate the model on 28362 samples, taking 10.86 seconds. Despite the extended training duration, the model predicts swiftly. For scenarios not demanding real-time responses, the model's time overhead remains within acceptable bounds.

7 Conclusion

In this paper, we introduce a novel traffic-image mapping algorithm, which splits the original wireless network traffic records into event cells and transforms these event cells into standard images with uniform size. This can effectively preserve the integrity and centrality of the traffic's original correlation features and can overcome the problem of poor detection performance caused by feature segmentation in other existing mapping methods. Then, we propose a new attack detection scheme TV-ADS based on our binary classification model with a 2D-CNN structure. Although TV-

ADS outperforms the existing schemes in terms of accuracy, precision, recall, and F1-score in our experiments, it can only perform post-event detection and cannot detect attacks in real-time. Next, following the main idea in this paper, we will consider designing and constructing one incremental traffic-image mapping method, and utilize techniques such as subgraph matching to detect malicious network traffic as early as possible.

Acknowledgement

This work was supported by the National Key R&D Program of China (Grant No. 2023YFB3107500), the Chinese Postdoctoral Science Foundation (2021MD703967), the Major Research plan of the National Natural Science Foundation of China (Grant No. 92267204), the Key R&D Program of Shaanxi Province (2021ZDLGY03-10), Shandong Provincial Natural Science Foundation (ZR2021LZH006).

References

[1] Cisco public, *Cisco Annual Internet Report (2018–2023) White Paper*, Report No. CIR-181831-04, March, 2020.

[2] T. Karygiannis, L. Owens, *Wireless Network Security: 802.11, Bluetooth and Handheld Devices*, National Institute of Standards and Technology, Special Publication (NIST SP) - 800-48, November, 2002.

[3] R. E. Navas, F. Cuppens, N. B. Cuppens, L. Toutain, G. Z. Papadopoulos, Physical Resilience to Insider Attacks in IoT Networks: Independent Cryptographically Secure Sequences for DSSS Anti-Jamming, *Computer Networks*, Vol. 187, Article No. 107751, March, 2021.

[4] P. Mishra, V. Varadharajan, U. Tupakula, E. S. Pilli, A Detailed Investigation and Analysis of Using Machine Learning Techniques for Intrusion Detection, *IEEE Communications Surveys & Tutorials*, Vol. 21, No. 1, pp. 686-728, Firstquarter, 2019.

[5] A. L. Buczak, E. Guven, A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection, *IEEE Communications Surveys & Tutorials*, Vol. 18, No. 2, pp. 1153-1176, Secondquarter, 2016.

[6] C. H. Tseng, Y. T. Chang, EBDM: Ensemble Binary Detection Models for Multi-Class Wireless Intrusion Detection Based on Deep Neural Network, *Computers & Security*, Vol. 133, Article No. 103419, October, 2023.

[7] Y. R. Wu, D. B. Wei, J. Feng, Network Attacks Detection Methods Based on Deep Learning Techniques: A Survey, *Security and Communication Networks*, Vol. 2020, pp. 1-17, August, 2020.

[8] M. Y. Qi, M. Liu, Y. M. Fu, Research on Network Intrusion Detection Using Support Vector Machines Based on Principal Component Analysis, *Netinfo Security*, Vol. 15, No. 2, pp. 15-18, February, 2015.

[9] A. Hadri, K. Chougdali, R. Touahni, Intrusion Detection System Using PCA and Fuzzy PCA Techniques, *International Conference on Advanced Communication Systems and Information Security (ACOSIS)*, Marrakesh, Morocco, 2016, pp. 1-7.

[10] Y. LeCun, Y. Bengio, G. Hinton, Deep Learning, *Nature*, Vol. 521, No. 7553, pp. 436-444, May, 2015.

[11] B. Abolhasanzadeh, Nonlinear Dimensionality Reduction for Intrusion Detection Using Auto-Encoder Bottleneck Features, *Conference on Information and Knowledge Technology (IKT)*, Urmia, Iran, 2015, pp. 1-5.

[12] M. A. Khan, M. R. Karim, Y. Kim, A Scalable and Hybrid Intrusion Detection System Based on the Convolutional-LSTM Network, *Symmetry*, Vol. 11, No. 4, Article No. 583, April, 2019.

[13] Z. Wang, T. Oates, Encoding Time Series as Images for Visual Inspection and Classification Using Tiled Convolutional Neural Networks, *Workshops at the twenty-ninth AAAI conference on artificial intelligence*, Austin, Texas, USA, 2015, pp. 1-7.

[14] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. Denton, S. K. S. Ghasemipour, B. K. Ayan, S. S. Mahdavi, R. Gontijo-Lopes, T. Salimans, J. Ho, D. J. Fleet, M. Norouzi, Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding, *2022 36th Conference on Neural Information Processing Systems (NeurIPS)*, New Orleans, Louisiana, USA, 2022, pp. 36479-36494.

[15] W. Wang, M. Zhu, X. Zeng, X. Ye, Y. Sheng, Malware Traffic Classification Using Convolutional Neural Network for Representation Learning, *2017 International Conference on Information Networking (ICOIN)*, Da Nang, Vietnam, 2017, pp. 712-717.

[16] E. Chatzoglou, G. Kambourakis, C. Kolias, Empirical Evaluation of Attacks Against IEEE 802.11 Enterprise Networks: The AWID3 Dataset, *IEEE Access*, Vol. 9, pp. 34188-34205, February, 2021.

[17] B. Riyaz, S. Ganapathy, A Deep Learning Approach for Effective Intrusion Detection in Wireless Networks Using CNN, *Soft Computing*, Vol. 24, No. 22, pp. 17265-17278, November, 2020.

[18] A. Ahmim, L. Maglaras, M. A. Ferrag, M. Derdour, H. Janicke, A Novel Hierarchical Intrusion Detection System Based on Decision Tree and Rules-Based Models, *International Conference on Distributed Computing in Sensor Systems (DCOSS)*, Santorini, Greece, 2019, pp. 228-233.

[19] Y. Wang, A Multinomial Logistic Regression Modeling Approach for Anomaly Intrusion detection, *Computers & Security*, Vol. 24, No. 8, pp. 662-674, November, 2005.

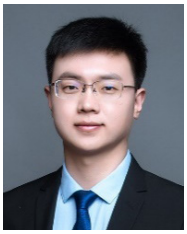
[20] F. Jiang, C. P. Wang, H. F. Zeng, Relative Decision Entropy Based Decision Tree Algorithm and Its Application in Intrusion Detection, *Computer Science*, Vol. 39, No. 4, pp. 223-226, April, 2012.

[21] R. Chowdhury, S. Sen, A. Roy, B. Saha, An Optimal Feature Based Network Intrusion Detection System Using Bagging Ensemble Method for Real-Time Traffic Analysis, *Multimedia Tools and Applications*, Vol. 81, No. 28, pp. 41225-41247, November, 2022.

[22] H. Yang, F. Wang, Wireless Network Intrusion Detection Based on Improved Convolutional Neural Network, *IEEE Access*, Vol. 7, pp. 64366-64374, May, 2019.

[23] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-Based Learning Applied to Document Recognition, *Proceedings of the IEEE*, Vol. 86, No. 11, pp. 2278-2324, November, 1998.
[24] G. E. Hinton, R. R. Salakhutdinov, Reducing the Dimensionality of Data with Neural Networks, *Science*, Vol. 313, No. 5786, pp. 504-507, July, 2006.
[25] J. Hall, M. Barbeau, E. Kranakis, Anomaly-Based Intrusion Detection Using Mobility Profiles of Public Transportation Users, *IEEE International Conference on Wireless And Mobile Computing, Networking And Communications*, Montreal, Que, 2005, pp. 17-24.
[26] I. Butun, I. H. Ra, R. Sankar, An Intrusion Detection System Based on Multi-Level Clustering for Hierarchical Wireless Sensor Networks, *Sensors*, Vol. 15, No. 11, pp. 28960-28978, November, 2015.
[27] X. D. Hoang, J. Hu, P. Bertok, A Program-Based Anomaly Intrusion Detection Scheme Using Multiple Detection Engines and Fuzzy Inference, *Journal of Network and Computer Applications*, Vol. 32, No. 6, pp. 1219-1228, November, 2009.
[28] D. E. Denning, P. G. Neumann, *Requirements and Model for IDES – A Real-Time Intrusion Detection System*, Computer Science Laboratory SRI International, 1985.
[29] N. Ye, S. M. Emran, Q. Chen, S. Vilbert, Multivariate Statistical Analysis of Audit Trails for Host-Based Intrusion Detection, *IEEE Transactions on Computers*, Vol. 51, No. 7, pp. 810-820, July, 2002.
[30] P. García-Teodoro, J. Díaz-Verdejo, G. Maciá-Fernández, E. Vázquez, Anomaly-Based Network Intrusion Detection: Techniques, Systems and Challenges, *Computers & Security*, Vol. 28, No. 1-2, pp. 18-28, February-March, 2009.
[31] C. M. K. Ho, K. C. Yow, Z. Zhu, S. Aravamuthan, Network Intrusion Detection via Flow-To-Image Conversion and Vision Transformer Classification, *IEEE Access*, Vol. 10, pp. 97780-97793, August, 2022.
[32] S. Golubev, E. Novikova, Image-based Intrusion detection in Network Traffic, in: L. Braubach, K. Jander, C. Bădică (Eds.), *International Symposium on Intelligent and Distributed Computing*, Springer, Cham, 2022, pp. 51-60.
[33] A. Sharma, E. Vans, D. Shigemizu, K. A. Boroevich, T. Tsunoda, Deepinsight: A Methodology to Transform a Non-Image Data to An Image for Convolution Neural Network Architecture, *Scientific reports*, Vol. 9, No. 1, Article No. 11399, August, 2019.
[34] K. Malialis, *Distributed Reinforcement Learning for Network Intrusion Response*, Ph. D. Thesis, University of York, York, UK, 2014.
[35] Q. Liu, J. W. Zhai, Z. Z. Zhang, S. Zhong, Q. Zhou, P. Zhang, J. Xu, A Survey on Deep Reinforcement Learning, *Chinese Journal of Computers*, Vol. 41, No. 1, pp. 1-27, January, 2018.
[36] G. Caminero, M. Lopez-Martin, B. Carro, Adversarial Environment Reinforcement Learning Algorithm for Intrusion Detection, *Computer Networks*, Vol. 159, pp. 96-109, August, 2019.
[37] C. Rong, G. Gou, M. Cui, G. Xiong, Z. Li, L. Guo, Transnet: Unseen Malware Variants Detection Using Deep Transfer Learning, *Security and Privacy in Communication Networks: 16th EAI International Conference*, Washington, DC, USA, 2020, pp. 84-101.
[38] K. He, X. Zhang, S. Ren, J. Sun, Deep Residual Learning for Image Recognition, *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2016, pp. 770-778.
[39] R. S. Ahmad, A. H. Ali, S. M. Kazim, Q. Niyaz, A GAF and CNN based Wi-Fi Network Intrusion Detection System, *IEEE INFOCOM 2023 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Hoboken, NJ, USA, 2023, pp. 1-6.
[40] G. Feng, B. Li, M. Yang, Z. Yan, V-CNN: Data Visualizing based Convolutional Neural Network, *2018 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC)*, Qingdao, China, 2018, pp. 1-6.
[41] S. Z. Lin, Y. Shi, Z. Xue, Character-Level Intrusion Detection Based on Convolutional Neural Networks, *2018 International Joint Conference on Neural Networks (IJCNN)*, Rio de Janeiro, Brazil, 2018, pp. 1-8.
[42] B. Yan, G. Han, LA-GRU: Building Combined Intrusion Detection Model Based on Imbalanced Learning and Gated Recurrent Unit Neural Network, *Security and Communication Networks*, Vol. 2018, Article No. 6026878, August, 2018.
[43] M. Natkaniec, M. Bednarz, Wireless Local Area Networks Threat Detection Using 1D-CNN, *Sensors*, Vol. 23, No. 12, Article No. 5507, June, 2023.
[44] E. Muhati, D. B. Rawat, Asynchronous Advantage Actor-Critic (A3C) Learning for Cognitive Network Security, *2021 Third IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*, Atlanta, GA, USA, 2021, pp. 106-113.

Biographies



Zhiwei Zhang received the Ph.D. degree in cryptography from Xidian University, Xi'an, Shaanxi, China, in 2019. His research interest includes authentication, access control, data storage security in cloud computing and endogenous security in intelligent swarms.

Guiyuan Tang is currently working toward the M.S. degree in Computer Science and Technology with the School of Computer Science and Technology, Xidian University, China. He received the B.S. degree in Computer Science and Technology from Fuzhou University, China. His research interests include authentication and attack detection.

Baoquan Ren is a senior engineer at the Academy of Systems Engineering of the Academy of Military Sciences. He received Ph.D. in military science. He is mainly engaged in the research of communication network technology.

Hongjun Li received the Ph.D. in engineering from PLA University of Science and Technology in 2017. He is now an engineer at the Academy of System Engineering of the Academy of Military Sciences. He is mainly engaged in the research of intelligent network, satellite communication, and IoTs.

Yulong Shen received the Ph.D. degree in cryptography from Xidian University, Xi'an, China, in 2008. He is currently a Professor with the School of Computer Science and Technology, Xidian University. His research interests include wireless network security and cloud computing security.