E-CIS: Edge-based Classifier Identification Scheme in Green & Sustainable IoT Smart City

Abstract

E1–Smart city has brought the unprecedented development and application of Internet of things (IoT) devices. Meanwhile, since both of the quantity and the type of IoT devices are growing rapidly, how to quickly identify the type of IoT devices is of paramount importance, especially in the fields of IoT Device Security, IoT Forensics, Cyber Defense, and Cyber Threats Intelligence Sharing, to make the IoT smart city green and sustainable. Traditional identification mode based on device or gateway often suffers from their limited computing and storage resources. Our work is motivated by the observation of the emergence of edge computing, in which computing and storage servers are placed in close proximity to IoT/mobile devices. In this paper, we propose an Edge-based Classifier Identification Scheme (E-CIS) for IoT Devices, where the neighboring edge servers provide powerful computing and storage capabilities. E-CIS changes the traditional centralized architecture and realizes low time delay and efficient identification of IoT devices based on edge computing. Experiments show that the average identification accuracy is as high as 99.2%. Besides, the optimization and security of the classification model can be maintained by the edge servers at the same time.

Keywords: IoT, Edge computing, Classifier Identification, Massive IoT Device Management, IoT Device Security & Cyber Defense

I. Introduction

E1–With the rapid development of IoT technology, the number of IoT devices continues to increase in our daily life [1]–[3]. It is estimated that there will be more than 24 billion IoT devices in the world in the next 25 years. Such a huge scale of devices not only brings convenience to our life but also give rise to problems of network security and device management. On the one hand, the original equipment manufacturers are the traditional home appliance manufacturers. Due to the lack of information security awareness, the equipment they produced often had security vulnerabilities [4], [5]. Attackers can use the vulnerabilities of these types of equipment to launch attacks on the equipment and the network, and then control the equipment or paralyze the network. On the other hand, due to the rapid growth of IoT devices in the network,

administrators may unable to know all device assets of the network and have no idea what devices are connected to the network, which causes the problem of device asset management. Therefore, device identification is of great importance in IoT security, device management, and cyber defense during building a green and sustainable IoT smart city.

IoT device identification is to identify new device types that access the network by extracting hardware or network communication features of different types of devices. In terms of management, device identification enables the gateway to quickly identify the type of the accessing device, so as to allocate reasonable resources, and provide users with a more intelligent experience according to the device type. In terms of security [6], [7], because the vulnerability of the same type is consistent, we can further identify whether there is a vulnerability in the device after identifying the device type, so as to actively alert or isolate the device with the vulnerability. Such as in [8], the author distinguished the vulnerable devices and adopted the network isolation strategy to prevent the vulnerable devices from being used to attack the network after identifying the device type, and combined the system with the vulnerability database to protect network security in real-time.

Q1-1—At present, most of the research on IoT device identification is focus on identification methods, only a few are studying the whole device identification framework [9]–[11]. In [12], the author proposed a sentinel system about IoT, which extracted the access device's fingerprint by the security gateway. And then the security gateway sent the fingerprint to the IoT service center for device identification. The AUDL device identification system designed in [13] is similar, where a large number of gateways provide the device's fingerprint to the cloud server in parallel, and then the cloud server identifies the device based on the fingerprint database. These identification systems are all the centralized architectures. The device identification model is based on a centralized cloud server, which provides the underlying gateway device identification function. However, once the cloud is attacked, the whole system will be paralyzed. Moreover, it usually cannot ensure real-time feedback and identification efficiency as well as accuracy.

In [14], a distributed device fingerprint technology is proposed. The device identification system implemented by this technology consists of two entities - control logic and gateway. The gateway sends the extracted device feature vector to the control logic, which learns the device fingerprint and constructs the classifier, and then deploys the classifier in all gateways for device identification. On the surface, this is a distributed identification architecture, and each user gateway has a device identification function. However, since all the classifiers used for device

identification in the gateway come from a control logic in the upper layer, it is still a centralized identification architecture in essence.

Q2-2—Generally, apart from the decentralized demand of an ideal secure IoT device identification system, it also should meet the following requirements in performance [15]–[17].

Q2-2—1) Low time delay: Ensuring short time delay of a device from access to successful clustering identification.

Q2-2—2) Effectiveness: The architecture of the system must be distributed, which makes the identification service quality not be influenced by the gateway user numbers.

Q2-2—3) Scalability of device identification type: The system can constantly update the identification model, and the more new device types in the access network, the more device types the system can identify.

Edge computing [18], [19] can effectively utilize various resources close to the device, including network, computing, storage resources, etc., to provide users with real-time, dynamic and intelligent services to improve the real-time cooperation capability of all network devices. In terms of time delay, it can process and analyze data in real-time or even faster to make data processing since the edge server is closer to the data source rather than the external data center or the cloud, which greatly reduces interactive waiting time [20]. In terms of cost, edge computing does not require the assistance of remote cloud, and the cost of data processing solution in local equipment is much lower than that in the cloud and the data center [21]. In terms of security, due to the massive increase of data from IoT devices, the data traffic is growing rapidly, while edge computing can split the data, so as to avoid network bandwidth congestion and relieve the pressure of the center [22]. Besides, edge computing can enable continuous learning to adapt models to individual needs for real-time and efficient intelligent services.

In this paper, a secure and efficient classifier identification scheme for IoT devices based on edge computing is proposed. The identification model is constantly updated by utilizing adjacent edge resources of the gateway so that the device identification system can identify more and more types of devices with continuous learning ability. In addition, the collaboration mechanism is established among the gateways to ensure the continuous learning ability of the whole system. To sum up, the main contributions of this paper are as follows:

- E-CIS changes the traditional centralized architecture and realizes low time delay and efficient identification of IoT devices based on edge computing.
- The types of IoT devices E-CIS can identify keeps growing.

- All gateways of E-CIS have the continuous learning ability synchronously.
- Experiments show that the average identification accuracy is as high as 99.2%.

The rest of this paper is organized as follows. Section II shows the related works briefly. Section III describes the proposed E-CIS including the description of E-CIS architecture, the workflow, and core algorithms. Section IV shows the experiment results.

II. RELATED WORK

In 2017, Sivanathan et al. [23] collected network traffic features according to activity patterns, signals, protocols, etc., and used it to identify and classify Internet of things devices and detect abnormal behaviors, so as to solve the problem of network equipment management. This method can distinguish the IoT device with a 95% accuracy rate. In 2018, Santos et al. [24] proposed a feasible method for device identification. They characterize a device by extracting the statistical characteristics of network flow and packet payload, then train the identification model with a random forest algorithm. The accuracy rate of the device reaches 99%. In 2018, Bezawada et al. [25] proposed using device behavior fingerprints for device identification. The extracted behavioral fingerprints include static behavioral fingerprints (protocol usage) and dynamic behavioral fingerprints (session interaction). These fingerprints are used to train machine learning models, which are used to identify similar device types. The experimental results show that the average accuracy rate of this method is 99%. Shahid et al. [26] proposed a machine learning-based device identification method for the Internet of things, which can identify the type of devices connected to the network by analyzing the data packet flow sent and received. When the random forest classifier is used, the overall recognition accuracy rate of the method is as high as 99.9%. Sivanathan et al. [27] developed a classification framework based on multi-stage machine learning to realize the identification of IoT devices. The framework can uniquely identify IoT devices with an accuracy rate of more than 99%. In 2019, Hamad et al. [28] proposed to extract features from a set of data packet sequences sorted by timestamp as device fingerprints, and then established a model based on supervised machine learning method to realize the identification of IoT devices. In addition, they combined the white list strategy to restrict the network of unknown or suspicious devices. The results show that their proposed method can effectively identify the devices on the white list with a 90.3% accuracy rate.

Although the above identification methods have a good recognition effect, they do not take into account the subsequent identification of new equipment types and lack of model updating

mechanism, so they can only identify the fixed device types in the model training set. This is fatal for the ever-expanding IoT environment.

In 2018, Thangavelu et al. [29] proposed DEFT, a distributed device fingerprint technology. Deft works in hierarchical network architecture and consists of two entities - control logic and gateway. The gateway extracts the device fingerprint and identifies the device. If the identification fails, the fingerprint is sent to the control logic. The control logic then updates the identification model and deploys it to all gateways. Experiments show that the recognition accuracy rate of DEFT is 97%.

In 2019, Marchal, Samuel et al. [13] proposed an AUDI system to quickly and effectively identify the type of equipment by analyzing the network communication of the equipment. The system extracts the fingerprint of the device by the gateway and provides the identification service by the cloud service center. If the identification fails, the cloud service center clusters the fingerprint and retrains the recognition model. With the continuous access of new types, the system can also identify more and more types. The results show that the accuracy of AUDI is 98.2%.

Both DEFT and AUDI have realized the continuous updating of the identification model, and the recognition accuracy is very high. However, they are all based on the centralized architecture, and both need a central organization (control logic and cloud service center). The model updating is carried out on the central organization, so there are problems of high delay and high cost. Although the model update can be deployed to the local gateway, it also has the problem of limited local resources.

III. Edge-based Classifier Identification Scheme for IoT Devices

In this article, we put forward a secure and efficient classifier identification scheme based on edge computing for IoT devices, E-CIS, which has the continuous learning ability and identifies more and more types of devices. It adopts a decentralized architecture where all gateways make independent identification tasks and once there is a new device unidentified the gateway invokes the adjacent edge resources for feature extraction, classifying, and then updates the identification model according to the edge server with the rest gateways cooperatively and synchronously.

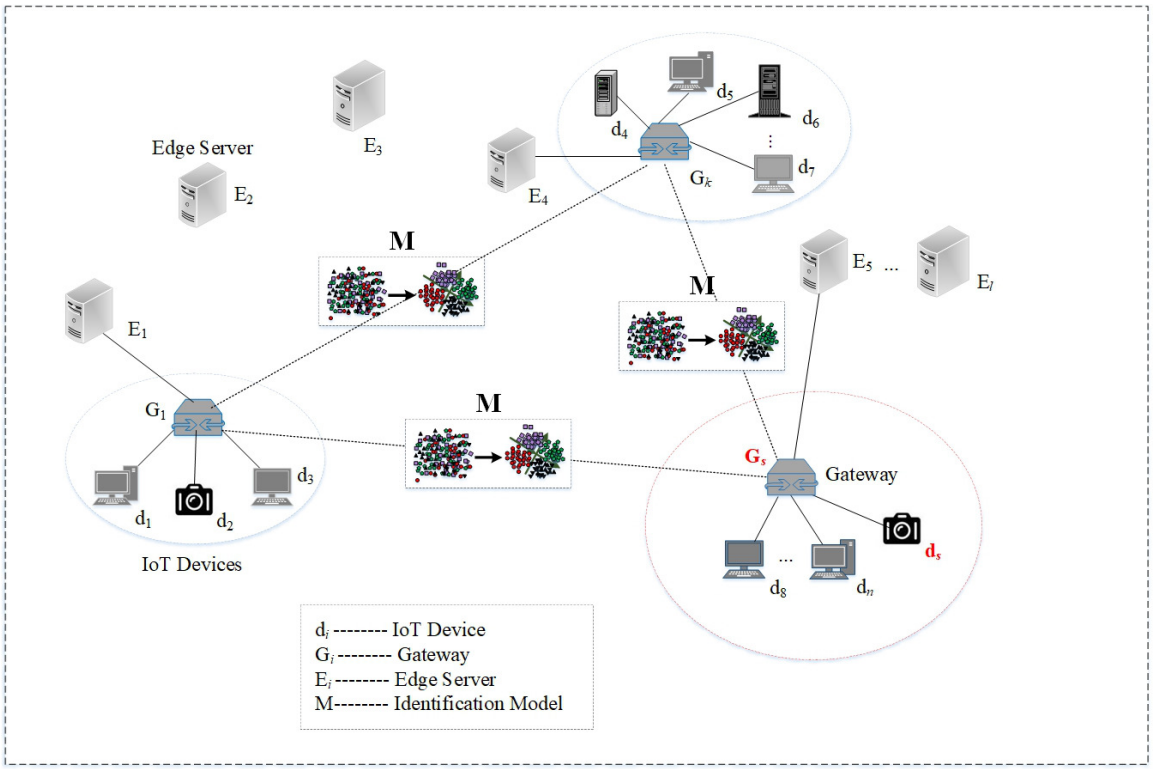

Fig. 1: Architecture of E-CIS

A. OVERVIEW OF E-CIS

The overall architecture of E-CIS is shown in Fig.1. Considering the scenario with a large number of IoT devices, multiple gateways are required to connect and manage the devices. However, there is no centralized entity like the cloud in the whole system. But all devices in IoT can connect to their neighboring edge servers for auxiliary calculation. All gateways in the system cooperate to process and manage all IoT devices and no centralized management servers are required. For the identification of IoT devices, the process is as follows:

Initialization: Assuming devices $d_1, \cdots, d_n$ are the initial devices when building the IoT environment, gateway $G_1, \cdots, G_k$ and edge servers $E_1, \cdots, E_l$ are initialized a training identification model denoted as M;

Device Classifier Identification: A device d_s joins by gateway G_s. G_s collects d_s's features, and performs device identification by calling the model M. In order to prevent the local gateway identification error, G_s distributes the feature vector to the rest gateways, and then they conduct secondary verification. If the identification results of the local gateway and most gateways are

consistent, then d_s is considered to be an unknown device, the feature vector will be sent to its edge server. The edge server updates the training set according to the newly added feature vector and updates the identification model of G_s. G_s then informs the rest gateways to collaboratively update the model synchronously. If the identification results are inconsistent, it indicates that there is a problem with the local gateway, and G_s will notify other gateways of the error, then they will cut off the connection with G_s; Herein, the edge servers help the gateways to updated synchronously to jointly maintain the identification model. Once a certain gateway model is attacked, there are two kinds of situations: one is that the device should be identified by the existed model that cannot be identified, the other is that the device should not be identified is identified by wrong classifying. The two kinds of security problems and the stable operation of the IoT system can be avoided by real-time updating the identification model synchronously.

B. E-CIS

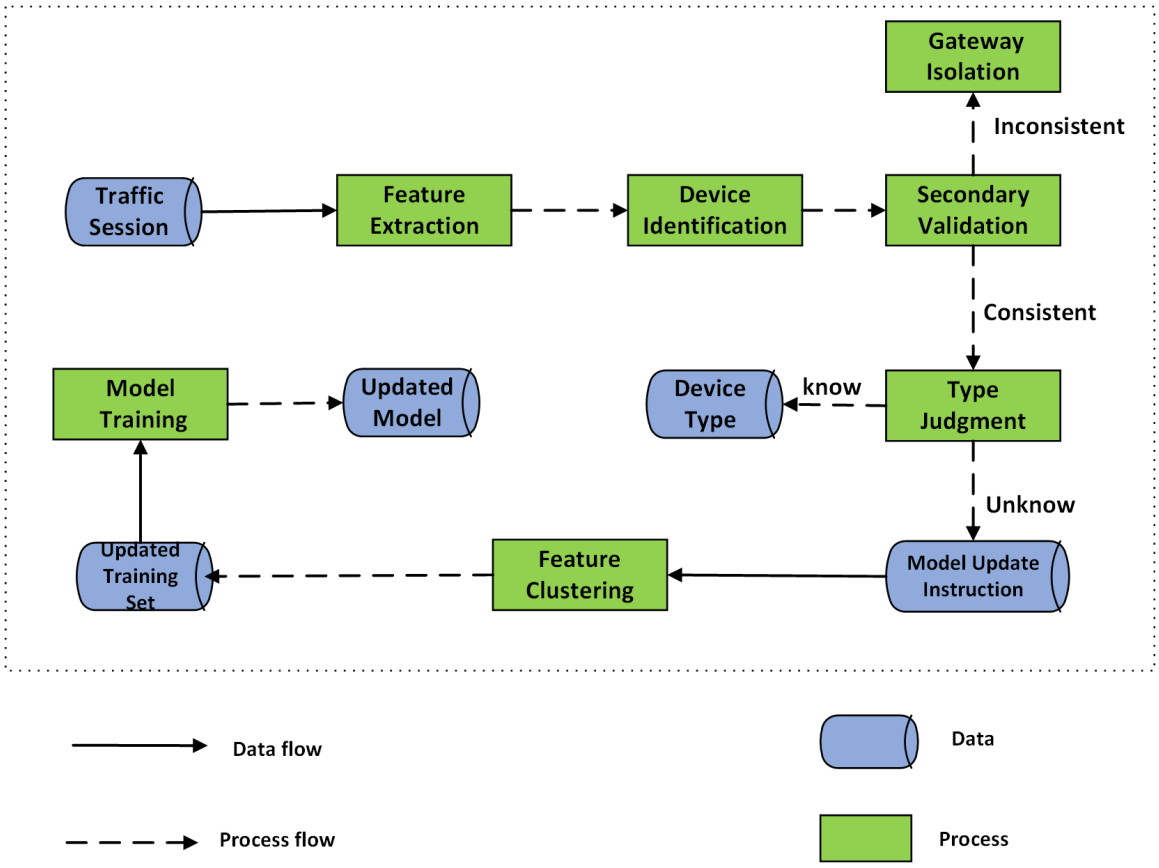

Fig. 2: **Q2-3**—The Workflow of E-CIS

Fig.2 shows the workflow of the scheme, the detailed processes are as follows. There are four main modules: **Feature Extraction Module**, **Device Identification Module**, **Training Set Updating Module** and **Model Retraining Module**. The first two modules are in the local gateway, the latter two modules work on the edge server.

a) **Feature Extraction Module**: Once a new device is to connect to the gateway, the module monitors the network traffic session and extracts the feature vector representing the device type;

b) **Device Identification Module**: the module mainly identifies the extracted feature vector through the identification model, and judges the device type (there are two types of "type" and "unknown" in the recognition result);

c) **Training Set Updating Module**: the module classifies the new feature vector with the original training set stored in the edge server to get the new model training set;

d) **Model Retraining Module**: the updated training set is used for model retraining to generate a new device identification model.

In order to provide good modularity and robustness of the scheme, the four modules **Feature Extraction Module**, **Device Identification Module**, **Training Set Updating Module**, **Model Retraining Module** can be developed as microservices and deployed to the gateway.

C. Core Algorithms

In this part, we will introduce the core algorithms in E-CIS in detail.

1) *Feature Extraction*

Specific features of network flow are extracted from traffic data, which are called features, and a group of such features is called the feature vector. The performance of the recognition model depends most on the feature vectors used for training. Therefore, it usually extracts as many features as possible from the network traffic to form the feature vectors to build the model. However, some features have limitations and are not suitable for large-scale IoT environments. For example, in 7, the features include domain name and port number. But some non-commercial entities may deploy devices that have never sent DNS requests, and the port number may be unique to the application. In addition, some features may cause security problems. For example, processing users' destination IP address can reveal their browsing behavior.

Q1-4—There are two kinds of traffic sessions for devices. One is based on the dynamic method. In this method, the interval between two consecutive sessions cannot be less than a

fixed time (for example, 1 minute). If it is less than the fixed time, the two sessions are regarded as one session. The other is to determine the session based on the static method which we used in this paper. Since a session between devices usually finishes in a very short period of time (a few seconds) and that the session package timestamp interval is minimal, so we treat different packages with intervals of less than 1 second as belonging to different sessions. Considering both the practicability and privacy of features, in this paper we select 15 device features that constitute the feature vector with the dimension of 15 ("stats" represents the average value and standard deviation of related features). The first 2 characteristics are the statistical characteristics (mean and standard deviation of packet size), and the remaining 13 features are the total number of protocol packets in a session. All of the features are shown in Tab.I.

TABLE I: Device Features

Feature Label	Feature Type	Feature
Label 0-2	Session statics	- packet count in session - packet length(stats)
Label 3-15	UDP	- ARP packet count
	TCP	- TCP packet count
	ARP	- ARP packet count
	BOOTP	- BOOTP packet count
	SSDP	- SSDP packet count
	NTP	- BTP packet count
	QUIC	- QUIC packet count
	mDNS	- mDNS packet count
	HTTP	- HTTP packet count
	TLS	- TLS packet count
	DNS	- DNS packet count
	STUN	- STUN packet count

2) *Feature Classfying*

In this section, we will propose two algorithms: classifying algorithm and model updating algorithm in detail. To improve readability, the main notations and their semantics used in this paper are given in Tab.II.

Herein, we use Euclidean Distance to measure the distance of the feature vectors. The Euclidean Distance between two feature vectors is calculated according to the following formula:

TABLE II: Notations

Notation	Description
f	the feature vector of device
F	Aggregate of feature vector
m	Dimension of the feature vector
M	Model for device identification
$\mathbb{T}$	List of feature aggregates
R	Dataset for model training

$$dist(u,v) = \sqrt{\sum_{i=1}^{m}(u_i - v_i)^2} \tag{1}$$

where *dist(u,v)* is the distance between the feature vectors *u* and *v*, *m* is the dimension of the vectors.

3) *Device Identification Algorithm*

This part mainly introduces the clustering algorithm in the cluster recognition module. In the selection of model establishment, considering that the gateway keeps the latest training set locally during model training, and the original type of feature set may change each time after the training set is updated. Therefore, it is not feasible to train a single class classifier for each type of device to recognize all devices. Because traditional *K-means algorithm* is unsupervised, it needs to know how many classes in advance. However, the goal of the proposed E-CIS is to continuously identify new types, so it is impossible to determine the number of clusters. Herein, in this paper, we use the random spanning tree algorithm to generate a multi-class classifier according to the local training set each time. In addition, we apply *seeded K-means algorithm* based on semi supervision.

In E-CIS, the clustering process is as follows. Before clustering, there are some known types of feature sets such as the original training set $\mathbb{R}$, which are used as seed feature sets to guide the subsequent clustering. When clustering a new feature, we only need to find the feature set nearest to the target feature in the seed feature set, and then judge whether the new feature comes

from the same type. If it is, the target feature set will be added to the feature set; otherwise, the target feature set will be added to the training set as a new type to become the seed feature set in the following clustering.

Algorithm 1 $Validation(f, Address)$

Input: f: feature vector; Address: Gateway IP address

Output: c:device type; p:prob of identification

1: $[c, p] \leftarrow identification(f, M)$
2: **if** $p < threshold$ **then**
3: $\quad type = "unknown"$
4: **else**
5: $\quad type = c$
6: **end if**
7: $sendToGateway(type, Address)$

Algorithm 1 is the validation algorithm for feature vector f. In order to verify the identification result of feature vector f, the gateway with IP address "Address" sends f to other gateways and collects the identification results of other gateways to judge whether there is any problem with its identification model. After receiving the feature vector f from "Address", other gateways identify it by using the local identification model, and return the results to "Address".

Algorithm 2 is the identification algorithm of a gateway. When a new device is connected, the local gateway monitors its traffic session S, and extracts the feature vector f of the device from it, and uses the local identification model M to identify it. In order to prevent the error of identification results, f is broadcast to other gateways for secondary verification. After collecting the identification results from all gateways, if the identification results of most gateways are consistent with the local gateways and the result is "unknown", the model updating instruction will be sent to other gateways to make all gateways update the model synchronously; If the result is a known type, the type "c" will be returned; If the identification results of the local gateway and most gateways are inconsistent, it indicates that the local gateway is under attack, and the gateway fault will be announced to other gateways.

Algorithm 2 $device\_identification(S)$

Input: S:Traffic session , Address: The IP address of this Gateway

Output: c:device type; p:prob of identification

$f = ExtractFeature(S)$
$[c, p] \leftarrow identification(f, M)$
if $p < threshold$ **then**
  $type = "unknown"$
else
  $type = c$
end if
$result = sendToAllGateway(f, Address)$
if $type = result$ **then**
  if $type = "unknown"$ **then**
    $updateModelSynchronously(f)$
  else
    return c
  end if
else
  $sendToAllGateway(fault, Address)$
end if

Algorithm 3 is to calculate the average distance between feature vectors in set P, which will be used as the basis for judging whether to generate a new class.

4) *Model Updating Algorithm*

Then in Algorithm 4, the edge server gets the latest feature set list $\mathbb{T}$. But at this time, the number of feature vectors contained in some feature sets in $\mathbb{T}$ may be very small. If the $\mathbb{T}$ is directly used as the training set, the samples of the training set will be extremely uneven, and the identification rate of the model generated from this will be very low. In addition, there is no

Algorithm 3 $mean\_distance(P)$

Input: P:Aggregate of cluster centroid

Output: ret:Mean of distance

1: $l = len(P)$
2: **if** $l < 2$ **then**
3: $ret = 0$
4: **else**
5: $ret = 0$
6: **for** $each\ i \in [0..l-1]$ **do**
7: $ret = ret + dist(P[i], P[i+1])$
8: **end for**
9: $ret = ret/(l-1)$
10: **end if**
11: **return** ret

need to retrain the model if $\mathbb{T}$ has no new feature set. Therefore, we added a parameter num to the algorithm, which is used to limit the number of samples of each type in the training set. Only when the number of samples of the new type reaches num, can it be added into the training set R. The if statement in line no. 8 is to judge whether it is necessary to update the model. If the updated training set R is larger than the original training set, it means that a new device type is generated, and there are enough samples of this type for model training. Otherwise, the model will not be retrained and continue to wait for the new type of samples.

IV. EVALUATION

In this section, we design experiments to evaluate the performance of the proposed E-CIS. Firstly, we introduce the experimental parameters including the types we used and then analyze the cost of each phase. Finally, we show the identification performance of E-CIS.

Algorithm 4 $model\_training(\mathbb{T})$

Input: $\mathbb{T}$:List of merged feature aggregates

Output: M:identification model

$l = len(R)$
$R = []$
for $i \in [0..len(\mathbb{T})]$ **do**
  if $len(\mathbb{T}[i]) > num$ **then**
    $R.add(\mathbb{T}[i])$
  end if
end for
if $len(R) > l$ **then**
  $M = train(R)$
  $sendToGateway(M)$
end if

A. Experimental Setup

The data set used in our experiment is from the one used in [30], which is collected for IEEE TMC 2018. The dataset contains traffic packets among 30 devices during September 23 to October 12, 2016. We use the CSV file of the first 5 days, and select the top 10 devices corresponding to the largest number of samples to ensure that the dataset is large enough, which is shown in Tab.III.

B. Computational Complexity

The computing process of E-CIS is all in the user gateway, whose main tasks are four: feature extraction, device identification, clustering, and model training. Because the time consumed by several well-known classifiers to classify a sample is linear with the number of features, and the number of features used in this paper is small and fixed, so the cost of device identification can be regarded as constant. Next, we mainly analyze the cost of the remaining three processes.

TABLE III: Information of IoT Devices

label	Device	numbel of instance
0	Amazon Echo	2500
1	Belkin Wemo switch	2500
2	HP Printer	2500
3	Netatmo weather station	2500
4	PIX-STAR Photo-frame	2500
5	Samsung SmartCam	2500
6	Smart Things	2500
7	TP-Link Day Night Cloud camera	2500
8	TP-Link Smart plug	2500
9	CamTriby Speaker	2500

- ***Feature Extraction***: **Q1-5, Q3-3—**— The goal of this process is to extract a feature vector from a session of the device. In essence, it traverses multiple packets contained in the session and finds the relevant features of each packet, such as the usage of various protocols, the length of the packet, and so on. Therefore, the cost of this process is $m \times O(n)$, where m represents the number of features that need to be extracted, and n is the total number of packets contained in the session.
- ***Clustering***: **Q3-3**— Analysis from Algorithm 4 shows that the most complex part of the algorithm is the for loop in line no. 8, so the cost of the clustering process is mainly determined by this for loop.The purpose of this for loop is to traverse F (The set of target feature vectors waiting for clustering) to find the possible cluster center (the farthest point). Line no. 9 inside the for loop traverses the entire P (It is composed of cluster centers of seed feature sets) to find the seed feature set closest to the target feature. So this process consumes $O(|P|) \times O(|F|)$.
- ***Model Training***: **Q3-3**— The algorithm used to establish the identification model is Random Forests. Random Forests classifiers contain multiple decision trees, and the classification results are generated by voting of all decision trees. So the time complexity is related to the number of decision trees, the final time complexity of this procedure is between $O(mB\,|f|\,log^2\,|f|)$ to $O(mB\,|f|^2\,log\,|f|)$ (Depends on data or how balanced the tree is) [31], Where m is the dimension of the feature vector, B is the number of decision trees initially set by the algorithm, and $|f|$ is the number of samples used for model training.

C. Performance

1) *Classification Algorithm*

E-CIS uses the classification algorithm of machine learning to realize device identification, but there are many kinds of classification algorithms available. Next, four common classification algorithms including KNN (K-nearest Neighbors), RF(Random Forest), LightGBM, and MLP(Multilayer Perceptron) were tested and the performance of classification models established by each algorithm was evaluated by using the method of "5 fold cross-verification". The specific experimental setting is as follows: We took all the samples (10 devices, 25000 samples in total) as the dataset. The "5 fold cross-validation" is to divide the data set into 5 parts in equal proportion, and takes one of them as the test set and the remaining 4 as the training set. Each classification algorithm carries out 5 rounds of experiments. Each round of experiment uses the divided training set for model training, and then uses the test set for classification test to obtain the classification accuracy rate of this round experiment. At the end of 5 rounds of experiments, the mean value of the accuracy rate of 5 times was calculated, which was the classification accuracy rate of the algorithm. The final experimental results are shown in Fig.3 and Tab. IV. It can be seen from the figure and table that the tree-based classifiers, random forest, and the LightGBM have better performance in both the overall dataset and the single category. Considering the high generalization ability of random forest, we adopt it as the identification algorithm in this paper.

TABLE IV: Accuracy of Four Classification Algorithms

Arithmetic	KNN	Random Forest	LightGBM	MLP
Accuracy	0.988	0.992	0.992	0.981

2) *Metrics For Evaluation*

The last experiment evaluated the classification algorithm, and we found that the RF algorithm had a higher accuracy rate. Next, we will evaluate the identification model established by using RF algorithm, and the evaluation metrics are $Precision$, $Recall$ and F_1Score which are commonly used to evaluate the classifier. Tab.V is the description of all parameters to be used in the calculation formula of the three evaluation metrics.

i) $Precision$ indicates the proportion of the samples with real prediction success among all the samples predicted as type A.

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

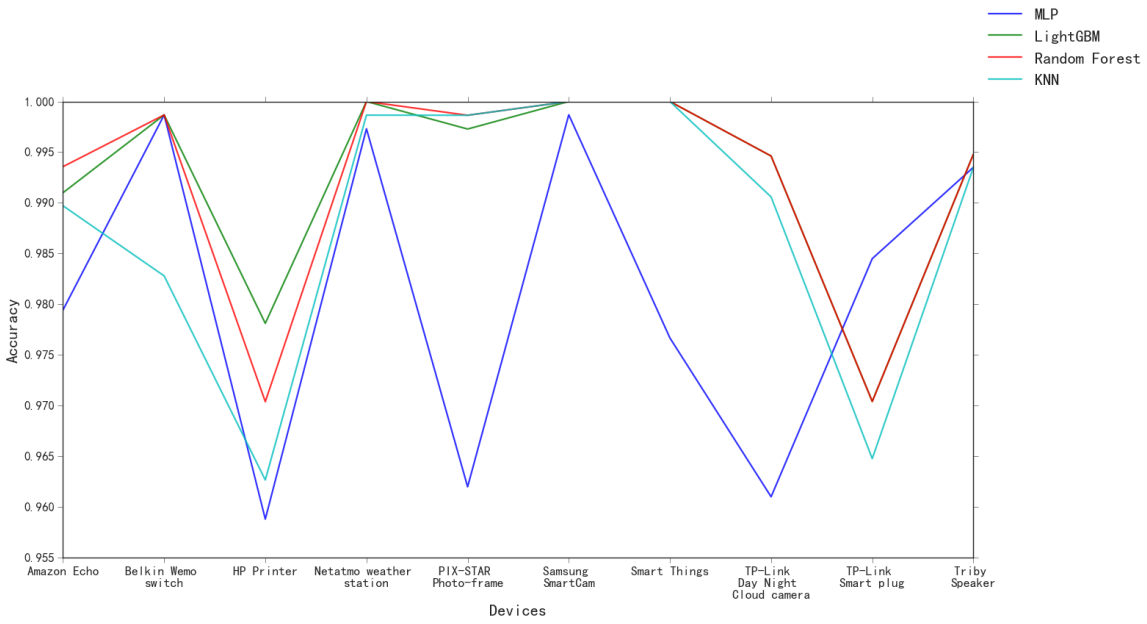

Fig. 3: Accuracy for 10 Device Types of Four Classification Algorithms

TABLE V: Table of Notations

Notation	Description
TP	The number of samples correctly identified as class A
FP	The number of samples incorrectly identified as class A
TN	The number of samples correctly identified as non-Class A
FN	The number of samples incorrectly identified as non-Class A

ii) $Recall$ indicates the proportion of the samples with prediction success among all the samples of actual type A.

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

iii) F_1score is the harmonic average of Precision and Recall, which is a comprehensive consideration of these two metrics.

$$F_1score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4}$$

3) *Experimental Results*

In order to objectively obtain the $Precision$, $Recall$ and F_1Score evaluation metrics of each device, we randomly divide the data set into a training set and test set according to the ratio of 7:3, and the final result is shown in Tab.VI. From the table, we can see that the F_1Score

of 10 kinds of devices is very high, among which 3 kinds of devices reach 1.0, and the lowest F_1Score is 0.985, which indicates that the identification model established by the RF algorithm is of high quality.

TABLE VI: Precision, Recall and $\mathrm{F_1Score}$ for 10 Types of IoT Devices

Device	**Precision**	**Recall**	$\mathbf{F_1Score}$
Amazon Echo	1.000	0.991	0.995
Belkin Wemo switch	1.000	0.999	0.999
HP Printer	1.000	0.978	0.989
Netatmo weather station	1.000	1.000	1.000
PIX-STAR Photo-frame	1.000	0.997	0.999
Samsung SmartCam	1.000	1.000	1.000
Smart Things	1.000	1.000	1.000
TP-Link Day Night Cloud camera	1.000	0.995	0.997
TP-Link Smart plug	1.0-0	0.970	0.985
CamTriby Speaker	1.000	0.995	0.997

In order to further evaluate the identification performance of the identification model on each device, we give a histogram in Tab.VII and the confusion matrix in Fig.4 of the accuracy of devices. From the figure, we can see that the lowest accuracy among the devices is also as high as 0.97. Therefore, it can be concluded that the identification model in this paper has good identification performance on these 10 devices.

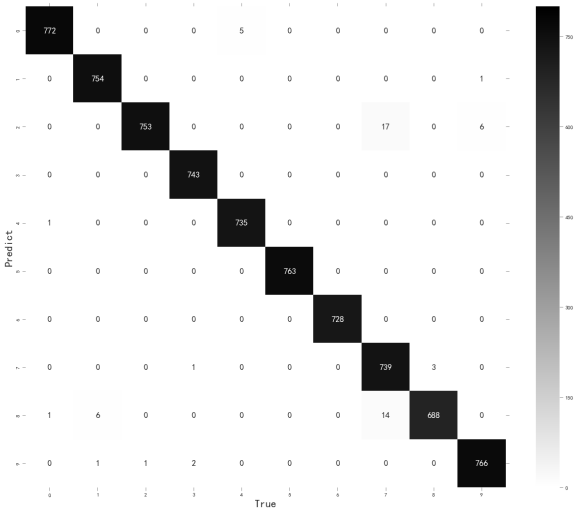

Fig. 4: Accuracy for 10 Device Types

TABLE VII: Accuracy for 10 Device Types

Device	Accuracy
Amazon Echo	0.994
Belkin WeMoSwitch	0.999
HP Printer	0.970
Netatmo Weather Station	1.00
Samsung SmartCam	1.00
PIX-STAR Photo-frame	0.999
Smart Things	1.00
TP-Link Day Night Cloud Camera	0.995
TP-Link Smart Plug	0.970
Triby Speaker	0.995

V. CONCLUSION

In order to resolve the problem of how to identify IoT devices with massive devices efficiently to build a green and sustainable IoT smart city, this paper proposes E-CIS based on edge computing. It simplifies the work of the device and gateway, introduces the adjacent edge computing resources to undertake the heavy computing tasks, and further improves the efficiency of the system. Experiments show that the E-CIS is effective and can continuously identify new device types. In the future, we will work on blockchain-based IoT device identification research to observe the advantages and disadvantages of the two technologies, edge computing and blockchain, in the fields of IoT device security, Internet of Things Forensics, cyber defense and Cyber Threats Intelligence Sharing.

VI. ACKNOWLEDGEMENT

This work is supported by Taif University Researchers Supporting Project number (TURSP-2020/126), Taif University, Taif, Saudi Arabia.

REFERENCES

[1] Z. Guo, K. Yu, T. Guo, A. Bashir, M. Imran, and M. Guizani, “Implicit feedback-based group recommender system for internet of thing applications,” 01 2021.

[2] L. Zhen, A. Bashir, K. Yu, Y. Al-Otaibi, C. Foh, and P. Xiao, “Energy-efficient random access for leo satellite-assisted 6g internet of remote things,” *IEEE Internet of Things Journal*, vol. PP, pp. 1–1, 10 2020.

[3] X. Lin, J. Wu, A. Bashir, J. Li, W. Yang, and J. Piran, "Blockchain-based incentive energy-knowledge trading in iot: Joint power transfer and ai design," *IEEE Internet of Things Journal*, vol. PP, pp. 1–1, 09 2020.

[4] J. Li, J. Jin, L. Lyu, D. Yuan, Y. Yang, L. Gao, and C. Shen, "A fast and scalable authentication scheme in iot for smart living," *Future Generation Computer Systems*, vol. 117, pp. 125–137, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X20330302

[5] I. H. Abdulqadder, S. Zhou, D. Zou, I. T. Aziz, and S. M. A. Akber, "Multi-layered intrusion detection and prevention in the sdn/nfv enabled cloud of 5g networks using ai-based defense mechanisms," *Computer Networks*, vol. 179, p. 107364, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128619310205

[6] R. M. A. Ujjan, Z. Pervez, K. Dahal, A. K. Bashir, R. Mumtaz, and J. Gonz á lez, "Towards sflow and adaptive polling sampling for deep learning based ddos detection in sdn," *Future Generation Computer Systems*, vol. 111, pp. 763–779, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X19318333

[7] S. ur Rehman, M. Khaliq, S. I. Imtiaz, A. Rasool, M. Shafiq, A. R. Javed, Z. Jalil, and A. K. Bashir, "Diddos: An approach for detection and identification of distributed denial of service (ddos) cyberattacks using gated recurrent units (gru)," *Future Generation Computer Systems*, vol. 118, pp. 453–466, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X21000327

[8] B. Genge and C. Enăchescu, "Shovat: Shodan-based vulnerability assessment tool for internet-facing services," *Security and communication networks*, vol. 9, no. 15, pp. 2696–2714, 2016.

[9] M. Shafiq, Z. Tian, Y. Sun, X. Du, and M. Guizani, "Selection of effective machine learning algorithm and bot-iot attacks traffic identification for internet of things in smart city," *Future Generation Computer Systems*, vol. 107, pp. 433–442, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X19334880

[10] N. Akatyev and J. I. James, "Evidence identification in iot networks based on threat assessment," *Future Generation Computer Systems*, vol. 93, pp. 814–821, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X17300857

[11] F. Xhafa, B. Kilic, and P. Krause, "Evaluation of iot stream processing at edge computing layer for semantic data enrichment," *Future Generation Computer Systems*, vol. 105, pp. 730–736, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X19321296

[12] P. D. Martin, D. Russell, A. D. Rubin, S. Checkoway, and M. B. Salem, "Sentinel: Secure mode profiling and enforcement for embedded systems," in *2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI)*. IEEE, 2018, pp. 105–116.

[13] S. Marchal, M. Miettinen, T. D. Nguyen, A.-R. Sadeghi, and N. Asokan, "Audi: Toward autonomous iot device-type identification using periodic communication," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1402–1412, 2019.

[14] J. Koo, S.-R. Oh, and Y.-G. Kim, "Device identification interoperability in heterogeneous iot platforms," *Sensors*, vol. 19, no. 6, p. 1433, 2019.

[15] G. G. Dagher, J. Mohler, M. Milojkovic, and P. B. Marella, "Ancile: Privacy-preserving framework for access control and interoperability of electronic health records using blockchain technology," *Sustainable Cities and Society*, p. S2210670717310685, 2018.

[16] A. Kh, B. Iud, C. Aa, D. Ia, and E. Mg, "Intelligent and secure edge-enabled computing model for sustainable cities using green internet of things," *Sustainable Cities and Society*, 2021.

[17] A. Yj, B. So, and C. Iar, "Trustworthy and sustainable smart city services at the edge," *Sustainable Cities and Society*, vol. 62, 2020.

[18] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE internet of things journal*, vol. 3, no. 5, pp. 637–646, 2016.

[19] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[20] G. Premsankar, M. Di Francesco, and T. Taleb, "Edge computing for the internet of things: A case study," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 1275–1284, 2018.

[21] W. Z. Khan, E. Ahmed, S. Hakak, I. Yaqoob, and A. Ahmed, "Edge computing: A survey," *Future Generation Computer Systems*, vol. 97, pp. 219–235, 2019.

[22] R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges," *Future Generation Computer Systems*, vol. 78, pp. 680–698, 2018.

[23] A. Sivanathan, D. Sherratt, H. H. Gharakheili, A. Radford, C. Wijenayake, A. Vishwanath, and V. Sivaraman, "Characterizing and classifying iot traffic in smart cities and campuses," in *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2017, pp. 559–564.

[24] M. R. Santos, R. M. Andrade, D. G. Gomes, and A. C. Callado, "An efficient approach for device identification and traffic classification in iot ecosystems," in *2018 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2018, pp. 00 304–00 309.

[25] B. Bezawada, M. Bachani, J. Peterson, H. Shirazi, I. Ray, and I. Ray, "Iotsense: Behavioral fingerprinting of iot devices," *arXiv preprint arXiv:1804.03852*, 2018.

[26] M. R. Shahid, G. Blanc, Z. Zhang, and H. Debar, "Iot devices recognition through network traffic analysis," in *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018, pp. 5187–5192.

[27] A. Sivanathan, H. H. Gharakheili, F. Loi, A. Radford, C. Wijenayake, A. Vishwanath, and V. Sivaraman, "Classifying iot devices in smart environments using network traffic characteristics," *IEEE Transactions on Mobile Computing*, vol. 18, no. 8, pp. 1745–1759, 2018.

[28] S. A. Hamad, W. E. Zhang, Q. Z. Sheng, and S. Nepal, "Iot device identification via network-flow based fingerprinting and learning," in *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. IEEE, 2019, pp. 103–111.

[29] V. Thangavelu, D. M. Divakaran, R. Sairam, S. S. Bhunia, and M. Gurusamy, "Deft: A distributed iot fingerprinting technique," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 940–952, 2018.

[30] A. Sivanathan, H. H. Gharakheili, F. Loi, A. Radford, C. Wijenayake, A. Vishwanath, and V. Sivaraman, "Classifying iot devices in smart environments using network traffic characteristics," *IEEE Transactions on Mobile Computing*, vol. 18, no. 8, pp. 1745–1759, 2018.

[31] F. Hutter, L. Xu, H. H. Hoos, and K. Leyton-Brown, "Algorithm runtime prediction: Methods & evaluation," *Artificial Intelligence*, vol. 206, pp. 79–111, 2014.