IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON SECURITY AND TRUSTED COMPUTING FOR INDUSTRIAL INTERNET OF THINGS

Received February 11, 2018, accepted April 8, 2018, date of publication April 30, 2018, date of current version May 24, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2831284

A DDoS Attack Detection and Mitigation With Software-Defined Internet of Things Framework

DA YIN[1], LIANMING ZHANG[1], AND KUN YANG[2], (Senior Member, IEEE)
[1]College of Information Science and Engineering, Hunan Normal University, Changsha 410081, China
[2]School of Computer Sciences and Electrical Engineering, University of Essex, Colchester CO4 3SQ, U.K.

Corresponding author: Lianming Zhang (zlm@hunnu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61572191 and Grant 61402170, and in part by the Research Foundation of Education Bureau of Hunan Province of China under Grant 17A130.

ABSTRACT With the spread of Internet of Things' (IoT) applications, security has become extremely important. A recent distributed denial-of-service (DDoS) attack revealed the ubiquity of vulnerabilities in IoT, and many IoT devices unwittingly contributed to the DDoS attack. The emerging software-defined anything (SDx) paradigm provides a way to safely manage IoT devices. In this paper, we first present a general framework for software-defined Internet of Things (SD-IoT) based on the SDx paradigm. The proposed framework consists of a controller pool containing SD-IoT controllers, SD-IoT switches integrated with an IoT gateway, and IoT devices. We then propose an algorithm for detecting and mitigating DDoS attacks using the proposed SD-IoT framework, and in the proposed algorithm, the cosine similarity of the vectors of the packet-in message rate at boundary SD-IoT switch ports is used to determine whether DDoS attacks occur in the IoT. Finally, experimental results show that the proposed algorithm has good performance, and the proposed framework adapts to strengthen the security of the IoT with heterogeneous and vulnerable devices.

INDEX TERMS Software-defined Internet of Things (SD-IoT), distributed denial of service (DDoS), attack detection, attack mitigation, cosine similarity.

I. INTRODUCTION

Internet of Things (IoT) is becoming more popular, and it can be seen in the home, vehicles and wearable devices. IoT involves a large number of interconnected devices, including household appliances, public facilities, wearable equipment, medical equipment, unmanned aerial vehicles, and interconnected vehicles as well as other applications that require networking [1]–[4]. In the next decade, tens of billions of devices with a variety of vulnerabilities will be connected to IoT. These networking devices have no user interface, no security protocol, and no computing and storage capacity to enable firewalls and diagnostic tools; moreover, they cannot directly connect to the Internet via WiFi. These vulnerabilities represent temptation not only for organizations that want to collect data to achieve intelligent management and digital evidence but also for those who want to disseminate DDoS attacks or other malicious intrusions. Once a DDoS attack is successful, it may threaten the safety of human life and even directly or indirectly cause death and destruction. In recent years, many examples have shown that IoT is vulnerable to viruses [5], [6]. Recent DDoS attacks have revealed that loopholes are ubiquitous in IoT, which is still in the initial stage. Without security precautions, the vast majority of IoT devices may unknowingly become accomplices to DDoS attacks.

The software-defined anything (SDx) paradigm addresses the above mentioned problems. SDx includes software-defined radio (SDR), software-defined networking (SDN), software-defined data centers (SDDC), software-defined infrastructure (SDI),and the software-defined world (SDW). SDN is undoubtedly the most recognized technology, and its core technology is the separation of the control plane and data plane in the network. It realizes flexible control of network traffic and provides a good platform for the innovation of core networks and applications. For example, Huo *et al.* [7] and Zhang *et al.* [8] focused on SDN for next-generation green wireless networks and future 5G wireless networks. Recently, we analyzed computational diversity in emerging networking paradigms, e.g., SDN, cloud radio access networks (C-RAN), and mobile cloud computing (MCC) [9], and proposed a box-covering-based routing algorithm for large-scale SDNs [10].

Recent studies indicate that SDN support for IoT is achieved through centralized control, abstraction of network devices and flexible, dynamic and automated reconfiguration of networks. The integration of SDN and IoT is a potentially feasible solution to strengthen the management and control capabilities of the IoT network [11]. For instance, the software-defined Internet of Things (SD-IoT) is favored in [12] and [13]. Liu *et al.* [14] proposed an SD-IoT architecture for smart urban sensing. Xiong *et al.* [11] proposed an SDN architecture for IoT and investigated the resource allocation problem in the proposed SD-IoT network. Kuang *et al.* [15] developed a transition tensor model for routing path recommendation by applying SDN technology to IoT for device management. A model of the IoT architecture that combines the benefits of SDN and fog computing has also been presented [16]. Sood *et al.* [17] highlighted recent significant developments in the wireless and optical domains with the aim of integrating SDN and IoT; in addition, they discussed challenges in SDN and IoT integration.

The key feature of SD-IoT is that it decouples network control and forwarding functions. The SD-IoT controller in the control layer is responsible for the logic centralized control of IoT. The SD-IoT switches, e.g., two-layer or three-layer switches, routers, base stations and wireless access points, only act as the data layer of IoT, and they only perform flow forwarding. Using a programming interface (such as REST [18]) provided by the SD-IoT controller, users can interact directly with IoT devices, configure the edge computing, analyze the environment and deploy security control. This decoupling avoids potential operation failure and service interruption, ensures the continued availability of IoT devices, prevents unauthorized access to peripheral devices, monitors and controls changes to the Internet of devices, detects legitimate and malicious traffic patterns on IoT devices, and ultimately reduces the risks managed by IoT security [19]–[21].

II. RELATED WORK

Recently, Bawany *et al.* [22] studied and proposed a framework for large-scale networks such as smart cities for DDoS attack detection and mitigation based on the SDN architecture. Lim *et al.* [23] proposed a scheduling-based architecture for the SDN controller that leads to effective attack confinement during DDoS attacks. Vizvary and Vykopal [24] summarized the execution, detection and mitigation of DDoS attacks in the SDN environment. Wu *et al.* [25] investigated a flow rule flooding DDoS attacks, and studied a novel DDoS attack targeting the data plane of SDN. Dayal *et al.* [26] summarized existing solutions to the DDoS problem of SDN and roughly divided them into three types: statistic-based approaches, rule-based approaches, and machine-learning approaches. Statistic-based approaches use statistical methods to analyze the SDN network traffic to achieve a distinction between normal flow and DDoS flow. Mousavi and St-Hilaire [27] proposed an entropy-based DDoS attack detection algorithm using the statistic-based approach. The algorithm calculates the entropy of the destination IP address in the packet-in message through the SDN controller and selects the threshold and sampling time. The DDoS attack can be considered detected when the value of the entropy is greater than the given threshold. It can detect DDoS attacks but cannot locate the specific IP address. Rule-based approaches extract the features of different DDoS attacks and then exchange these features at the flow table to avoid DDoS attacks [28]. The benefit of rule-based approaches is that they have relatively high accuracy, but the disadvantage is that features must be re-extracted when new attack modes emerge. Machine-learning approaches use the machine to learn, following by training according to the SDN in the flow characteristics of the sample to obtain the best learning model. Kokila *et al.* [29] used support-vector networks (SVM) to detect DDoS attacks. In [30], a DDoS attack detection algorithm of SDN devices based on IP filtering was proposed. For convenience, we call it the DPPC algorithm. The DPPC algorithm is simple to implement but cannot cope with the requests of large traffic. More importantly, the DPPC algorithm divides the DDoS packets based on IP address, which is affected by forged source IP addresses. A backtracking algorithm must be introduced to find a real DDoS source, which will increase the response time. Other works on DDoS attacks in SDN can been found in [31]–[33].

Previous findings indicate that SDN technology can help enterprises resist DDoS attacks, and Wang *et al.* [34] proposed a DDoS attack mitigation architecture that is suitable for a cloud computing environment. The architecture integrates a highly programmable network and a flexible control structure, where the programmable network is responsible for monitoring and detecting attacks, and the control structure is responsible for the rapid response to attacks. The current situation and progress of DDoS attack mitigation based on SDN in a cloud computing environment can be found in [35] and [36].

There is preliminary research on how to strengthen and enhance the security of the IoT with SDN technology. Flauzac *et al.* [19] proposed a secure and distributed architecture for IoT based on the SDN domain. Recently, Li *et al.* [20] investigated the potential threats of man-in-the-middle attacks on the OpenFlow control channel. Ahmed and Kim [21] conducted a preliminary discussion of the mitigation of DDoS attacks in IoT and attempted to use the SDN architecture to mitigate the sampling-based anomaly detection system constraints to collect traffic statistics in the SDN switches and achieve detection accuracy. Nguyen and Yoo [37] proposed a hybrid countermeasure to prevent link spoofing attacks in the SD-IoT controller. Ge *et al.* [38] developed two proactive defense mechanisms for SD-IoT with non-patchable vulnerabilities.

Research on integrating SDN and IoT networks is still in its infancy. Many problems remain open, e.g., network framework and security. In this paper, we study DDoS attack detection and mitigation in our proposed SD-IoT framework. The main contributions of this paper are as follows:

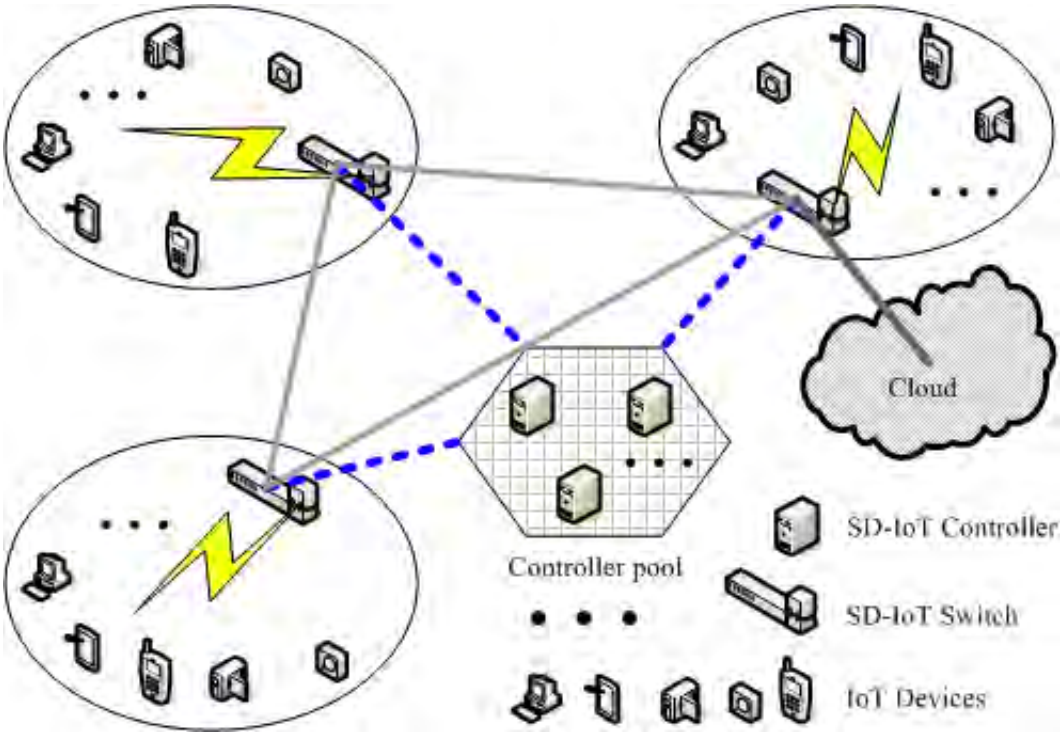

FIGURE 1. SD-IoT framework.

- A general framework for the SD-IoT is described, and the proposed framework consists of a controller pool, SD-IoT switches integrated with the IoT gateway, and IoT devices. The controller pool is designed as a vertical control structure, including the main control layer and basic control layer.
- An algorithm for detecting and mitigating DDoS attacks with the SD-IoT framework is proposed based on the cosine similarity of the vectors of the packet-in rate at the port of the boundary SD-IoT switches, and the threshold value of the cosine similarity and length of the vectors are obtained.
- The simulation results show that the proposed algorithm can find the IoT device from which a DDoS attack is launched within a shorter time period; the proposed approach quickly manages and mitigates the DDoS attack in IoT.

This paper is organized as follows. We summarize related work and the main contributions of this paper in Section II. In Section III, we introduce the general framework for SD-IoT, present the structure of the controller pool, and describe the problem of DDoS attacks in IoT. We propose an algorithm for detecting and mitigating DDoS attacks based on the proposed SD-IoT framework in Section IV. Section V presents the experimental settings and evaluates the performance. Finally, conclusions are given in Section VI.

III. DDoS ATTACKS IN THE SD-IoT ENVIRONMENT

The SDx paradigm uses software to control various types of hardware. SDN is one of the most extensive applications of the SDx paradigm and one of the most important areas in software-defined technology. The devices in SDN are programmable, and thus the networks themselves are more dynamic, manageable, cost-effective, and adaptable. SDN decouples the data plane and control plane. The primary components of SDN [39] are controllers and switches. The controllers in SDN are in charge of the management of the entire network, and the switches in SDN are responsible for network traffic forwarding based on the instructions deployed through the SDN controllers. SDN strives to be suitable to the changing traffic patterns, high-bandwidth, and dynamic nature of today's applications, and it is an emerging technology that can provide security defense solutions because it is capable of detecting attacks [39]. Recent studies indicate that SDN is regarded as a critical enabler of the security defense of IoT [19]–[21]. SDN can simplify IoT's plug-and-play predefined policies for network devices, automatically detect and remedy security vulnerabilities, configure edge computing and analyze the environment of the data flow.

A. SD-IoT FRAMEWORK

In this section, we describe a general framework for SD-IoT using the SDx paradigm, as illustrated in Fig.1. The proposed SD-IoT framework can be seen as an extended version of the SDN architecture applied to IoT, as well as a general type of IoT architecture based on the SDN as proposed in [13]. The framework can be divided into three layers: the application layer, control layer and infrastructure layer. The application layer consists of the IoT server in the cloud computing center, which is connected to controllers

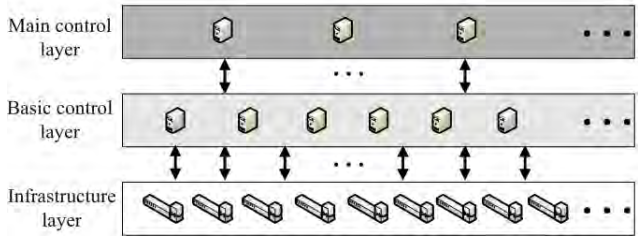

FIGURE 2. Structure of the controller pool.

in the SD-IoT controller pool, and the IoT server provides different applications and services via APIs. The control layer consists of a controller pool with many SD-IoT controllers, which run a distributed operating system that provides a logical centralized control and a topology view for IoT data forwarding in a distributed physical network environment. The infrastructure layer is composed of a mass of SD-IoT switches. Every SD-IoT switch integrates the function of an IoT gateway and an SDN switch, and each SD-IoT switch can access different IoT actuating devices and sensing devices, such as cameras, digital cameras, smart phones, and personal computers, by controlling the data plane interface. The IoT gateway in the proposed SD-IoT framework is integrated with the function of an SDN switch, whereas the IoT gateway in the framework mentioned by Hakiri *et al.* [13] is separate from the SDN switch. In addition, the proposed SD-IoT framework adopts the SD-IoT controller pool, and whereas the framework in [13] adopts a single controller.

B. STRUCTURE OF THE CONTROLLER POOL

In the above mentioned SD-IoT framework, the controller pool is designed as a vertical control structure, as illustrated in Fig.2. It is divided into two layers: the main control layer and basic control layer. The main control layer interacts with the application layer upwards and interacts with the basic control layer downwards, and the basic control layer interacts with the infrastructure layer.

Each controller of the main control layer, called the main controller, manages some base controllers, whereas the other basic controllers in the basic control layer are used as standby control objects. The main controllers are responsible for resource management, security and coordination of the basic control layer and also offer a northbound interface for the application layer services. In the main controllers, a Leader is elected by using the Paxos algorithm to solve the consensus problem. The Leader can obtain the global network topology information, control the main controllers and coordinate the basic controllers.

The basic controllers are responsible for resource management and security in a domain of IoT, as shown in Fig.1. IoT devices in the same domain can communicate through the basic controller of the domain. Each switch connects a master controller, and other controllers are known as slave controllers. Two IoT devices in different domains communicate through the main controller. The basic controllers

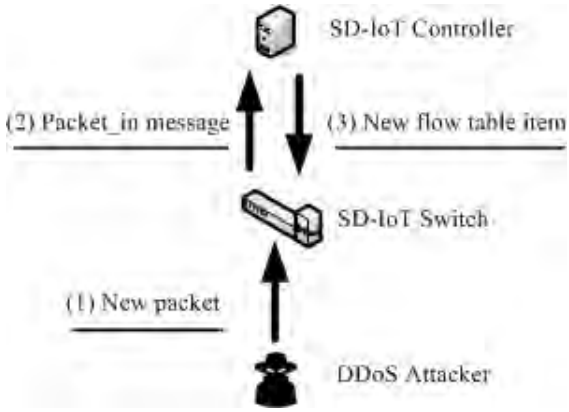

FIGURE 3. The process of DDoS attacks in SD-IoT.

communicate with the switches of the infrastructure layer, send packet-out messages to the switches at regular intervals, and obtain information on the switches through feedback packet-in messages. The basic controllers submit their own control information by interacting with the main controllers so that the main controllers can obtain the entire network global view. The load balance of the entire IoT can be improved through the coordination of messages in the main controllers and by using the dynamic load-balancing algorithm in the basic controllers [40].

The vertical structure of the controller pool is not only easy to manage but also solves a series of problems caused by the single point failure of the simple controller, e.g., incompatibility in the underlying controllers, load imbalance and message inconsistency between the main controllers.

C. DDoS ATTACKS IN SD-IoT

A DDoS attack is a type of attack in which devices are attacked from multiple sources in a distributed manner, creating a denial of service to users. SDN offers defense solutions for DDoS attacks through its programmability feature [39]. In the proposed SD-IoT framework, the SD-IoT controllers in the controller pool of IoT are responsible for the logic centralized control of the entire IoT environment. The logic centralized control is easy to manage and configure but also causes security issues. Similar to SDN, SD-IoT is capable of offering constructive schemes for detecting and mitigating DDoS attacks through the programmability of the proposed SD-IoT framework. In the following, we will analyze the process of DDoS attacks in SD-IoT, as illustrated in Fig. 3.

1) The DDoS attacker sends a new packet to a certain SD-IoT switch, where the attack packet is generated by the attack script.
2) If there is no corresponding match in the flow table items of the SD-IoT switch, the SD-IoT switch will encapsulate the header of the packet into a packet-in message and send it to one controller of the controller pool in SD-IoT.

3) The SD-IoT controller unpacks the packet-in message and generates a new flow table item, which is issued to the SD-IoT switch.

When a DDoS attacker generates a large number of packets without matching, the SD-IoT controller in the controller pool receives a large number of packet-in messages, which not only occupy the network resources between the SD-IoT controller and SD-IoT switch but also consume the CPU, memory and other resources of the SD-IoT controller, resulting in increased delay and even downtime. In addition, new flow table items are continually added to the SD-IoT switches, and the SD-IoT switches cannot continue to manage the new incoming packets when the number of flow table items in the SD-IoT switches exceeds the maximum. As a result, the SD-IoT switches cannot work properly. This process is similar to [41].

IV. DDoS ATTACK DETECTION AND MITIGATION ALGORITHM

A. PROBLEM DESCRIPTION

In the proposed SD-IoT framework, when an SD-IoT device is attacked by a DDoS attacker, the attacker generates and sends data packets consisting of randomly forged sources and random destination addresses via script. To perform DDoS attack detection and mitigation of SD-IoT devices, the proposed algorithm calculates the cosine similarity of the vectors of the packet-in rate for each port of the boundary switches in the SD-IoT and then determines whether a DDoS attack has occurred based on the value of the cosine similarity. First, most of the packets sent by a DDoS attack are generated through a programming method, and thus the rate at which an attacker sends a DDoS packet can be considered stable and relatively smooth in milliseconds. Then, when one of the data packets in the SD-IoT switch has no corresponding match to flow table items, the header of the data packet is encapsulated into a packet-in message and sent to the SD-IoT controller, and the packet-in message records which port of which SD-IoT switch the message was sent from. In the physical network, each boundary switch in SD-IoT is connected to the only host, and thus the boundary switch is not affected by the fake source IP address and destination IP address when finding the real DDoS attack source. We will design the algorithm for detecting and mitigating DDoS attacks with the proposed SD-IoT framework in the next subsection.

B. ALGORITHM DESIGN

The key of the proposed algorithm is the introduction of cosine similarity for measuring the similarity between the vectors with respect to the packet-in rate of the input port of the boundary SD-IoT switches. In this subsection, the main steps of the proposed algorithm for detecting and mitigating DDoS attacks are as follows:

1) OBTAINING THE VECTORS X AND Y OF THE INPUT PORT RATE λ_m

At a time interval Δt, we first obtain the packet-in rate λ_m ($m = 1, 2, \ldots$) of the input port α of the boundary SD-IoT switch, and then we obtain the set $X = \{X_1, X_2, \ldots, X_n\} = \{\lambda_2, \lambda_4, \lambda_6, \ldots, \lambda_{2k}\}$, $Y = \{Y_1, Y_2, \ldots, Y_n\} = \{\lambda_1, \lambda_3, \lambda_5, \ldots, \lambda_{2k-1}\}$, where k represents the length of the sets or vectors X and Y. In the following subsection, we will discuss the selection of the k value in detail.

2) CALCULATING THE COSINE SIMILARITY $\rho_{X,Y}$ OF THE VECTORS X AND Y

Cosine similarity is a measure of similarity between two vectors. Cosine similarity is very efficient, particularly for sparse vectors, because only the non-zero dimensions need to be considered. We can obtain the cosine similarity $\rho_{X,Y}$ of the vectors of the packet-in rate of the port α by the following equation (1):

$$\begin{aligned}\rho_{X,Y} &= \cos(\theta) \\ &= \frac{X \bullet Y}{||X|| \times ||Y||} \\ &= \frac{\sum_{i=1}^{n}(X_i \times Y_i)}{\sqrt{\sum_{i=1}^{n} X_i^2} \times \sqrt{\sum_{i=1}^{n} Y_i^2}} \\ &= \frac{\sum_{i=2m,j=2m-1}^{2k}(\lambda_i \times \lambda_j)}{\sqrt{\sum_{i=2m}^{2k} \lambda_i^2} \times \sqrt{\sum_{j=2m-1}^{2k-1} \lambda_j^2}}\end{aligned} \tag{1}$$

where $m = 1, 2, \ldots, k$. Obviously, all elements of the two vectors X and Y are greater than or equal to 0, and thus we have $0 \leq \rho_{X,Y} \leq 1$. If the cosine similarity $\rho_{X,Y}$ is closer to 1, the closer the angle of the two vectors is, the closer the two vectors X and Y are. If the cosine similarity $\rho_{X,Y}$ is equal to 1, the similarity between vectors X and Y is highest.

3) DETERMINING WHETHER A DDoS ATTACK OCCURS

We assume that the threshold value of the cosine similarity for distinguishing a DDoS attack flow and a normal flow is η_U. If $\eta_U \leq \rho_{X,Y} \leq 1$, the input port of the SD-IoT switch may have a DDoS attack. If $0 \leq \rho_{X,Y} < \eta_U$, the data packet of the port is likely to be a normal request. The selection of the threshold η_U of the cosine similarity is crucial for the performance of the proposed algorithm. In the next subsection, we discuss the threshold η_U of the cosine similarity in detail.

4) DISCARDING THE DDoS ATTACK PACKET

To improve the reliability and accuracy of the results, multiple ρ values must be sampled to distinguish the DDoS attack flow and normal flow. We obtain the set $P = \{\rho_1, \rho_2, \ldots, \rho_l\}$, where l is the number of the sample ρ. We use *sum* to denote the number of samples ρ that meet the condition $\eta_U \leq \rho \leq 1$ in the set P; then, we can determine which port is attacked by the DDoS attacker according to the value of *sum* and discard the specific type of data packet at the port. The pseudocode of the proposed algorithm is given in Algorithm 1.

C. PARAMETER SETTING

The threshold η_U of the cosine similarity and length k of the vectors X and Y are two important parameters of the

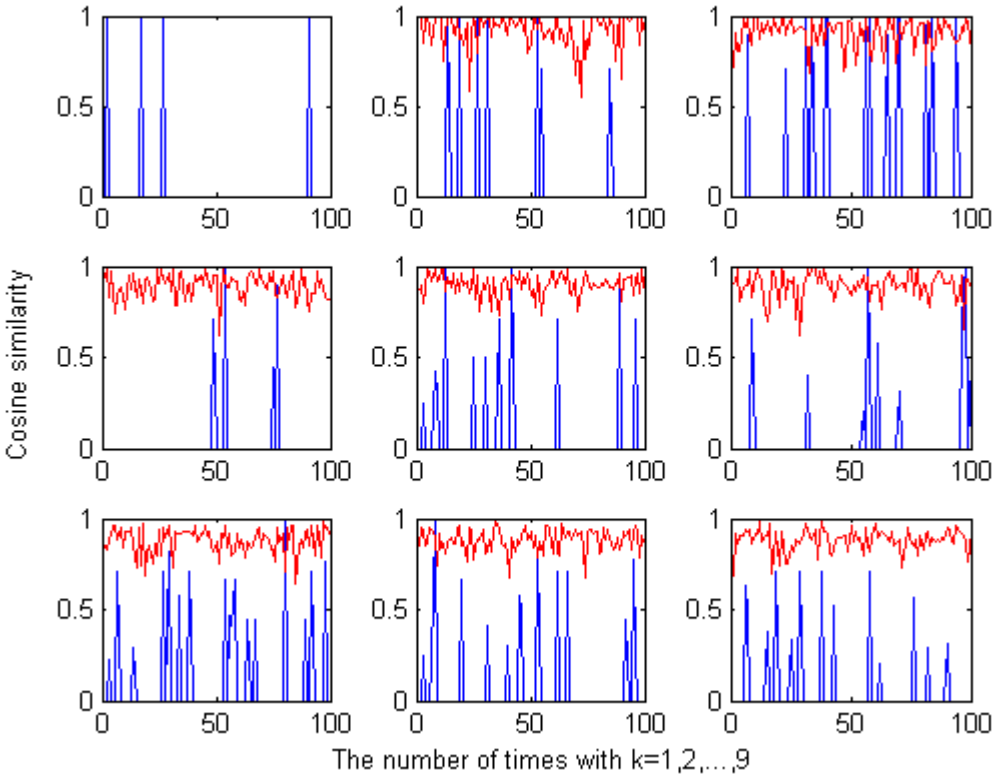

FIGURE 4. The cosine similarity of the DDoS attack flow (red curve) and normal flow (blue curve).

proposed algorithm. If η_U is too high, many DDoS attack packets will be determined as legal requests. If η_U is too low, many normal packets will be considered DDoS attacks. If the value of k is greater, the detection time is longer. This can lower the performance of the IoT and even result in a breakdown of the SD-IoT controllers and switches. If the value of k is too small, it can reduce the accuracy of the detection results and ultimately lead to a high error rate. To obtain the best possible value of the cosine similarity η_U and length k, we will perform a large number of experiments and analyses. The experimental environment is detailed in Section V.

1) THE THRESHOLD η_U OF THE COSINE SIMILARITY

In the experiment where the values of η_U and k are selected, the time interval t_0 for sending DDoS attack packets is random and ranges from 0.005s to 0.05s, and the step size Δt_0 is 0.005s. The time interval t_1 for sending packets of normal flow is in the range of 0.05s to 1s, and the step length Δt_1 is 0.05s. When the value of k increases from 1 to 9, we can obtain the influence of different k values on the cosine similarity. For each k value, we continuously obtain 100 values for cosine similarity, and we can then separately calculate the mean and standard deviation of the cosine similarity. The cosine similarities of the DDoS attack flow (red curve) and normal flow (blue curve) are shown in Fig.4. According to the statistical analysis of the data, k changes from 1 to 9, and the

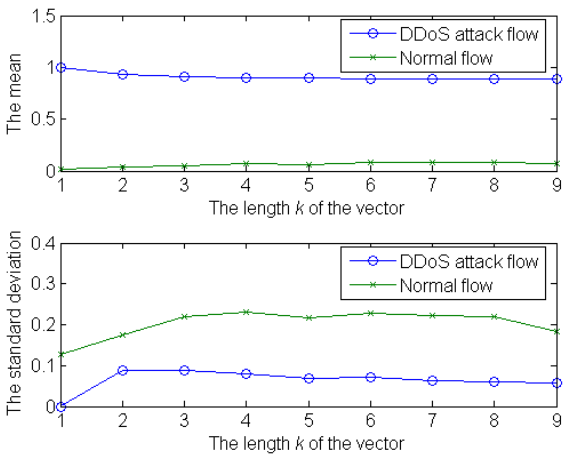

FIGURE 5. The mean and standard deviation of the cosine similarity under different length k of the vectors.

value of the cosine similarity is greater than or equal to 0.7. Therefore, we select 0.7 as the value of the threshold η_U in the next experiment.

2) THE LENGTH K OF THE VECTORS

The mean and standard deviation of the cosine similarity between the DDoS attack flow and normal flow are shown in Fig.5. When the value of the length k of the vectors X and Y changes from 2 to 9, the mean of the cosine similarity

Algorithm 1 DDoS Attack Detection and Mitigation Algorithm for SD-IoT

Input: The set *edgeSW_set* of all boundary SD-IoT switches, the number *sum* of samples ρ meeting the condition, the length k of the vectors X and Y, the time interval Δt, the number of packet-in messages *pkin*, and the threshold η_U of the cosine similarity.

Output: The port *ddosPort* carrying out the DDoS attack and being detected.

```
if α ∈ edgeSW_set then
  j = 1
  m = 1
  s = 1
  while m ≤ sum do
    while j ≤ 2k do
      for each Δt do
        count = the number of pkin
        λ_j = count/Δt
      end for
    end while
    X = {λ_i}, where i = 2, 4, 6, ..., 2k
    Y = {λ_j}, where j = 1, 3, 5, ..., 2k − 1
    Calculate ρ_{X,Y} by Equ.(1)
    if η_U ≤ ρ_{X,Y} ≤ 1 then
      s = s + 1
    end if
  end while
  if s = sum then
    return α
  end if
end if
```

of the DDoS attack flow is basically flat, and the change is not obvious. A significant change occurs when the value of k is between 1 and 2; when $k = 1$, each of the two nonzero vectors X and Y has only one nonzero element. Regardless of the value of the nonzero element, the value of the cosine similarity is equal to 1, as calculated by the Equ.(1). In addition, the mean of the cosine similarity of the normal flow increases with the increase in the value of k. In general, all of the changes are very small. When $k = 1$, the standard deviation of the cosine similarity is the smallest regardless of whether it is normal flow or DDoS attack flow. When $k = 2$, the standard deviation of the cosine similarity of the DDoS attack flow is maximum. The standard deviation of the cosine similarity of the DDoS attack flow drops to the lowest point at $k = 5$. There is a slight increase after $k = 6$ followed by another decreasing trend until $k = 9$, when the standard deviation of the cosine similarity of the DDoS attack flow attains the minimum. Here, we need to select the most appropriate value of the length k of the vectors X and Y. That is, we need to find the minimum standard deviation of the cosine similarity of the DDoS attack flow. The standard deviation of the cosine similarity of the DDoS attack flow is similar from $k = 6$ to $k = 9$ than for $k = 5$, but the time required for continuous sampling of 5 times ρ is longer than that required when $k = 5$. In the case of $k = 1$, the standard deviation of the cosine similarity of the DDoS attack flow is smallest, but the value of k is too small, and there is a large probability that continuous 5 times ρ is greater than the threshold η_U for normal flow. Obviously, when $k = 5$, the standard deviation of the cosine similarity of the DDoS attack flow is small, and the amount of time consumed for detection is reasonable. A complete detection takes only 5s. Without loss of generality, we set the parameter k to 5 in the follow-up experiment.

V. PERFORMANCE EVALUATION

In this section, we first configure the experimental environment, and then elaborate the experimental method. Finally, we evaluate the performance of the proposed algorithm.

A. EXPERIMENTAL SETTINGS

The hardware of the experimental environment for evaluating the performance of the proposed algorithm consists of two computers. One of them acts as a controller, and we configure its settings as follows: i5 2.7GHz processor, 8G DDr3 1866GHz memory, 256GB PCIe-based flash storage, 7200RPM 1T hard disk drive and killer E2200 gigabit ethernet adapter. The other computer is used to simulate different network topologies in IoT and is configured with an i7 2.5GHz processor and 16G DDr3 memory; all other configurations are the same as above. The network bandwidth is 100Mbp/s. All software is deployed in the VirtualBox virtual machine platform of the Windows 10 system, and Ubuntu 16.0.4 LTS is installed in the virtual machine. The essential tools of the experimental platform include software-defined controllers, software-defined switches and DDoS attack software. In particular, Floodlight deployed on the OS X platform is used as the SD-IoT controller; Open vSwitch is used as the SD-IoT switch; Mininet deployed on Ubuntu 16.0.4 LTS is used to simulate the network topology of the SD-IoT, and its API interface is used in development and testing; these software products support OpenFlow 1.4. In addition, the DDoS attack flow and normal flow are both generated through the Scapy library of the python script. The Scapy library is an interactive and powerful program that developers can use to fake or parse the contents of the data package and current network protocol. The Scapy library supports real-time capture and generation of data packets. The python script can simulate complex network operations through the Scapy library and also control the number and rate of sending packets. In our experiments, the forged source IP address, the random destination IP address and destination IP address of the UDP packets are all generated and sent through the Scapy library of the python script.

An instance of the SD-IoT topology generated based on Mininet, is used to validate the proposed algorithm, as

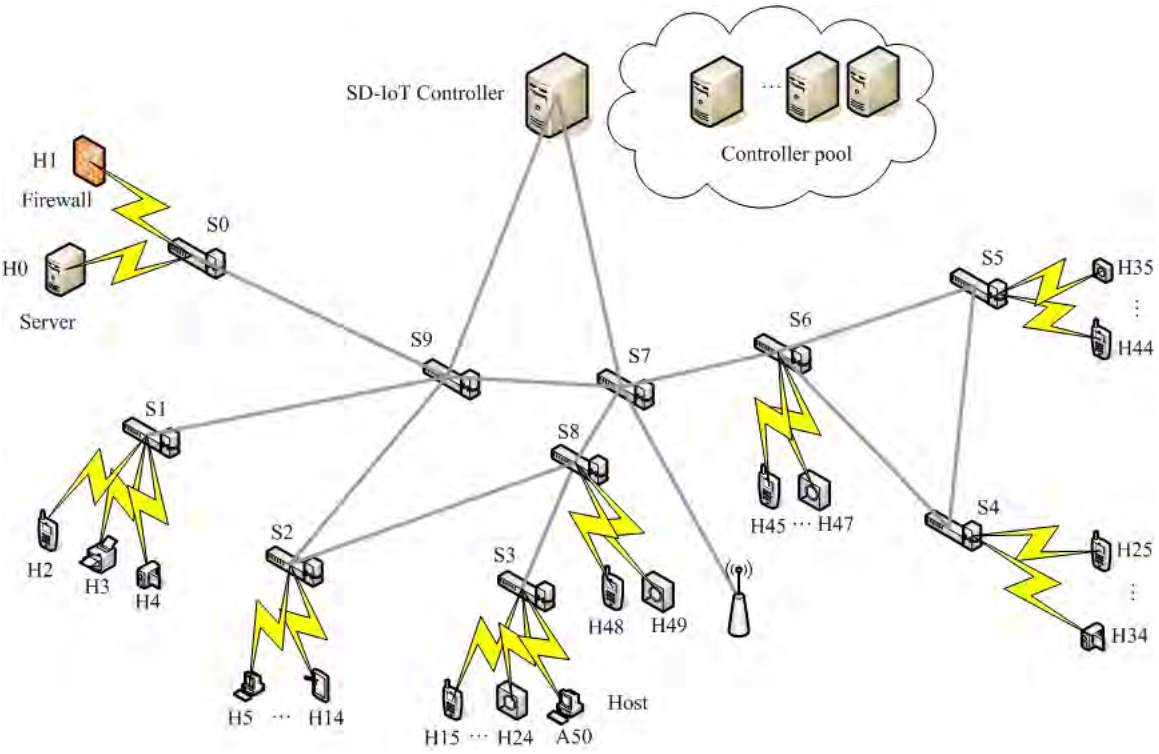

FIGURE 6. An instance of DDoS attack detection and mitigation for SD-IoT.

TABLE 1. List of experimental parameters.

Parameter	Description	Value
k	The length of the vectors X and Y	5
η_U	The threshold of the cosine similarity $\rho_{X,Y}$	0.7
sum	The number of the sample $\rho \geq \rho_{X,Y}$	5
Δt	The time interval for obtaining the packet-in rate of the port α	0.1s
Δt_1	The time interval for sending DDoS attack flow	0.05s
Δt_2	The time interval for sending normal flow	0.005s
$idle\_timeout$	a parameter in the flow table item for normal flow	20s
	a parameter in the flow table item for DDoS attack flow	300s
$hard\_timeout$	a parameter in the flow table item for normal flow	600s
	a parameter in the flow table item for DDoS attack flow	600s

illustrated in Fig.6. We can determine a controller from the controller pool to manage a domain by using the method in [40]. Without loss of generality, we randomly choose one Floodlight from the controller pool as the current SD-IoT controller and use ten Open vSwitches as SD-IoT switches, which are referred to as S0, S1, …, and S9. These are 50 terminal devices of the IoT, and they are referred to as H0, H1, …, and H49. A50 is a host. The experimental network topology is shown in Fig.6. We deploy python script in terminal device H15 and host A50 and take terminal device H15 as a frequent-accessing legitimate user who randomly sends requests to terminal devices H0, H1, …, and H49. We take host A50 as a DDoS attacker, who randomly sends data packets with the forged source IP address, the source MAC address and destination IP address. The experimental parameters are summarized in Tab.1.

B. EXPERIMENTAL METHOD

The experimental method for detecting and mitigating DDoS attacks based on the proposed SD-IoT framework is as follows:

1) CONFIGURING THE SD-IoT SWITCHES

First, all flow table items in SD-IoT switches S0, S2, and S9 are cleared. Then the time period t_1 for sending the packet-in message of the normal flow from normal user H15 is set to random. For example, the value of t_1 is from 0.05s to 10s, and the time interval Δt_1 is 0.05s. Finally, the destination IP address of one of the terminal devices H0, H2, …, and H49. The time period t_2 of the packet-in message sent by the DDoS attacker is set from 0.005s to 0.035s, and the time interval Δt_2 is 0.005s.

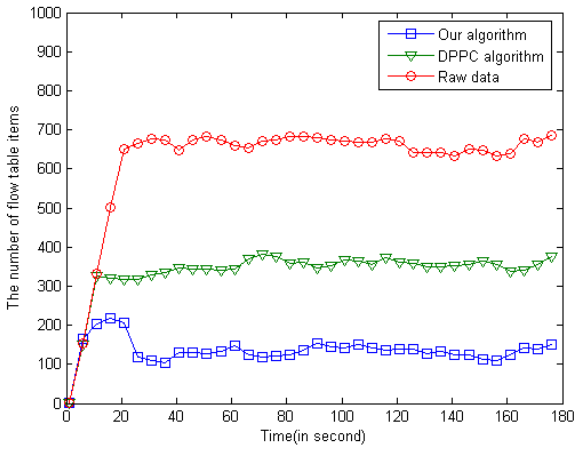

FIGURE 7. The number of flow table items in SD-IoT switch S3.

2) DEPLOYING THE PROPOSED ALGORITHM FOR DETECTING AND MITIGATING DDoS ATTACKS ON THE SD-IoT CONTROLLER

For the proposed algorithm, we first set the parameters *idle_timeout* and *hard_timeout* of the flow table items of the normal flow in the SD-IoT controller to 20s and 600s, respectively. Once the DDoS attack is detected, *idle_timeout* and *hard_timeout* are set to 300s and 600s, respectively. Then, host A50 sends a DDoS attack packet, and terminal device H15 sends the normal packet so that the SD-IoT controller generates a flow table item for each packet; the entire process lasts about 180s. We can use the same approach for the DPPC algorithm in [30]; *idle_timeout* and *hard_timeout* from flow table items of the SD-IoT controller are set to 10s and 60s, respectively. Once the DDoS attack is detected, we set *idle_timeout* and *hard_timeout* to 60s and 600s, respectively.

3) OBTAINING THE STATISTICS DATA

For the three types of conditions - our algorithm, the DPPC algorithm and the raw data - the SD-IoT controller separately counts the number of flow table items of switch S3, the number of packet-in messages sent by SD-IoT switch S3 to the SD-IoT controller, and the number of data packages that the SD-IoT controller receives.

C. ANALYSIS OF THE RESULTS

In the following subsection, we analyze the number of flow table items of the SD-IoT switches, the number of packet-in messages sent by the SD-IoT switches to the SD-IoT controller, and the number of data packets received by the SD-IoT controller under the two algorithms and the raw data.

The impact of time on the number of flow table items in SD-IoT switch S3 is shown in Fig.7. Three curves are very steep and have the highest slope in the beginning stages. For our algorithm, the DPPC algorithm and the raw data, the number of flow table items increases from zero to 218, 347, and 677, respectively. The curve of the raw data that does not deploy a detection or mitigation algorithm is basically stabilized after 20s, as we set *idle_timeout* to 20s, and flow table items are automatically removed when the time exceeds *idle_timeout*. Therefore, the flow table items of SD-IoT switch S3 change slightly rather than continuing to increase significantly, and the overall trend is relatively stable. Similarly, for the results obtained by using the DPPC algorithm in [30], its curve is also stable after 10s since *idle_timeout* is set to 10s, but the number of flow table items does not decrease after 20s. Therefore, the DPPC algorithm has no practical effect in our experimental environment; that is, no DDoS attacks are detected. The curve of the results obtained by using our algorithm shows that the increasing trend of flow table items within the time frame of 5s to 10s is smaller than that from 0s to 5s. With our algorithm, the packet is discarded if a DDoS attack is detected after 5s, and thus SD-IoT switch S3 does not send the packet-in message to the SD-IoT controller. However, since the SD-IoT controller has a buffer for packet-in messages in memory, it continues to process the packet-in message and send flow table items to SD-IoT switch S3, but the number of flow table items is less than it was between 0s and 5s. The curve of flow table items in SD-IoT switch S3 is basically flat between 10s and 20s; SD-IoT switch S3 discards the DDoS attack packets, the SD-IoT controller does not receive a large number of packet-in messages generated by DDoS attacks, and flow table items are not issued to SD-IoT switch S3. After 20s, the curve begins to decline rapidly; we set *idle_timeout* to 20s. Flow table items added between 0s and 20s are overtime after 20s, and SD-IoT switch S3 automatically removes them. The number of flow table items from SD-IoT switch S3 continues to decrease and level off until SD-IoT switch S3 only contains flow table items of the normal flow.

The impact of time on the number of packet-in messages sent by SD-IoT switch S3 to the SD-IoT controller is given in Fig.8. The number of packet-in messages sent by SD-IoT switch S3 to the SD-IoT controller increases with time. The three curves indicate a slower rate of growth for our algorithm than that of the DPPC algorithm and the raw data. The rate of growth for our algorithm is only 12.09 percent of that of the raw data and 12.77 percent of that of the DPPC algorithm. Our approach adds rules to flow table items and discards this packet using our algorithm when a DDoS attack packet is sent to a port. Therefore, SD-IoT switch S3 no longer needs to send a large number of packet-in messages to the SD-IoT controller. For the raw data or the DPPC algorithm in [30], SD-IoT switch S3 still needs to send a large number of packet-in messages to the SD-IoT controller, and therefore the slope of the curves is very steep.

The impact of time on the number of data packets received by the SD-IoT controller is given in Fig.9. The varying tendencies of the rate of growth are all consistent with the rate of growth of the packet-in messages sent by SD-IoT switch S3 to the SD-IoT controller, as shown in Fig.8. The rate of growth for our algorithm is only 12.55 percent of that of the raw data and 13.83 percent of that of the DPPC algorithm.

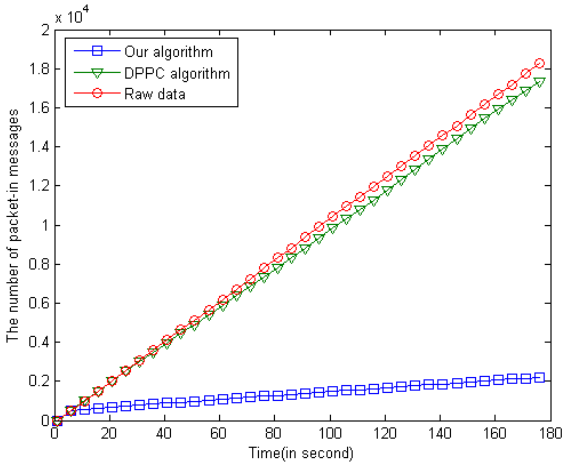

FIGURE 8. The number of packet-in messages sent by SD-IoT switch S3 to the SD-IoT controller.

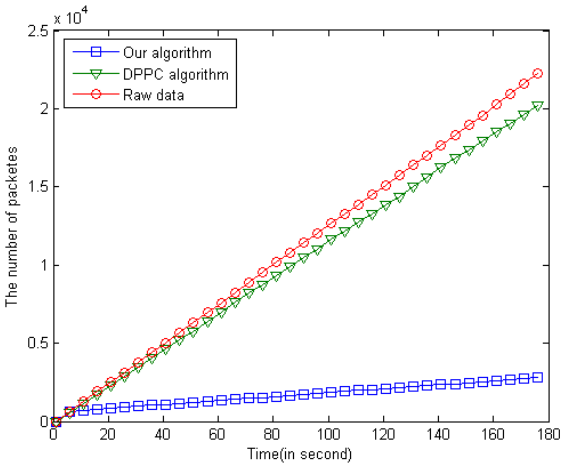

FIGURE 9. The number of data packets received by the SD-IoT controller.

Compared to the number of packet-in messages in Fig.8, there are larger rates of growth for data packets received by the SD-IoT controller in Fig.9. The negligible difference is caused by the forwarding rules and centralized control scheme in SD-IoT.

The impact of time on the bandwidth of controller-switch channels in SD-IoT is shown in Fig.10. For our algorithm, the bandwidth of the controller-switch channels in SD-IoT increases rapidly from 0s and 5s. SD-IoT switch S3 discards DDoS attack packets when our algorithm deployed in the SD-IoT controller detects DDoS attacks in 5s. Therefore, SD-IoT switch S3 does not have to send a large number of packet-in messages to the SD-IoT controller. The value of data packets received by the SD-IoT controller gradually drops to the lowest level until it is stable. The minor fluctuations are a result of the normal flow not being blocked; SD-IoT switch S3 still sends packet-in messages to the SD-IoT controller. For the raw data, the value of data packets received by the SD-IoT controller reaches the maximum

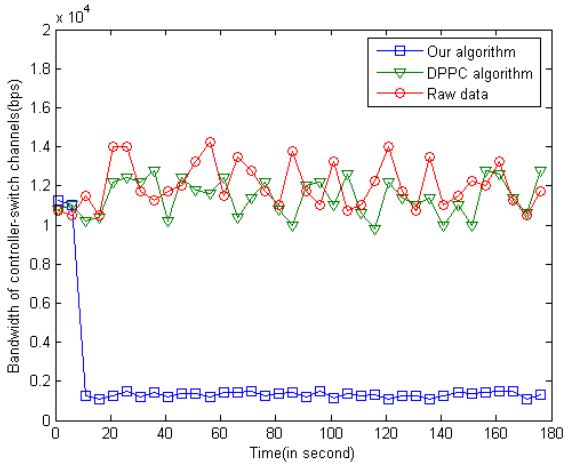

FIGURE 10. The bandwidth of controller-switch channels in SD-IoT.

at 10s and then gradually becomes stable. The ups and downs that follow occur because the DDoS attack flow and normal flow both generate packet-in messages. For the DPPC algorithm of [30], the value of data packets received by the SD-IoT controller becomes stable after 20s and is high; this is similar to the raw data after 20s. Obviously, the algorithm does not successfully detect DDoS attacks.

To summarize, the number of flow table items of the SD-IoT switches, the number of packet-in messages from the SD-IoT switches to the SD-IoT controller, and the number of data packets received by the SD-IoT controller are smaller than those of other approaches when using our algorithm. The results indicate that our algorithm improves the performance of IoT under DDoS attacks.

VI. CONCLUSIONS

In this paper, we describe a general framework for SD-IoT composed of an SD-IoT controller pool with controllers, SD-IoT switches integrated with the IoT gateway, and terminal IoT devices. Then, we propose an algorithm for detecting and mitigating DDoS attacks with the proposed SD-IoT framework. In the proposed algorithm, we obtain the threshold value of the cosine similarity of the vectors of the packet-in rate at the ports of the SD-IoT boundary switches; we use the threshold value to determine whether a DDoS attack has occurred, find the real DDoS attacker, and block the DDoS attack at the source. Finally, the simulation results show that the proposed algorithm can find the IoT device from which a DDoS attack is launched within a shorter time period, quickly handle and mitigate the DDoS attack, and ultimately improve the unveiled glaring vulnerabilities in IoT, in which the terminal devices have computational and memory requirement constraints. Future work will focus on how to proactively defend against DDoS attacks in SD-IoT. In addition, dynamic load-balancing algorithms in the controller pool will be designed and implemented, and more efficient algorithms for detecting and mitigating DDoS attacks based on the SD-IoT framework will be investigated.

REFERENCES

[1] H. Ma, L. Liu, A. Zhou, and D. Zhao, "On networking of Internet of Things: Explorations and challenges," *IEEE Internet Things J.*, vol. 3, no. 4, pp. 441–452, Aug. 2016.

[2] P. G. Neumann, "Risks of automation: A cautionary total-system perspective of our cyberfuture," *Commun. ACM*, vol. 59, no. 10, pp. 26–30, Oct. 2016.

[3] X. Liu, S. Zhao, A. Liu, N. Xiong, and A. V. Vasilakos, "Knowledge-aware proactive nodes selection approach for energy management in Internet of Things," *Future Generat. Comput. Syst.*, Aug. 2017, doi: https://doi.org/10.1016/j.future.2017.07.022

[4] Y. Liu, A. Liu, S. Guo, Z. Li, Y.-J. Choi, and H. Sekiya, "Context-aware collect data with energy efficient in cyber–physical cloud systems," *Future Generat. Comput. Syst.*, Jun. 2017, doi: https://doi.org/10.1016/j.future.2017.05.029

[5] K. Sonar and H. Upadhyay, "A survey: DDoS attack on Internet of Things," *Int. J. Eng. Res. Develop.*, vol. 10, no. 11, pp. 58–63, Nov. 2014.

[6] U. Lindqvist and P. G. Neumann, "The future of the Internet of Things," *Commun. ACM*, vol. 60, no. 2, pp. 26–30, Jan. 2017.

[7] R. Huo *et al.*, "Software defined networking, caching, and computing for green wireless networks," *IEEE Commun. Mag.*, vol. 54, no. 11, pp. 185–193, Nov. 2016.

[8] J. Zhang, X. Zhang, M. A. Imran, B. Evans, Y. Zhang, and W. Wang, "Energy efficient hybrid satellite terrestrial 5G networks with software defined features," *J. Commun. Netw.*, vol. 19, no. 2, pp. 147–161, Apr. 2017.

[9] K. Wang, K. Yang, H.-H. Chen, and L. Zhang, "Computation diversity in emerging networking paradigms," *IEEE Wireless Commun.*, vol. 24, no. 1, pp. 88–94, Feb. 2017.

[10] L. Zhang, Q. Deng, Y. Su, and Y. Hu, "A box-covering-based routing algorithm for large-scale SDNs," *IEEE Access*, vol. 5, no. 1, pp. 4048–4056, Mar. 2017.

[11] X. Xiong, L. Hou, K. Zheng, W. Xiang, M. S. Hossain, and S. M. M. Rahman, "SMDP-based radio resource allocation scheme in software-defined Internet of Things networks," *IEEE Sensors J.*, vol. 16, no. 20, pp. 7304–7314, Oct. 2016.

[12] Z. Qin, G. Denker, C. Giannelli, P. Bellavista, and N. Venkatasubramanian, "A software defined networking architecture for the Internet-of-Things," in *Proc. IEEE Netw. Oper. Manage. Symp. (NOMS)*, May 2014, pp. 1–9.

[13] A. Hakiri, P. Berthou, A. Gokhale, and S. Abdellatif, "Publish/subscribe-enabled software defined networking for efficient and scalable IoT communications," *IEEE Commun. Mag.*, vol. 53, no. 9, pp. 48–54, Sep. 2015.

[14] J. Liu, Y. Li, M. Chen, W. Dong, and D. Jin, "Software-defined Internet of Things for smart urban sensing," *IEEE Commun. Mag.*, vol. 53, no. 8, pp. 55–63, Sep. 2015.

[15] L. Kuang, L. T. Yang, and K. Qiu, "Tensor-based software-defined Internet of Things," *IEEE Wireless Commun.*, vol. 23, no. 5, pp. 84–89, Oct. 2016.

[16] S. Tomovic, K. Yoshigoe, I. Maljevic, and I. Radusinovic, "Software-defined fog network architecture for IoT," *Wireless Pers. Commun.*, vol. 92, no. 1, pp. 181–196, Jan. 2017.

[17] K. Sood, S. Yu, and Y. Xiang, "Software-defined wireless networking opportunities and challenges for Internet-of-Things: A review," *IEEE Internet Things J.*, vol. 3, no. 4, pp. 453–463, Aug. 2016.

[18] F. Belqasmi, R. Glitho, and C. Fu, "RESTful Web services for service provisioning in next-generation networks: A survey," *IEEE Commun. Mag.*, vol. 49, no. 12, pp. 66–73, Dec. 2011.

[19] O. Flauzac, C. Gonzalez, A. Hachani, and F. Nolot, "SDN based architecture for IoT and improvement of the security," in *Proc. IEEE 29th Int. Conf. Adv. Inf. Netw. Appl. Workshops*, Mar. 2015, pp. 688–693.

[20] C. Li, Z. Qin, E. Novak, and Q. Li, "Securing SDN infrastructure of IoT–fog network from MitM attacks," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1156–1164, Mar. 2017.

[21] M. E. Ahmed and H. Kim, "DDoS attack mitigation in Internet of Things using software defined networking," in *Proc. IEEE 3rd Int. Conf. Big Data Comput. Service Appl.*, Apr. 2017, pp. 271–276.

[22] N. Z. Bawany, J. A. Shamsi, and K. Salah, "DDoS attack detection and mitigation using SDN: Methods, practices, and solutions," *Arabian J. Sci. Eng.*, vol. 42, no. 2, pp. 425–441, Feb. 2017.

[23] S. Lim, S. Yang, Y. Kim, and S. Yang, "Controller scheduling for continued SDN operation under DDoS attacks," *Electron. Lett.*, vol. 51, no. 16, pp. 1259–1261, Aug. 2015.

[24] M. Vizvary and J. Vykopal, "Future of DDoS attacks mitigation in software defined networks," in *Proc. IFIP Int. Conf. Auto. Infrastruct., Manage. Secur.*, Jul. 2014, pp. 123–127.

[25] X. Wu, M. Liu, W. Dou, and S. Yu, "DDoS attacks on data plane of software-defined network: Are they possible?" *Secur. Commun. Netw.*, vol. 9, no. 18, pp. 5444–5459, Dec. 2016.

[26] N. Dayal, P. Maity, S. Srivastava, and R. Khondoker, "Research trends in security and DDoS in SDN," *Secur. Commun. Netw.*, vol. 9, no. 18, pp. 6386–6411, Feb. 2016.

[27] S. M. Mousavi and M. St-Hilaire, "Early detection of DDoS attacks against SDN controllers," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, Feb. 2015, pp. 77–81.

[28] L. Wei and C. Fung, "FlowRanger: A request prioritizing algorithm for controller DoS attacks in software defined networks," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2015, pp. 5254–5259.

[29] R. T. Kokila, S. T. Selvi, and K. Govindarajan, "DDoS detection and analysis in SDN-based environment using support vector machine classifier," in *Proc. 6th Int. Conf. Adv. Comput.*, Dec. 2015, pp. 205–210.

[30] N.-N. Dao, J. Park, M. Park, and S. Cho, "A feasible method to combat against DDoS attack in SDN network," in *Proc. Int. Conf. IEEE Inf. Netw. (ICOIN)*, Aug. 2015, pp. 309–311.

[31] Q. Yan, Q. Gong, and F. R. Yu, "Effective software-defined networking controller scheduling method to mitigate DDoS attacks," *Electron. Lett.*, vol. 53, no. 7, pp. 469–471, Mar. 2017.

[32] R. Wang, Z. Zhang, L. Ju, and Z. Jia, "A novel OpenFlow-based DDoS flooding attack detection and response mechanism in software-defined networking," *Int. J. Inf. Secur. Privacy*, vol. 9, no. 3, pp. 21–40, Jul. 2015.

[33] X. Wang, M. Chen, X. Wei, and G. Zhang, "Defending DDoS attacks in software defined networking based on improved Shiryaev–Roberts detection algorithm," *J. High Speed Netw.*, vol. 21, no. 4, pp. 285–298, Nov. 2015.

[34] B. Wang, Y. Zheng, W. Lou, and Y. T. Hou, "DDoS attack protection in the era of cloud computing and software-defined networking," *Comput. Netw.*, vol. 81, pp. 308–319, Mar. 2015.

[35] Q. Yan and F. Yu, "Distributed denial of service attacks in software-defined networking with cloud computing," *IEEE Commun. Mag.*, vol. 53, no. 4, pp. 52–59, Apr. 2015.

[36] Q. Yan, F. R. Yu, Q. Gong, and J. Li, "Software-defined networking (SDN) and distributed denial of service (DDoS) attacks in cloud computing environments: A survey, some research issues, and challenges," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 602–622, 1st Quart., 2016.

[37] T. H. Nguyen and M. Yoo, "A hybrid prevention method for eavesdropping attack by link spoofing in software-defined Internet of Things controllers," *Int. J. Distrib. Sensor Netw.*, vol. 13, no. 11, pp. 1–9, Nov. 2017.

[38] M. Ge, J. B. Hong, S. E. Yusuf, and S. K. Dong, "Proactive defense mechanisms for the software-defined Internet of Things with non-patchable vulnerabilities," *Future Generat. Comput. Syst.*, vol. 78, no. 2, pp. 568–582, Jan. 2018.

[39] D. B. Rawat and S. R. Reddy, "Software defined networking architecture, security and energy efficiency: A survey," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 1, pp. 325–346, 1st Quart., 2017.

[40] X. Zhong, *Research on Distributed Controller Deployment Algorithm Based on Software Defined Network*. Changsha, China: Hunan Normal Univ., 2017, pp. 29–33.

[41] R. Kandoi and M. Antikainen, "Denial-of-service attacks in OpenFlow SDN networks," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage.*, May 2015, pp. 1322–1326.

DA YIN was born in Youxian, China, in 1990. He received the B.S. degree in communication engineering from Hunan Normal University, Changsha, China, in 2014.

He is currently pursuing the master's degree with Hunan Normal University. He is a member of the Key Laboratory of Internet of Things Technology and Application, Hunan Normal University. His research interests include network security and software-defined networking.

LIANMING ZHANG was born in Xinshao, China, in 1972. He received the B.S. degree in physics and the M.Ed. degree in curriculum and teaching methodology from Hunan Normal University, Changsha, China, in 1997 and 2000, respectively, and the Ph.D. degree in computer application technology from Central South University, Changsha, in 2006.

From 2007 to 2010, he completed a two-year Post-Doctoral Fellowship in complex networks at the School of Computer Science and Engineering, South China University of Technology. He was a Teaching Assistant, a Lecturer, and an Associate Professor with Hunan Normal University from 2000 to 2002, from 2002 to 2007, and from 2007 to 2010, respectively. Since 2010, he has been a Professor with the College of Information Science and Engineering, Hunan Normal University. He manages research projects funded by various sources such as the National Natural Science Foundation of China and companies. He has authored or co-authored over 100 papers. His main research interests include future Internet technology, software-defined networking, complex networks, and network calculus.

KUN YANG (SM'08) received the B.Sc. and M.Sc. degrees from the Computer Science Department, Jilin University, China, in 1991 and 1994, and the Ph.D. degree from the Department of Electronic and Electrical Engineering, University College London (UCL), U.K., in 2006.

He was with UCL, focused on several EU research projects for several years. In 2003, he joined the University of Essex, where he is currently a Chair Professor with the School of Computer Science and Electronic Engineering and leads the Network Convergence Laboratory. He has authored or co-authored over 80 journal papers. His main research interests include wireless networks, future Internet technologies, and mobile cloud computing. He manages research projects funded by various sources, such as U.K. EPSRC, EU FP7/H2020, and companies. He serves on the editorial boards of both the IEEE and the non-IEEE journals. He is a fellow of IET.