IEICE TRANS. INF. & SYST., VOL.E106–D, NO.9 SEPTEMBER 2023

PAPER Special Section on Log Data Usage Technology and Office Information Systems

Few-Shot Learning-Based Malicious IoT Traffic Detection with Prototypical Graph Neural Networks

Thin Tharaphe THEIN[†a)], ***Nonmember***, **Yoshiaki SHIRAISHI**[†], ***Senior Member***, ***and*** **Masakatu MORII**[†], ***Fellow***

SUMMARY With a rapidly escalating number of sophisticated cyber-attacks, protecting Internet of Things (IoT) networks against unauthorized activity is a major concern. The detection of malicious attack traffic is thus crucial for IoT security to prevent unwanted traffic. However, existing traditional malicious traffic detection systems which relied on supervised machine learning approach need a considerable number of benign and malware traffic samples to train the machine learning models. Moreover, in the cases of zero-day attacks, only a few labeled traffic samples are accessible for analysis. To deal with this, we propose a few-shot malicious IoT traffic detection system with a prototypical graph neural network. The proposed approach does not require prior knowledge of network payload binaries or network traffic signatures. The model is trained on labeled traffic data and tested to evaluate its ability to detect new types of attacks when only a few labeled traffic samples are available. The proposed detection system first categorizes the network traffic as a bidirectional flow and visualizes the binary traffic flow as a color image. A neural network is then applied to the visualized traffic to extract important features. After that, using the proposed few-shot graph neural network approach, the model is trained on different few-shot tasks to generalize it to new unseen attacks. The proposed model is evaluated on a network traffic dataset consisting of benign traffic and traffic corresponding to six types of attacks. The results revealed that our proposed model achieved an F1 score of 0.91 and 0.94 in 5-shot and 10-shot classification, respectively, and outperformed the baseline models.

key words: *few-shot learning, graph neural networks, internet of things, machine learning, network anomaly detection*

1. Introduction

The Internet of Things (IoT) has become one of the fastest-growing technologies in recent years. With an increase in extensive use of IoT devices in fields such as healthcare, smart homes, smart cities, manufacturing and automotive industry, the importance of Internet-connected devices in daily life has increased. In addition, with the increasing number of cyber-attacks, it is critical to ensure that IoT devices are protected from potential cyber threats. However, most IoT devices are known for their vulnerabilities and lack of on-device security controls to defend against cyber-attacks [1]–[3]. The reason for this is that most IoT devices have resource constraints and insufficient computing power, which allows only limited functions to be executed [3]. Attackers exploit loopholes in IoT devices to perform malicious activities, such as using IoT devices for botnet attacks. One example is the Mirai [4] botnet attack that occurred in 2016 and caused massive Internet security breaches by exploiting insecure IoT devices.

To protect IoT networks from cyber-attacks, intrusion detection systems [5]–[7] have been used extensively to monitor unauthorized activities and identify notorious attack network traffic in the IoT ecosystem. Depending on the technique employed, intrusion detection systems can be mainly sorted into two: signature-based and anomaly-based systems. The former systems compare incoming traffic to predefined attack signatures in a database through pattern matching. This technique is effective and highly accurate for previously seen (known) attacks, but is ineffective in zero-day attacks detection since the new attack signatures are not present in the database. Anomaly-based intrusion detection systems can overcome this limitation using artificial intelligence, by defining any behavior that differs from the observed behavior as an anomaly. With recent advancements in machine learning (ML), many studies have applied ML algorithms for the detection of malicious attacks in the IoT network monitoring system (e.g., [5]–[7]). Providing that there is a considerable number of labeled samples for training the ML models, the detection systems can distinguish normal behavior from abnormal behavior with relatively high accuracy.

However, there are several problems in ML-based IoT intrusion detection systems that must be addressed. Namely, the amount of benign traffic is huge compared to abnormal malicious traffic, and some IoT malware attack types have significantly fewer samples, resulting in an unbalanced dataset [8]. Most importantly, new IoT malware threats are constantly emerging; consequently, ML-based detection systems have a new challenge known as zero-day attacks [9], which are types of attacks that did not exist at the time of model training. It may take time for security vendors and researchers to publish a huge number of samples of the new threats; thus, there is a need for a system that can detect IoT network traffic by using a limited number of new malicious samples. This can be achieved by a few-shot learning method. Few-shot learning is a type of ML method that aims to make predictions with only a few labeled data in a supervised setting. Several previous studies [10]–[13] on few-shot learning used the principle of meta-learning, where a certain number of correlated tasks are learned during training. In the meta-testing stage, the trained learner can be utilized to predict unseen but related tasks with only a few labeled samples.

To defend against potential cyber-attacks in the IoT ecosystem, IoT traffic detection and identification are

Manuscript received November 22, 2022.
Manuscript revised March 23, 2023.
Manuscript publicized June 22, 2023.

[†]The authors are with Kobe University, Kobe-shi, 657–8501 Japan.
a) E-mail: thein.thin@gsuite.kobe-u.ac.jp

DOI: 10.1587/transinf.2022OFP0004

crucial for IoT traffic intrusion management and IoT security. In this work, we propose a detection scheme based on the binary visualization of network traffic and on few-shot learning to detect unknown malware attacks on IoT networks. The rationale for representing network traffic as an image is that a malware traffic image exhibits significantly more clustered patterns than a benign traffic image, which is more consistent and static [14]. Moreover, representing the captured IoT traffic as an image can lead to a clearer comprehension of the network traffic since the same type of malware attack generates similar image patterns [15] that can be recognized by a deep learning model, overcoming the requirements of manual feature engineering and prior domain knowledge. [16], [17] claimed that deep convolutional neural network model performs better on image dataset and transformed the network traffic dataset into three-dimensional image and showed that the image-based method is superior to conventional machine learning. Therefore, we propose a method that can identify malicious IoT network traffic based on a traffic visual representation and a few-shot learning-based prototypical graph neural network. Our main contributions to this work are as follows:

1) A few-shot learning-based IoT network traffic detection system using binary visualization and a graph neural network is proposed. It can be applied to detect and identify new malware attack traffic using only a few labeled samples.
2) The network traffic is preprocessed as a red–green–blue (RGB) image, and a malware traffic image dataset is constructed from the generated images to evaluate the proposed few-shot learning scheme with a prototypical graph neural network model. A pretrained convolutional neural network (CNN) is employed to extract the image features.
3) The experimental results indicate that our proposed model can detect new IoT network attacks with a few labeled samples and is applicable to a diverse range of malware attacks.

The remainder of this paper is arranged as follows. Section 2 discuss the previous works on IoT network traffic detection, while our proposed method is explained in Sect. 3. Meanwhile, Sect. 4 describes the dataset, evaluation of the proposed method, and experiment results, while Sect. 5 summarized our proposed method.

2. Related Work

This section reviews previous studies related to IoT network traffic identification and intrusion detection systems. In addition, background knowledge of few-shot learning and graph neural networks is provided.

2.1 IoT Network Traffic Detection Techniques

With the growing number and accelerated use of IoT devices, the vulnerabilities of these devices have become a target for cybercriminals, contributing to a surge in cyber-attacks and information leakage. As discussed below, extensive efforts have been made by the research community to tackle security and privacy concerns in IoT networks. In previous studies [3], [5], [6], ML was the most commonly used approach to distinguish malicious traffic intrusion in IoT networks. The intelligent integrated intrusion detection system proposed by [5] used a deep learning algorithm to discover malicious attacks in real IoT network traffic. After sorting the incoming traffic into sessions, features such as the source and destination IP address, transmission mode, duration, transmission and reception rate, and transmission-to-reception ratio were extracted and forwarded to a deep neural network. An average precision of 95% and recall of 97% were achieved for five attack scenarios: blackhole, sinkhole, wormhole, distributed denial-of-service (DDoS), and opportunistic service attacks.

In [6], the authors addressed cyber threats in a smart city infrastructure by proposing the random forest classifier-based anomaly detection system in fog nodes. The authors claimed that the proposed model could effectively detect compromised IoT devices. The classification results on the UNSW-NB15 dataset indicated that the model predicted the normal class with an F1 score of 0.99 and attack traffic with an F1 score of 0.86. Likewise, the authors of [7] proposed a fog-computing-based intrusion detection system by utilizing a multilayer recurrent neural network model. The model was experimented on the NSL-KDD dataset and detected denial-of-service (DoS) attacks with high sensitivity; however, the detection rate was much lower for other types of attacks. The authors of [18] reported that misclassification in ML algorithms occurs due to inappropriate feature selection. Therefore, they proposed a novel metric-based feature selection approach for malicious IoT traffic detection. Their model selected only efficient and helpful features for the ML algorithm and achieved the detection accuracy of more than 96% on the Bot-IoT dataset.

Some studies [19], [20] also converted network traffic into an image based on the network packets, flow, and session and applied a CNN model to the produced images to identify malicious traffic. In [19], the authors proposed a malicious IoT traffic classification technique that represents network traffic as an image and utilizes ResNet50 to analyze the visualized data. The approach was evaluated on a dataset of 1,000 PCAP files of benign and malicious traffic and demonstrated promising results. Similar to [19], the approach proposed in [20] transformed traffic data packet and flow information into images and classified the images using the residual neural network (ResNet) model. Multiclass classification on the CICDDoS2019 dataset achieved an F1 score of 0.86.

Network intrusion detection systems based on the few-shot learning paradigm have also been proposed in recent studies [21], [22]. In these systems, a trained meta-learner can detect new types of attacks using only a few labeled data. A detailed explanation of the meta-learning-based few-shot framework is provided in Sect. II-B. The approach proposed in [21] consists of two main parts:

deep-neural-network-based feature extraction and feature comparison. The feature extraction part produces feature map pairs for network traffic sample pairs, while the comparison part calculates the delta score, which indicates whether a pair of traffic samples belong to the same class. A few-shot dataset was constructed from the ISCX2012 and CICIDS2017 datasets, and malicious traffic could be detected up to 99% on average. In [22], the authors proposed an approach that performs feature embedding of traffic samples with a trained embedding function and calculates the cosine distance between the samples to predict which traffic samples are closer in the embedding space. The experimental results indicate that the choice of the embedding and distance functions affects the accuracy of the classification model.

2.2 Few-Shot Learning

In general, the objective of few-shot learning is to recognize new unobserved tasks using a small quantity of labeled training sample. Various few-shot learning approaches utilize the meta-learning framework with episodic training [10]–[13]. In these approaches, diverse few-shot tasks are sampled from the meta-train dataset for each training episode during the model training. Each few-shot task is made up of a known support set and unknown query set. Instead of determining which class each sample belongs to, the few-shot meta-learner learns the similarity or dissimilarity between the support and query sets to assign label information from a support instance to a query instance. The authors of [10]–[12] presented few-shot learning paradigms that compare a labeled support set and unknown query samples in a shared embedding space to predict the label information for query instances. Matching Networks [10] utilizes the weighted nearest neighbor search with the use of the attention mechanism, while Relation Networks [11] trains the model to learn the nonlinear relation between the support and query sets. Prototypical Networks [12] calculates the prototype representation of each class in the support set in the embedding space, and the label of the query sample is predicted by computing the Euclidean distance. The few-shot network traffic detection method proposed in this paper follows the principle of Prototypical Networks.

2.3 Graph Neural Networks

Graph neural networks (GNNs) are a deep learning architecture designed to analyze graph-structured data [23]. A graph consists of a set of vertices (nodes) joined by edges. A GNN learns the current node representation by repeatedly combining the features of neighboring nodes using the message passing algorithm, thereby generating similar representations for strongly linked nodes. Due to the advantages of GNNs, some approaches [13], [24], [25] incorporated GNNs in few-shot learning domain and demonstrated promising results. The authors of [13] used a GNN as a label propagation module to forecast the label of unlabeled nodes. In addition, the edge-labeling GNN (EGNN) proposed in [24] predicted the edge labels between the support and query sets by iteratively updating the node and edge features. Furthermore, the fuzzy GNN (FGNN) [25] employed the fuzzy membership function to iteratively update the edge labels; then, node classification was performed on the constructed graph to predict the unlabeled nodes.

3. Proposed Method

3.1 Few-Shot Learning Strategy

The proposed few-shot learning approach follows the approach of Prototypical Networks described in [12] by using the meta-learning framework with episodic training. Few-shot learning is also known as M-way K-shot classification. M-way denotes how many classes are in each task T, while K-shot signifies that each class has K samples. Like traditional supervised learning with a training and test dataset, few-shot learning has a meta-training set M_{Train} and meta-test set M_{Test}. Rather than training on the entire training dataset at once, the meta-learner is trained to learn over diverse few-shot tasks T in multiple episodes. For every task T, it is made up of support set S and query set Q, and the samples in both sets are of the same class. The support set is labeled, while the query set is unlabeled and needs to be predicted. Therefore, for every episode at the meta-training stage, we sample the M-way K-shot task T from dataset M_{Train} as

$$S_T = \{(x_j, y_j) \,|\, y_j \in C,\ j = 1, \ldots, K \times N_S\},$$
$$Q_T = \{(x_k, y_k) \,|\, y_k \in C,\ k = 1, \ldots, K \times N_Q\},\ \text{and}$$
$$S_T \cap Q_T = \emptyset.$$

The symbol C denotes the set of train classes and N_S, N_Q represent the number of samples for each class in the support set and query set, respectively. During the training phase, the meta-training task $T_{train} = (S_T, Q_T)_{T=1}^{i}$ is trained by i number of episodes with known label information for both the support and query sets. The trained model is then used to predict the label of the query sample of M_{Test}, with a few labeled support samples. The M-way K-shot task T for the meta-test dataset is prepared in the same way as the meta-training task. Ideally, the classes used in the meta-training and meta-testing phases are completely different, which achieves the goal of predicting zero-day IoT network traffic attacks with limited labeled data.

3.2 Data Preprocessing

Because the proposed model represents IoT network traffic as an image, some data preprocessing must be carried out on the captured IoT network traffic. The data preprocessing step is illustrated in Fig. 1. First, from the collected network traffic file, the individual bidirectional flows are separated. A network flow is a group of several associated packets [26] grouped by the 5-tuple: source IP address, destination IP

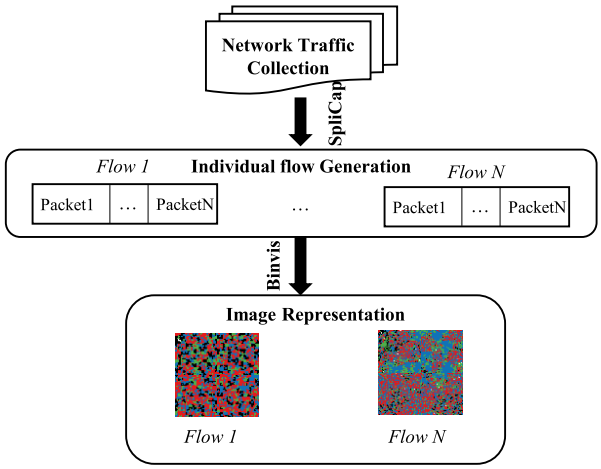

Fig. 1 Network traffic preprocessing.

Table 1 Color mapping by the Binvis binary data visualization tool.

Color	Description of ASCII characters
White	0xFF
Black	0x00
Blue	Printable
Green	Control
Red	Extended

address, source port, destination port, and protocol. Each flow contains the hexadecimal sequence of the time-adjacent packets identified by the same 5-tuple. Then, the hexadecimal values in each flow are transformed into the RGB color image using a binary data visualization tool called Binvis [27]. Binvis maps each hexadecimal value in a flow to a predefined color conversion scheme as illustrated in Table 1. The generated output is the one-dimensional color sequence of each flow. The next step is to lay out each generated color sequence as an RGB color image while preserving the proximity of the elements in the one-dimensional sequence to be as near as possible in the two-dimensional image. This can be achieved by means of the Hilbert space-filling curve [28]. After positioning the color sequence of the network flow in a two-dimensional layout, the desired RGB color image can be produced. When visualizing the individual flow as an image, we consider the whole packet (i.e., both header and payload). In our approach, each IoT network traffic flow is visualized as a color image of size 256×256. Examples of visualized network traffic are presented in Fig. 2. One benefit of visual representation is that it provides a clearer view and comprehensive understanding of the overall network traffic since the same malware traffic families tend to generate similar image patterns.

3.3 Few-Shot Prototypical Graph Neural Network

An overview of the proposed few-shot GNN model is illustrated in Fig. 3. The architecture consists of four main parts: feature extraction, a prototype encoder, a graph constructor, and a graph neural network classifier.

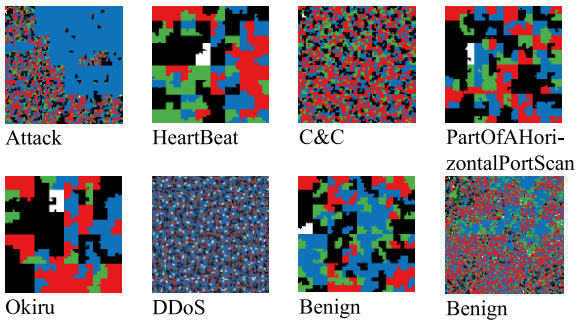

Fig. 2 Visual representation of network flows.

(i) Feature Extraction
The aim of the feature embedding function is to extract important features in the embedding space. A variety of CNNs or deep neural networks can be used to extract features. The proposed model utilizes a pretrained ResNet18 [29] as the feature embedding function. ResNet18 is a CNN model that consists of 18 convolution layers. After removing the final fully connected layer intended for classification, the remaining layers in ResNet18 can be regarded as the feature embedding module. The feature extractor applied in our proposed scheme is pretrained on the ImageNet dataset [30]. The dimension of the output features vector is 512.

(ii) Prototype Computation
After feature embedding, the prototype of each class is computed with the label information from the support set. The approach described in [12] is used to obtain the prototype of each class by obtaining the mean of the support set's feature embeddings associated with that class. The following equation calculates the prototype belonging to class c:

$$Proto_c = \frac{1}{|S_c|} \sum_{(x_j, y_j) \in S_c} f_\theta(x_j),$$

where $f_\theta(x_j)$ is the feature embedding of the j^{th} class from the support set, while S_c represents the set of support samples belonging to class c. Next, the Euclidean distance is calculated, which is the distance between the computed class prototype and query instance. The minimum distance is chosen as the initially predicted label for the query instance.

(iii) Graph Construction
Following the computation of the initial labels for the query set instances, a graph consisting of support set images and query set images is constructed. Generally, a graph is formed by a group of nodes with the link (edge) between nodes. In the proposed scheme, we consider images from both the query and support sets as nodes. Since the labels of the instances in the support set are already known, an edge is added between support nodes if they are from the same class. The relations between the support and query nodes

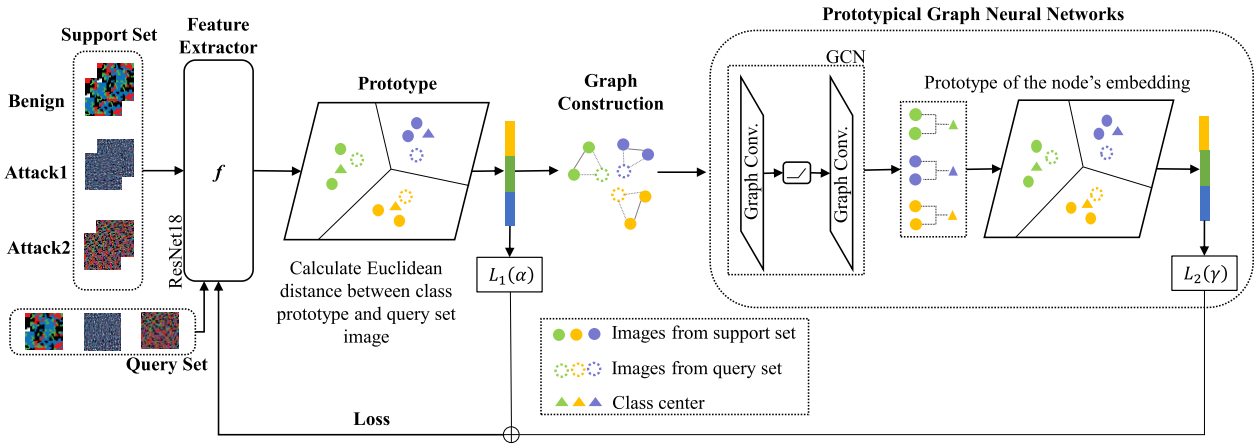

Fig. 3 Proposed few-shot learning-based network traffic detection system.

are obtained from the previously computed Euclidean distance. If the initially predicted class of the query node and the class of the support nodes are the same, there is a relation (edge) between the nodes. The constructed graph and the concatenated feature embeddings of both sets are forwarded to the GNN classifier.

(iv) Prototypical Graph Neural Networks

The proposed method employs a graph convolutional network (GCN), a variant of a GNN proposed in [31], as the graph classifier. Similar to a CNN, which is commonly employed in the domain of computer vision, a GCN performs multilayer convolution on graph-structured data. A GCN can be used for tasks such as link prediction, graph classification, and node classification. In the proposed approach, node classification is applied to the graph constructed in the previous step to predict the label information of the unlabeled query nodes. The outcome of the graph classifier is the predicted labels of the query set. Below we explain a multilayer GCN for node classification used in our proposed method.

For a given graph G with vertices V and edges E, the input of the GCN is the node features, adjacency matrix, and label information. The output of the GCN is the node classification results, which predict the unlabeled nodes. A multilayer GCN [31] defines the layer-wise propagation rule as follows:

$$H^{(l+1)} = \sigma\left(\tilde{D}^{(-\frac{1}{2})}\tilde{A}\tilde{D}^{(-\frac{1}{2})}H^{(l)}W^{(l)}\right). \quad (1)$$

For an undirected graph $G = (V, E)$ with nodes $v_i \in V$ and edges $(v_i, v_j) \in E$, let $A \in R^{N\times N}$ and $X \in R^{N\times C}$ be the adjacency matrix and feature vector of G, respectively, and let $D = \sum_j A_{ij} \in R^{N\times N}$ be the degree matrix of A, where $i, j = (1, \ldots, N)$, N is the total number of vertices V, and C is the dimension of the feature vector. In Eq. (1), $H^{(l)} \in R^{N\times D}$ is the feature vector that is input to the l^{th} layer of the GCN. Therefore, we can say that H^0 is identical to the input feature vector X. Here σ denotes the activation function, and $W^{(l)}$ is the layer-specific trainable weight matrix of the l^{th} hidden layer. The notation $\tilde{D}^{(-1/2)}\tilde{A}\tilde{D}^{(-1/2)}$ is the normalized adjacency matrix with a self-loop, where $\tilde{A}$ is the identity matrix of A, and $\tilde{D}$ is the degree matrix of $\tilde{A}$. The GCN performs aggregation, combining steps recursively in each layer. Each node calculates the mean value of the neighboring nodes' features and its own features. The aggregated feature values are then multiplied by weight W, and the status of every node in each layer is updated using the ReLU activation function. W is updated by the minimum cross-entropy loss function.

The prototypical graph neural network model used in the proposed approach is similar to [31]. Our graph model is made up of two parts: two-layer graph convolution and prototype computation of support node embedding. After the graph convolution layer computed the node embeddings for each support and query nodes, we calculate the prototype of the support node embeddings as described in Sect. 3.3. (ii). Simply, the mean value of the support node embeddings belonging to the same classes are calculated as the node prototypes (i.e. if the number of the support node class is four, then the total node prototype is four). Then, we find the Euclidean distance between the node prototypes and the query node embeddings. The graph classifier finally outputs the predicted class of the query nodes based on the nearest distance between the node prototypes and the query nodes.

3.4 Training Objectives and Parameters

During the meta-training, the model is optimized with two cross-entropy loss functions to make predictions for each query sample using its respective class prototypes. The first loss function for the prototype computation module is computed via cross-entropy loss as: $L_1(\alpha) = -\sum log\, P_\alpha(y \mid x, S_T)$, which is the negative log-likelihood of the true class of each query sample. Similarly, the Prototypical Graph Neural Networks module is optimized again with the cross-entropy loss function: $L_2(\gamma) = -\sum log\, P_\gamma(y \mid x, S_T)$. P_α and P_γ denote the class probabilities of the query sample in each episode over parameters α and γ. It can be calculated by taking the softmax over the Euclidean distance between the query sample x and each

class prototype P_c as: $P_\alpha(y = c \mid x) = \frac{exp(-ED(f_\alpha(x), Proto_c))}{\sum_{c'} exp(-ED(f_\alpha(x), Proto_{c'}))}$, where ED is the euclidean distance and $P_{c'}$ is the prototype of the class c'. The calculation of P_γ is similar to P_α. Therefore, the total loss during meta-training is: Loss = $L_1(\alpha) + L_2(\gamma)$ and the model parameters are updated by minimizing the total loss. The complete training algorithm for one few-shot task is provided in Algorithm 1.

The proposed prototypical graph neural network model consists of 2 hidden graph convolution layers with 8 hidden units. To avoid overfitting during training, a dropout layer of 0.002 is applied in the graph model. The proposed model utilizes the Adam optimizer with a learning rate of 0.001 and 5e-5. The model parameters are determined by tuning the hyperparameters.

Algorithm 1: The training algorithm for one few-shot task

Input: Meta-train dataset $M_{train} = \{(x_j, y_j) \mid y_j \in C\}$, C: the set of training classes, N_C: The number of classes in one few-shot task, N_S: The number of support samples per class, N_Q: The number of query samples per class

Requires: Feature extractor f

- $RANDOMSAMPLE(M_{train}, N)$ represents a set of N examples chosen randomly from M_{train} without replacement.
- $GCN(g)$ denotes the graph convolutional neural network with input graph g.
- $GRAPH(S_T, Q_T, img\_dist)$ constructs each image in S_T and Q_T as a node. The img_dist is the edge between the support node and query node. The edges between the support nodes are obtained from the label of the support set.

Output: Loss L for back propagation

$L \leftarrow 0$ #initialize loss
for T in $\{1, 2, \ldots, N_C\}$ **do**
  #select the support samples set
  $S_T \leftarrow RANDOMSAMPLE(M_{train}, N_C \times N_S)$
  #select the query samples set
  $Q_T \leftarrow RANDOMSAMPLE(M_{train} \setminus S_T, N_C \times N_Q)$
end
Compute the support feature $f_\theta(x_j)$ and query feature $f_\alpha(x_j)$ from S_T and Q_T
for c in $\{1, 2, \ldots, N_C\}$ **do**
  #compute image prototype for class c
  $Proto_c = \frac{1}{|S_T|} \sum_{(x_j, y_j) \in S_T} f_\theta(x_j)$
end
#calculate Euclidean distance
$img\_dist = min(euclidean\_dist(f_\alpha(x_j), Proto_c))$
#loss for image prototype computation
$L_1(\alpha) = -\sum log\, P_\alpha(y \mid x, S_T)$, where,
$P_\alpha(y = c \mid x) = \frac{exp(img\_dist)}{\sum_{c'} exp(img\_dist)}$

#construct graph
$g = GRAPH(S_T, Q_T, img\_dist)$
#compute two-layer graph convolution on graph g
$output = GCN(g)$
Compute the support node embedding $f_{g\theta}(x_j)$ and query node embedding $f_{g\alpha}(x_j)$ from $output$

for c in $\{1, 2, \ldots, N_C\}$ **do**
  #compute node prototype for class c
  $nProto_c = \frac{1}{|S_T|} \sum_{(x_j, y_j) \in S_T} f_{g\theta}(x_j)$
end
$node\_dist = min(euclidean\_dist(f_{g\alpha}(x_j), nProto_c))$
#loss for graph prototype computation
$L_2(\gamma) = -\sum log\, P_\gamma(y \mid x, S_T)$, where,
$P_\gamma(y = c \mid x) = \frac{exp(node\_dist)}{\sum_{c'} exp(node\_dist)}$
$L \leftarrow L + L_1 + L_2$

4. Evaluation

This section evaluates and discusses the efficiency of our proposed few-shot learning-based IoT network traffic detection method. The IoT-23 dataset [32] was used to show the efficiency and performance of the proposed model, and the proposed method was also compared to two baseline methods under the same setting: Prototype Networks [12] and FGNN [25]. The proposed model is trained on NVIDIA Quadro RTX 5000 with 16GB memory.

4.1 Dataset Description

The experiment in this study was conducted on the IoT-23 dataset, which consists of 20 malware traffic captures and three benign traffic captures, collected from 2018 to 2019. The benign traffic was gathered from a Philips Hue LED lamp, Amazon Echo, and Somfy smart door lock, while the malicious traffic was generated by simulating various botnet attacks on those IoT devices. The dataset includes the original captured network files (in PCAP format) and the log files, generated by the Zeek network analyzer [33], with a total of 21 feature attributes, including the label information. The dataset is made up of approximately 325 million labeled traffic flows. Since the proposed method converts network traffic flow into an image, the original PCAP file of the IoT-23 dataset was used to build the dataset. Although the original dataset consisted of millions of flows, our dataset contained only a small portion of the original dataset. From the PCAP files, each bidirectional network flow was separated by SplitCap [34], and the ground truth information for each flow was gathered from the conn.labeled.log file provided in the original dataset. The statistics of the IoT traffic flow dataset utilized in the experiment are presented in Table 2. The dataset had the following seven classes: Attack, Heartbeat, C&C, PartOfAHorizontalPortScan, DDoS, Okiru, and Benign.

Table 2 Statistics of dataset used in experiment.

Class label	Number of flows
Benign	4,871
Attack	1,327
Heartbeat	3,678
C&C	2,139
DDoS	28,041
Okiru	22,409
PartOfAHorizontalPortScan	21,033
Total	83,498

Table 3 Evaluation results. The evaluation metrics for multi-class classification are macro-averaged.

Model	Test class	4-way 5-shot			4-way 10-shot		
		Precision	Recall	F1 score	Precision	Recall	F1 score
Prototypical Networks [12]	Benign	0.7270	0.8230	0.7720	0.7707	0.8640	0.8147
	DDoS	**0.9738**	0.8540	0.9100	**0.9747**	0.8640	0.9058
	Okiru	0.9585	0.9010	0.9289	**0.9935**	0.9200	0.9555
	PortScan	0.8354	0.8700	0.8562	0.8618	0.9300	0.9020
	Average	0.8737	0.8620	0.8668	0.9002	0.8860	0.8945
FGNN [25]	Benign	**0.9185**	**0.9010**	**0.9096**	**0.9409**	**0.9560**	**0.9484**
	DDoS	0.8863	0.8730	0.8796	0.9612	0.9260	0.9435
	Okiru	**0.8821**	**0.9800**	**0.9285**	0.9306	0.9260	0.9283
	PortScan	0.8613	0.7950	0.8268	0.8918	0.9150	0.9033
	Average	0.8871	0.8873	0.8861	0.9313	0.9308	0.9306
Proposed	Benign	0.8293	0.8160	0.8226	0.8870	0.8870	0.8870
	DDoS	0.9624	**0.9730**	**0.9677**	0.9677	**0.9880**	**0.9777**
	Okiru	0.9594	0.9690	0.9642	0.9694	**0.9510**	**0.9601**
	PortScan	**0.8764**	**0.8720**	**0.8742**	**0.9339**	**0.9320**	**0.9329**
	Average	**0.9069**	**0.9075**	**0.9072**	**0.9395**	**0.9395**	**0.9394**

4.2 Evaluation Metrics

To determine the performance of the few-shot learning method, a confusion matrix was computed from the final classification results. The confusion matrix contained four outcomes. True positive (TP) and true negative (TN) indicates the number of benign/malicious samples that were accurately classified. False positive (FP) denotes the number of benign samples that were incorrectly classified as malicious samples, while false negative (FN) represents the number of malicious samples that were incorrectly classified as normal samples. Based on these four values, the evaluation metrics: precision, recall, and F1 score were computed as follows:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

4.3 Results

The effectiveness of the model was measured on a dataset consisting of seven types of network traffic, as illustrated in Table 2. The dataset was separated into a meta-training set, which included benign traffic and three types of malicious traffic (Attack, Heartbeat, C&C), and a meta-test set, which included benign traffic and the rest of the malicious traffic. The reason for choosing DDoS, Okiru, and PortScan classes as the meta-test dataset is to study a scenario where a large volume of unseen and unlabeled malware traffic is available to analyze. Only these three classes in our dataset satisfy the condition. Moreover, it is possible to train the model with any combination of attack classes as long as the attack classes in the meta-train and meta-test dataset are not overlapped since the goal of our few-shot model is to investigate how well the meta-trained model generalizes the unseen classes, given a small number of labeled samples of each unseen class.

Both the training and test datasets had a total of four classes, and the few-shot learning problem became a 4-way K-shot classification problem. Since benign traffic is used in both datasets, 1,690 benign samples are used for training, thus making a total of 8,834 and 74,664 samples for meta-train and meta-test dataset, respectively. As mentioned in Sect. 3.1, to train the few-shot model in an episodic manner, we need to sample multiple tasks such that each task consists of a support set and a query set. We did not explicitly split the meta-train dataset as a support set and query set. Rather, the support and query examples for one training task are randomly sampled from the meta-train dataset. For the meta-test dataset, the labeled 2% of the meta-test dataset is regarded as the support test set and the rest of the unlabeled samples are considered as the query test set. At the meta-testing stage, a test task consists of support and query examples are randomly drawn from the support test set and the query test set. As the purpose is to use limited labeled samples as much as possible, our model is trained and tested with 5-shot and 10-shot samples, and the experiment results are presented in Table 3. During the meta-training, we train a total of 30 episodes, with each episode consisting of 100 tasks. For instance, for 4-way 5-shot classification with one query image per class, we sample a total of $4 * (5 + 1) = 24$ images for each task based on the formula $N_{way}(N_{shot} + N_{query})$. Therefore, for each episode, we sample a total of 24 * 1000 = 2400 images from meta-train dataset. For simplicity, the number of query sample for one task is

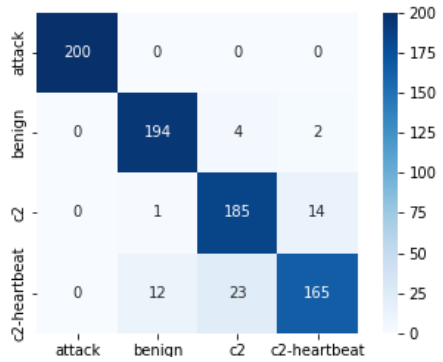

Fig. 4 Evaluation results of the trained 4-way 5-shot model on attack, benign, C&C, and Heartbeat classes.

chosen as one per class, but we can choose any number of query samples per class in one task.

The proposed model achieved an F1 score of 0.91 in 5-shot classification and 0.94 in 10-shot classification. The average recall and precision in both 5-shot and 10-shot classification was above 90%. To exhibit the efficiency of the proposed method, we compared it to two baseline methods: Prototypical Networks [12] and FGNN [25]. The former had an F1 score of 0.8945, while the latter had an F1 score of 0.9306 in 10-shot setting. The proposed approach outperforms both methods in terms of the average F1 score in both 5-shot and 10-shot classification. Comparing our proposed method to FGNN in 10-shot classification, our model performed only slightly better in all average evaluation metrics than FGNN. However, in 5-shot classification, all average evaluation metrics of our proposed method were approximately 2% higher than those of FGNN. Thus, our proposed method can effectively use a few labeled samples to classify IoT network traffic.

In addition, to investigate how well the trained model performs on the attack, benign, C&C, and Heartbeat classes, the evaluation results of the 4-way 5-shot model for those classes are illustrated in Fig. 4. 10% of each test class is labeled, and used as the support set. The rest of the samples are regarded as the query set that is needed to be predicted by the few-shot learner. The overall accuracy is 93%. The model can correctly classify the attack class, however, the classes C&C and Heartbeat incorrectly predict the output label as each other since the Heartbeat class in the IoT-23 can be considered a sub-class of the C&C class.

4.4 Discussion

The proposed model visualizes the individual network flow as an image to train the model. Therefore, the data preprocessing includes two steps: the individual network flow separation and the conversion of each flow as an image. Since the size of the PCAP file in the IoT-23 dataset is large and the captured duration is long, generating the bidirectional network flows from the IoT-23 dataset takes longer. However, just a few milliseconds are required to visualize each flow as an image. Making the image dataset (i.e., Table 2) from the network flows takes approximately 40 minutes. Since our few-shot model operates upon the graph structure, it takes additional cost to construct the graph. As the graph is built for each few-shot task, for instance, for 4-way 5-shot classification with one query image per class, each task consists of 20 support images and 4 query images, producing 24 nodes for the graph. Since the number of nodes is small and the edges are already estimated by the prototype computation module, the graph can be generated immediately. The training time of the proposed model for 30 episodes, excluding the data preprocessing time (i.e., network flow separation and image visualization), takes approximately 20 minutes for 4-way 5-shot learning with 100 tasks in one episode.

Even though the proposed model exhibited excellent performance, it also has limitations, such as the validity of the dataset and the deployment of the proposed model in a real environment. The number of IoT devices deployed to create the IoT-23 dataset is somewhat small, and the variations of the simulated attack are considerably fewer, which could further limit the real-world capability of the IoT-23 dataset. Though the IoT-23 dataset used in the study was published in 2020, some IoT traffics was captured in 2018. It means the trained few-shot model is behind the current IoT ecosystem by almost four years. Since IoT is an emerging technology, this time gap enables various changes in IoT devices, such as protocol, firmware, and OS version. Consequently, these changes can open up a potentially exploitable environment for malicious attackers. Therefore, constant supervision and frequent update of the IoT data is required to maintain the model's effectiveness.

Our insight on the possibility of deploying the proposed model in the real environment is that as long as the individual network flow is provided into the proposed model in real time, the deployment of the trained model should be possible for real-time detection. The testing time of the proposed model might be inferior to the conventional machine learning model since the model needs to construct the few-shot task from the meta-test dataset and creates the graph for traffic classification. However, considering the resource-limited IoT devices, the size of the classifier and the data preprocessing step could be a heavy burden while deploying on a single IoT device. Instead, the proposed model could be deployed in a place (e.g., an edge server) with sufficient resources to analyze the network traffic transmitted by the IoT devices in real-time. One advantage of the proposed few-shot learning over conventional machine learning is that the model retraining is unnecessary if a new type of attack has emerged. We only need to add a limited labeled sample of the unseen attack classes in the support set of the meta-test dataset so that the trained model can classify the unseen classes.

5. Conclusion

In this paper, we propose a few-shot meta-learning approach based on the prototypical graph neural network to classify malicious network traffic in IoT devices. Although ML

models perform reasonably well in recognizing malicious samples, they require a considerable amount of traffic samples to train the model. This problem is addressed by our proposed method by the use of the episodic few-shot learning paradigm. By directly converting bidirectional network flows into images, useful features can be automatically extracted using a pretrained CNN model. After that, our model learns how close the two network flows are in the embedding space using the Euclidean distance function. With this information, the model constructs the traffic image as a graph structure and classifies it using the proposed approach. Our experimental results demonstrate that the proposed model is better than the baseline models. Our approach can identify the malicious IoT traffic with an F1 score of 0.94 and 0.91 in 10-shot and 5-shot classification, respectively. Even though the advantage of few-shot learning is that the training classes and test classes do not need to be the same, it has its own limitation. Since the number of classes is fixed during the meta-training, meaning that if we train the model with N classes, unfortunately, only N classes can be predicted at a time in the meta-testing stage.

Acknowledgments

This work was partially supported by JSPS KAKENHI Grant Number JP21H03444. This research results were partly obtained from the commissioned research under a contract of "Research and development on IoT malware removal/make it non-functional technologies for effective use of the radio spectrum" among "Research and Development for Expansion of Radio Wave Resources (JPJ000254)", which was supported by the Ministry of Internal Affairs and Communications, Japan.

References

[1] Y. Yang, L. Wu, G. Yin, L. Li, and H. Zhao, "A survey on security and privacy issues in Internet-of-Things," IEEE Internet Things J., vol.4, no.5, pp.1250–1258, April 2017. DOI: 10.1109/JIOT.2017.2694844

[2] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications," IEEE Internet Things J., vol.4, no.5, pp.1125–1142, March 2017. DOI: 10.1109/JIOT.2017.2683200

[3] M.B.M. Noor and W.H. Hassan, "Current research on Internet of Things (IoT) security: A survey," Computer networks, vol.148, pp.283–294, Jan. 2019. DOI: 10.1016/j.comnet.2018.11.025

[4] M. Antonakakis, et al., "Understanding the mirai botnet," in 26th USENIX security symposium (USENIX Security 17), pp.1093–1110, Aug. 2017.

[5] G. Thamilarasu and S. Chawla, "Towards deep-learning-driven intrusion detection for the internet of things," Sensors, vol.19, no.9, p.1977, Jan. 2019. DOI: 10.3390/s19091977

[6] I. Alrashdi, A. Alqazzaz, E. Aloufi, R. Alharthi, M. Zohdy, and H. Ming, "Ad-iot: Anomaly detection of iot cyberattacks in smart city using machine learning," in 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), pp.0305–0310, Jan. 2019. DOI: 10.1109/CCWC.2019.8666450

[7] M. Almiani, A. AbuGhazleh, A. Al-Rahayfeh, S. Atiewi, and A. Razaque, "Deep recurrent neural network for IoT intrusion detection system," Simulation modelling practice and theory, vol.101, p.102031, May 2020. DOI: 10.1016/j.simpat.2019.102031

[8] J.L. Leevy, T.M. Khoshgoftaar, and J.M. Peterson, "Mitigating class imbalance for iot network intrusion detection: A survey," in 2021 IEEE Seventh International Conference on Big Data Computing Service and Applications (BigDataService), pp.143–148, Aug. 2021. DOI: 10.1109/BigDataService52369.2021.00023

[9] L. Bilge and T. Dumitraş, "Before we knew it: An empirical study of zero-day attacks in the real world," in Proceedings of the 2012 ACM conference on Computer and communications security, pp.833–844, Oct. 2012. DOI: 10.1145/2382196.2382284

[10] O. Vinyals, C. Blundell, T. Lillicrap, and D. Wierstra, "Matching networks for one shot learning," Advances in neural information processing systems, vol.29, pp.3630–3638, 2016

[11] F. Sung, Y. Yang, L. Zhang, T. Xiang, P.H.S. Torr, and T.M. Hospedales, "Learning to compare: Relation network for few-shot learning," in Proceedings of the IEEE conference on computer vision and pattern recognition, pp.1199–1208, 2018. DOI: 10.1109/CVPR.2018.00131

[12] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," Advances in neural information processing systems, vol.30, pp.4077–4087, 2017.

[13] V. Garcia and J. Bruna, "Few-shot learning with graph neural networks," arXiv preprint arXiv:1711. 04043, 2017.

[14] R. Shire, S. Shiaeles, K. Bendiab, B. Ghita, and N. Kolokotronis, "Malware squid: A novel IoT malware traffic analysis framework using convolutional neural network and binary visualisation," in NEW2AN, 2019. DOI: 10.1007/978-3-030-30859-9_6

[15] G. Conti, E. Dean, M. Sinda, and B. Sangster, "Visual reverse engineering of binary and data files," Visualization for computer security, pp.1–17, Sept. 2008. DOI: 10.1007/978-3-540-85933-8_1

[16] W. Liu, X. Liu, X. Di, and H. Qi, "A novel network intrusion detection algorithm based on Fast Fourier transformation," in 2019 1st International Conference on Industrial Artificial Intelligence (IAI), pp.1–6, July 2019. DOI: 10.1109/ICIAI.2019.8850770

[17] F. Hussain, S.G. Abbas, M. Husnain, U.U. Fayyaz, F. Shahzad, and G.A. Shah, "IOT dos and ddos attack detection using ResNet," 2020 IEEE 23rd International Multitopic Conference (INMIC), pp.1–6, Nov. 2020. DOI:10.1109/inmic50486.2020.9318216

[18] M. Shafiq, Z. Tian, A.K. Bashir, X. Du, and M. Guizani, "CorrAUC: A malicious bot-IoT traffic detection method in IoT network using machine-learning techniques," IEEE Internet Things J., vol.8, no.5, pp.3242–3254, June 2020. DOI: 10.1109/JIOT.2020.3002255

[19] G. Bendiab, S. Shiaeles, A. Alruban, and N. Kolokotronis, "IoT malware network traffic classification using visual representation and deep learning," in 2020 6th IEEE Conference on Network Softwarization (NetSoft), pp.444–449, June 2020. DOI: 10.1109/NetSoft48620.2020.9165381

[20] F. Hussain, S.G. Abbas, M. Husnain, U.U. Fayyaz, F. Shahzad, and G.A. Shah, "IoT DoS and DDoS attack detection using ResNet," in 2020 IEEE 23rd International Multitopic Conference (INMIC), pp.1–6, 2020. DOI: 10.1109/INMIC50486.2020.9318216

[21] C. Xu, J. Shen, and X. Du, "A method of few-shot network intrusion detection based on meta-learning framework," IEEE Trans. Inf. Forensics Security, vol.15, pp.3540–3552, May 2020. DOI: 10.1109/TIFS.2020.2991876

[22] Y. Yu and N. Bian, "An intrusion detection method using few-shot learning," IEEE Access, vol.8, pp.49730–49740, March 2020. DOI: 10.1109/ACCESS.2020.2980136

[23] F. Scarselli, M. Gori, A.C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," IEEE Trans. Neural Netw., vol.20, no.1, pp.61–80, Dec. 2009. DOI: 10.1109/TNN.2008.2005605

[24] J. Kim, T. Kim, S. Kim, and C.D. Yoo, "Edge-labeling graph neural network for few-shot learning," in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp.11–20, 2019. DOI: 10.1109/CVPR.2019.00010

[25] T. Wei, J. Hou, and R. Feng, "Fuzzy graph neural network

for few-shot learning," in 2020 International joint conference on neural networks (IJCNN), pp.1–8, July 2020. DOI: 10.1109/IJCNN48605.2020.9207213
[26] A. Dainotti, A. Pescape, and K.C. Claffy, "Issues and future directions in traffic classification," IEEE Netw., vol.26, no.1, pp.35–40, Jan. 2012. DOI: 10.1109/MNET.2012.6135854
[27] "Visual analysis of binary files," http://binvis.io
[28] H. Sagan, Space-filling curves, Springer Science & Business Media, 2012.
[29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proceedings of the IEEE conference on computer vision and pattern recognition, pp.770–778, 2016. DOI: 10.1109/cvpr.2016.90
[30] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in 2009 IEEE conference on computer vision and pattern recognition, pp.248–255, June 2019. DOI: 10.1109/CVPR.2009.5206848
[31] T.N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," arXiv preprint arXiv:1609. 02907, 2016.
[32] S. Garcia, A. Parmisano, and M.J. Erquiaga, "IoT-23: A labeled dataset with malicious and benign IoT network traffic," Zenodo, 2020, http://doi.org/10.5281/zenodo.4743746
[33] "An open source network monitoring tool," https://zeek.org
[34] "SplitCap," https://www.netresec.com/?page=SplitCap

Thin Tharaphe Thein received the B.E. degree from Yangon Technological University, Myanmar in 2019 and M.E. degree from Kobe University, Japan in 2021. She is currently pursuing doctoral degree at Kobe University. Her research focuses on data-driven cybersecurity, machine learning and threat intelligence.

Yoshiaki Shiraishi received the B.E. and M.E. degrees from Ehime University, Japan, and the Ph.D. degree from the University of Tokushima, Japan, in 1995, 1997, and 2000, respectively. From 2002 to 2006 he was a Lecturer at the Department of Informatics, Kindai University, Japan. From 2006 to 2013 he was an Associate Professor at the Department of Computer Science and Engineering, Nagoya Institute of Technology, Japan. Since 2013, he has been an Associate Professor at the Department of Electrical and Electronic Engineering, Kobe University, Japan. His current research interests include information security, cryptography, computer network, and machine learning based cyberattack analysis. He is a member of IEEE, ACM, and a senior member of IEICE, IPSJ.

Masakatu Morii received the B.E. degree in electrical engineering and the M.E. degree in electronics engineering from Saga University, Saga, Japan, and the D.E. degree in communication engineering from Osaka University, Osaka, Japan, in 1983, 1985, and 1989, respectively. From 1989 to 1990 he was an Instructor in the Department of Electronics and Information Science, Kyoto Institute of Technology, Japan. From 1990 to 1995 he was an Associate Professor at the Department of Computer Science, Faculty of Engineering, Ehime University, Japan. From 1995 to 2005 he was a Professor at the Department of Intelligent Systems and Information Science, Faculty of Engineering, the University of Tokushima, Japan. Since 2005, he has been a Professor at the Department of Electrical and Electronic Engineering, Faculty of Engineering, Kobe University, Japan. His research interests are in error correcting codes, cryptography, discrete mathematics, computer networks and information security. He is a member of IEEE and a fellow of IEICE.