FuzzyGuard: A DDoS attack prevention extension in software-defined wireless sensor networks

Meigen Huang[1*] and Bin Yu[1]
[1] Zhengzhou Information Science and Technology Institute
Zhengzhou, Henan, 450001, China
[e-mail: huang_meigen@163.com]
*Corresponding author: Meigen Huang

Received September 25, 2018; revised January 17, 2019; accepted February 21, 2019; published July 31, 2019

Abstract

Software defined networking brings unique security risks such as control plane saturation attack while enhancing the performance of wireless sensor networks. The attack is a new type of distributed denial of service (DDoS) attack, which is easy to launch. However, it is difficult to detect and hard to defend. In response to this, the attack threat model is discussed firstly, and then a DDoS attack prevention extension, called FuzzyGuard, is proposed. In FuzzyGuard, a control network with both the protection of data flow and the convergence of attack flow is constructed in the data plane by using the idea of independent routing control flow. Then, the attack detection is implemented by fuzzy inference method to output the current security state of the network. Different probabilistic suppression modes are adopted subsequently to deal with the attack flow to cost-effectively reduce the impact of the attack on the network. The prototype is implemented on SDN-WISE and the simulation experiment is carried out. The evaluation results show that FuzzyGuard could effectively protect the normal forwarding of data flow in the attacked state and has a good defensive effect on the control plane saturation attack with lower resource requirements.

Keywords: Distributed denial of service, control plane saturation attack, wireless sensor networks, software-defined networking, fuzzy inference

This research was supported by the National Natural Science Foundation of China (NSFC), grant number: 61602513; The Key Laboratory of Information Assurance Technology Open Fund, grant number: KJ-15-104.

1. Introductions

With the rapid development of Internet of Things (IoT), the disadvantages of the traditional wireless sensor network (WSN) [1], as the key supporting technology of its perceptual layer, are increasingly prominent in the aspects of resource sharing and environmental dynamic perception and so on [2][3]. The new WSN adopting the idea of software-defined network (SDN) [4] gradually gained popularity, namely software-defined wireless sensor network (SDWSN) [5][6]. SDWSN generally follows the principles of SDN. From south to north, the architecture consisting of data plane, control plane and application plane maintains the connectivity between adjacent planes through the southbound and northbound interfaces. In addition, flow-based method [7] (flow: a series of packets with some of the same nature) is still adopted in data forwarding. When the sensor node cannot match a data flow (mismatched flow), the header of the data packet (including the source and destination addresses, packet type, etc.) will be sent to the controller to request a new rule. This method is called the Packet-in mechanism. Due to the dynamic nature of the deployment of network applications and the mobility of sensor nodes, the generation of mismatched flows is inevitable. At the same time, part of the mismatched flow plays a special role for the application or algorithm based on learning mechanism, making the Packet-in mechanism an integral component of SDWSN [8].

However, attackers could launch a new type of distributed denial of service (DDoS) attack based on the Packet-in mechanism—control plane saturation attack (abbreviation as saturation attack). The attack method is to inject a large amount of fake invalid data flows into the network, then the sensor nodes cannot match any rules after receiving it. As a result, the control plane converges lots of Packet-in flows within a short period of time, and eventually becomes unable to respond to normal network requests after saturation [9]. What is more unfortunate is that due to the extreme importance of the control plane, its single point of failure will paralyze the entire network [10]. Different from the traditional DDoS attack, the saturation attack uses the Packet-in flow to attack the control plane. Therefore, the destination address in the attack flow could be arbitrarily set and does not need to be fixed as the attack target (control plane), thus making the saturation attack easy to launch, difficult to detect and hard to defend [11].

Based on a deep analysis of the realization and threat of the saturation attack, a DDoS attack prevention extension FuzzyGuard including components of control network construction, fuzzy attack detection and probabilistic flow suppression is proposed in this paper. The first component is responsible for selecting some sensor nodes from the data plane to specifically transmit control flows such as Packet-in. Therefore, when the network is attacked, the attack flow would be quickly isolated and converged so that the data flow is not affected. The second component uses the Mamdani fuzzy inference system [12] to achieve an efficient perception of attacks. The motivation of adopting fuzzy detection is that all fields in the attack flow can be arbitrarily set, and it is difficult and costly to accurately identify the attack flow. The method of evaluating the overall security state of the network through fuzzy inference can be free from identifying the attack flow, so that efficient attack detection can be realized at a lower cost. Referring to the fuzzy detection results, the last component adopts various suppression modes such as zero-probability, low-probability, and penalty-probability to dynamically restrain the attack flows in the network.

The main contributions are as follows:

1. Based on a review of multiple saturation attack defense schemes, it is pointed out that there is no relevant research result in SDWSN.
2. Considering the resource-limited nature of WSN, an easy-to-implement saturation attack method is given, and security threats are analyzed accordingly.
3. A DDoS attack prevention extension FuzzyGuard is proposed which includes control network construction, fuzzy attack detection, and probabilistic flow suppression.
4. The prototype of FuzzyGuard is implemented, and its performance is evaluated on SDN-WISE.

2. Related Work

Since the combination of SDN and WSN is currently in the initial stage of exploration, security has drawn less attention [13][14], especially in the DDoS attack defense. However, in the SDN environment, a large number of defense schemes have been proposed. Because WSN has the defects of resource constrains and unreliable links, these schemes are usually difficult to directly be extended to SDWSN, but they are still valuable for reference.

AVANT-GUARD [9] holds that the root cause of saturation attack lies in the scalability defects that the SDN centralized control faces. Therefore, it proposes two security extensions of connection migration and actuating triggers in the data plane. The former is able to effectively reduce the frequency of interaction between the control and data planes, while the latter could quickly improve the response speed of the control plane. However, LineSwitch [15] points out that the process of state maintenance in the above connection migration may lead to a new denial of service (DoS) attack—buffer overflow attack. Therefore, the authors take the SYN proxy and network flow blacklist technologies to resist saturation attack, which has an advantage over AVANT-GUARD in processing time. However, they have the same drawback that they are valid only for the TCP protocol but not for UDP and ICMP.

In response, Wang et al. propose the protocol-independent DoS attack prevention extension FloodGuard [11], which includes two novel modules of proactive flow rule analyzer and packet migration. Packet migration is achieved by setting the Packet-in flow cache center in the data plane. Similarly, based on access control and flow classification, SGuard [16] also designs a cache center to handle burst Packet-in flows. However, this Packet-in flow cache mechanism does not work well in SDWSN. The reason is that the node itself has very limited storage resources, and the node selected as the cache center will quickly overflow after the attack is initiated, failing to perform the function of buffering the Packet-in flows effectively. In addition, [8] put forward a Packet-in flow filtering mechanism from the perspective of identifying the attack flow. The data plane is preset in advance with the re-match rules for important mismatched flows. After the flow is re-matched, it is sent to the control plane in rate-limiting mode to deal with the saturation attack without losing important Packet-in flows. However, this mechanism requires a large amount of valuable rule space, and once the attacker masters these rules, the filtering mechanism fails.

In addition, [17] gives a DDoS attack defense scheme based on a legitimate source and destination IP address database, but it may misjudge many burst normal flows as attack flows. In [18], artificial neural network is used to detect known and unknown DDoS attacks. The detection ability excessively dependents on the effect of centralized training. Therefore, this method may not be able to cope with saturation attack. The reason is that the malicious flow could be arbitrarily forged. Similarly, the DDoS attack detection method [19] based on the difference of the destination IP address entropy is also hard to resist saturation attack. Further,

[20] uses the source IP address entropy, the destination IP address entropy, and the flow request rate as detection elements, and adopts the fuzzy synthetic evaluation decision-making model to evaluate the degree of attack on the controller. Because the source and destination IP addresses of the attack flow can be arbitrarily specified by the attacker, thus distorting the address entropy, this method is difficult to deal with saturation attack. However, using fuzzy inference system to detect DDoS attacks has lower resource overhead, which provides us with some research ideas.

3. Attack Implementation and Threat Analysis

This paper focuses on the DDoS attack defense in SDWSN with main attention attached to saturation attack. We first give an easy-to-implement attack method, and then analyze the related attack threat, as shown in **Fig. 1**.

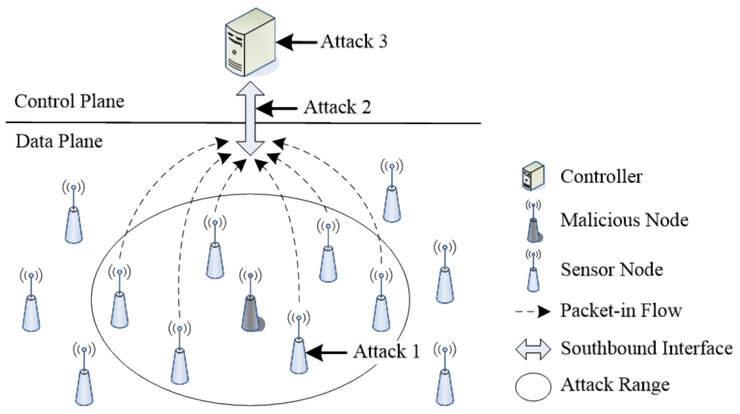

Fig. 1. The threat model of control plane saturation attack

The model involves the control plane and the data plane. The control plane consists of a controller and the data plane includes all sensor nodes. The control plane communicates with the data plane through the southbound interface. To launch a saturation attack, a malicious node needs to complete two steps: mismatched flow generation and mismatched flow injection. Note that a malicious node is manipulated by an attacker, which may be a compromised internal node or an external attack node.

(1) mismatched flow generation

Each attack packet in the mismatched flow includes packet header and packet payload. The field of the network identifier in the packet header needs to be acquired by eavesdropping in advance [21]; the fields such as the source and destination addresses could be generated arbitrarily or with specified pattern; the length and content of packet payload could also be generated randomly. If the target network is equipped with security mechanisms such as access control or authentication, the attacker is able to steal the security materials from legitimate sensor nodes through capture attacks [22].

(2) mismatched flow injection

After the mismatched flow is generated, the malicious node needs to inject it into the network. In the model, the malicious node adopts a method of increasing the transmission power (modifying the radio frequency parameters or adding a radio frequency front-end chip), leading to that all sensor nodes in the attack range would effectively receive the attack packets, thereby implementing a distributed attack.

During the attack, the sensor node cannot match any rule after receiving the mismatched flow. Subsequently, the sensor node requests the new rule from the control plane through Packet-in mechanism, and the packet payload is stored directly in itself. Therefore, it may occupy the node with excessive storage space, and it may also install too many useless rules, resulting in that the legitimate data flow cannot be efficiently forwarded (**Attack 1**). After that, lots of Packet-in flows emerge in the southbound interface channel, which consumes the communication bandwidth and causes the normal control flows to delay or fail to obtain the service (**Attack 2**). Finally, the influx of Packet-in flows significantly exceeds the real-time processing capability of the control plane which would no longer serve the network after saturation (**Attack 3**). At this point, the purpose of the saturation attack is achieved. Note that enhancing the controller's processing power or deploying multiple controllers is essentially more of a mitigating method than a fundamentally defending approach.

4. DDoS Attack Defense Framework

Based on the threat model, combined with the resource-limited nature, it is analyzed that the design of FuzzyGuard must satisfy the following requirements.

Requirement 1: Defense usually lag attack. From the point of network utility, it is required to realize the normal forwarding of legitimate data flows when the network is attacked.

Requirement 2: The energy supply of WSN is relatively limited. Therefore, from an energy-efficient perspective, it is required to increase the cost-effectiveness of the attack defense as much as possible.

Requirement 3: The Packet-in mechanism is the key to saturation attack. Thus, from the aspect of attack defense, it is required to reasonably adjust the total amount of Packet-in flows in the network.

The DDoS attack defense framework is shown in **Fig. 2**, which involves control plane and data plane. The addition of FuzzyGuard to the control plane includes three components: control network construction, fuzzy attack detection, and probabilistic flow suppression. Management application and data application are general-purpose applications in the controller which are responsible for responding to control flows (i.e. Packet-in and report) and data flows. In addition, the network global view is used to store the network topology and state parameters and provide the required security parameters to the fuzzy attack detection component. The control network and the data network in the data plane are implemented according to the construction rules generated by the control network construction component. The suppression of Packet-in flows is the responsibility of the suppression rules in the data plane, which are generated by the probabilistic flow suppression component and transmitted by the control network. To comply with the current general principles of security enhancement, all the above components are implemented as controller applications, so FuzzyGuard is independent of SDWSN architecture.

To meet **Requirement 1**, the control network construction component protects the normal forwarding of data flows by taking the idea of independent routing control flow. First, select part of the sensor nodes from the data plane as the control nodes. All control nodes then form a control network and communicate with the management application, specifically responsible for control flows routing such as Packet-in. All remaining sensor nodes (called data nodes) form a data network and connect with data applications, responsible for data flow routing. Finally, the data flow generated by the control network is also transmitted by the data network, to reduce the impact of the saturation attack on the data flow forwarding as much as possible.

To satisfy **Requirement 2**, the fuzzy attack detection component adopts the method of overall network security assessing instead of the resource-intensive attack flow identification method. It reads statistical data related to network security status (security parameters) from the network global view and obtains the attack probability through fuzzy inference. Therefore, effective attack detection could be realized at a lower cost, and the cost-effectiveness of attack defense would be effectively improved.

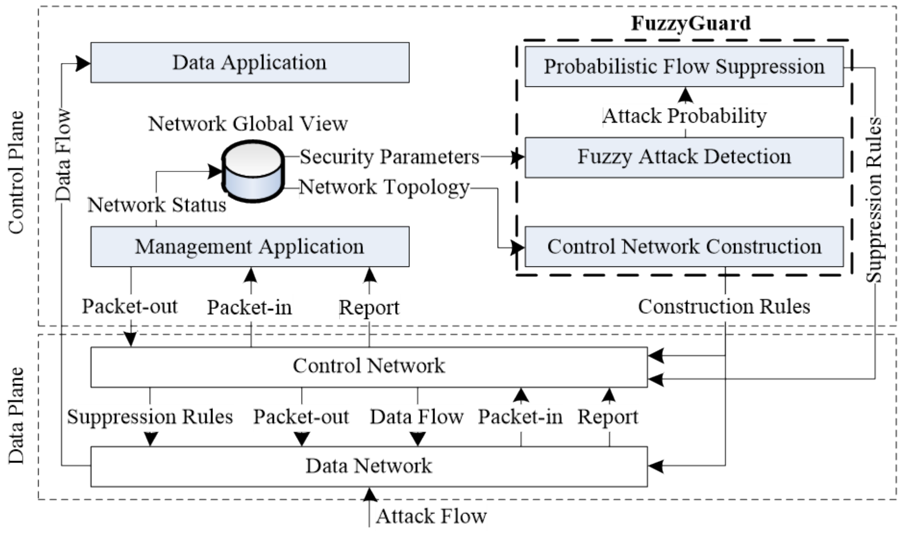

Fig. 2. The DDoS attack defense framework

For Requirement 3, the probabilistic flow suppression component dynamically adjusts the Packet-in flow suppression probability based on the attack severity. The suppression modes include zero-probability, low-probability and penalty-probability. The probability is sent to the data network in the manner of suppression rule, thereby limiting the rate at which sensor nodes generate Packet-in flows to prevent instantaneous saturation of the control plane.

5. FuzzyGuard

Based on the DDoS attack defense framework, this section systematically introduces FuzzyGuard. By designing components such as control network construction, fuzzy attack detection and probabilistic flow suppression, the defending ability against saturation attack of FuzzyGuard is greatly enhanced.

5.1 Control Network Construction

Although the control network is located on the data plane, it is built by the control plane. After all control nodes are selected, this component generates corresponding control rules and data rules and distributes them. After the sensor node receives the control rule, its identity changes to the control node. In the end, the control network is constructed. Analogously, all data nodes establish the data network according to the data rules.

Therefore, selecting the control node is the primary task. On the one hand, building a control network will reduce the number of data nodes and inevitably affect the efficiency of data flow transmission. Therefore, the number of control nodes should be as small as possible. On the other hand, the premise of the efficient operation of the control network is that it must be able to communicate with all data nodes and be self-connected. It is assumed that SDWSN itself is connected. We adopt the idea of loop optimization to design control node selecting algorithm. For ease of explanation, the relevant definitions are given first.

Definition 1. The sensor node set *S* is composed of all sensor nodes in the data plane. There are bidirectional links between adjacent nodes, and $|S|=n$. If the elements in matrix $R=[r_{i,j}]_{n\times n}$ satisfy equation (1), then *R* is called the adjacency matrix of *S*; if the elements in matrix $M=[m_{i,j}]_{n\times n}$ meet equation (2), then *M* is called the reachability matrix of *S*.

$$r_{i,j}=\begin{cases}1 & \text{Nodes } i \text{ and } j \text{ are neighbors}\\ 0 & \text{Nodes } i \text{ and } j \text{ are not neighbors}\end{cases} \tag{1}$$

$$m_{i,j}=\begin{cases}1 & \text{At least one path between nodes } i \text{ and } j\\ 0 & \text{No path between nodes } i \text{ and } j\end{cases} \tag{2}$$

Definition 2. Let α and β be any two row vectors of *R*, and $\alpha\neq\beta$, then define the operation "$\circ$" satisfies $\alpha\circ\beta=\left[\alpha_1\circ\beta_1,\alpha_2\circ\beta_2,\cdots,\alpha_k\circ\beta_k,\cdots,\alpha_n\circ\beta_n\right]$, where the operation rule of $\alpha_k\circ\beta_k$ is shown as equation (3).

$$\alpha_k\circ\beta_k=\begin{cases}1 & \alpha_k=1,\beta_k=1\\ 0 & \text{Other}\end{cases} \tag{3}$$

On the basis of the above definitions, the idea of matrix transformation to achieve the neighboring and reachable relationship judgment between nodes is adopted. The control node selecting algorithm includes three steps of optimization, shrinkage and connection, as shown in **Algorithm 1**. The purpose of optimization is to search for the node with the largest number of neighbor nodes, thus ensuring the minimum number of control nodes. The optimization in the algorithm is implemented by the adjacency matrix, and the node corresponding to the row vector with the most "1" elements (neighbor nodes) in the matrix is set as the control node (line 3-7). The shrinkage is used to delete the nodes that have been determined to be connectable, so that the optimization result would connect all data nodes. The shrinkage is implemented by simplifying the adjacency matrix. The method is to delete the column vector node corresponding to the "1" element in the row vector (the data node could be connected by the control node), thereby continuously reducing the size of the adjacency matrix (line 8-13). The connection is responsible for finding the best data nodes to connect all control nodes and the selected data nodes become the control nodes. In the algorithm, the reachable matrix is used to determine whether the control network is self-connected. For the "0" element in the matrix (the row vector node and the column vector node are not connected), look for a row vector that satisfies the non-zero vector of the "$\circ$" operation results with the above two vectors, and mark it as control node (line 14-28).

Algorithm 1: Control node selecting

Input: *R*: adjacency matrix of *S*

Output: *C*: control node set

```
1: $copy\_R \leftarrow R$, $temp\_R \leftarrow O$, $M \leftarrow O$   // O represents zero matrix
2: while $copy\_R == O$ then
3:   for each row vector $R_i$ in $copy\_R$ do
4:     if count_one($temp\_R$) < count_one($R_i$) then  //count_one($x$) is used to count the number of "1" in $x$
5:       $temp\_R = R_i$, $temp\_i = i$  // $temp\_i$ is temporary variable
6:     end if
7:   end for
8:   do $C$ += node $temp\_i$  // add node $temp\_i$ to $C$
```

```
9:  for each element $r_j$ in temp_R do
10:   if $r_j$ == 1 then  // node j could be connected by the control node temp_i
11:     copy_R $-=$ $R'_j$  // delete the jth column vector $R'_j$ from copy_R
12:   end if
13:  end for
14: while (1) then
15:  do $M$ = solve_reachability($C$)  // solve_reachability($x$) is used to solve the reachability matrix of $x$
16:  if ones($M$) == True then  // ones($x$) returns true iff the elements in $x$ are all "1"
17:    return $C$
18:  else then
19:    for each element $M_{ij}$ in $M$ do
20:      if $M_{ij}$ == 0 then // node i and j are disconnected
21:        for each row vector $R_k$ in $R$ do
22:          if $R_k \circ R_i \neq O$ && $R_k \circ R_j \neq O$ then  // $R_i$ and $R_j$ are the ith and jth row vectors in $R$
23:            $C$ += node $k$  // node k could connect node i and j at the same time, add k to C
24:          end if
25:        end for
26:      end if
27:    end for
28:  end if
```

The necessary explanation of the algorithm is as follows. The optimization and shrinkage are applied to the replica *copy_R* of *R*, and only when the *copy_R* is null indicates the completion of the steps of optimization and shrinkage, that is, the minimum control node set that could connect all the data nodes is determined. The function solve_reachability(C) in line 15 solves the reachability matrix M by generating the adjacency matrix of C. Considering the bidirectional links, both the adjacency matrix and the reachability matrix are symmetric square matrices, so the order in which the "0" element is found in the line 20 does not affect the correctness of the algorithm. The row vector R_k in the line 22 must exist, so the line 23 must be executed. Otherwise, there is an isolated node set in the original network, which does not conform to the algorithm assumption.

5.2 Fuzzy Attack Detection

The fuzzy attack detection component is implemented by a fuzzy inference system consisting of four modules: fuzzification, fuzzy inference, defuzzification, and rule-base. Fuzzification, fuzzy inference, and defuzzification are implemented by the triangular membership function, the Mamdani and the gravity methods, respectively. The establishment of a rule-base is given by administrators or experts based on network operating experience. The reason that rule-base is not established through self-learning such as training is that all elements in the attack flow are variable. Considering the energy efficiency, the detection component operates in a periodic manner, and the runtime threshold is denoted as T.

(1) Fuzzification

The fuzzification module is responsible for converting the network status parameters (exact values) into fuzzy input variables for the fuzzy inference module. According to the threat points in the model (**Fig. 1**), four security parameters are selected: average storage load of the node (**Attack 1**), average flow table space occupation of the node (**Attack 1**), Packet-in flow in the control network (**Attack 2**), and load of the control plane (**Attack 3**), which are respectively denoted as NS, NF, CP and CL. The time window for all parameters is the same as T.

To facilitate parameter fuzzification, a normalization process is first performed. The parameter is expressed as a ratio ([0,1]) of current value to the feasible maximum value. For example, *CP* is the ratio of the current Packet-in flow to the maximum Packet-in flow that the control network supports or allows. Triangular membership function is shown in **Fig. 3**.

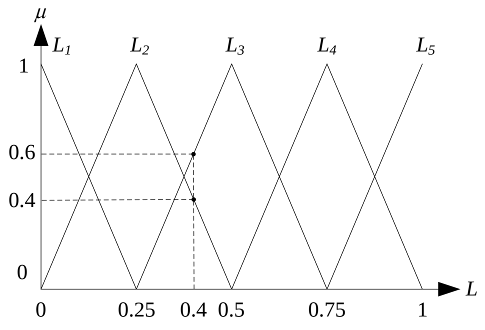

Fig. 3. The triangular membership function

A total of 5 fuzzy levels from L_1 to L_5 are designed in function μ, which correspond to the demarcation points (called fuzzy single values) in the range of the independent variables in **Fig. 3**, namely 0, 0.25, 0.5, 0.75, and 1. In terms of semantics, L_1 to L_5 are interpreted as very low, low, general, high, and very high, and the probability of the network being attacked increases in turn. Since all inputs have been normalized, the fuzzification process of all input parameters could be completed according to **Fig. 3**. Taking $NF = 0.4$ as an example, the fuzzification result could be expressed as $\mu_{(NF=0.4)} = \frac{0.4}{L_2} + \frac{0.6}{L_3}$, indicating that when the average flow table space occupancy of the node reaches 40%, 40% probability belongs to L_2 (low) and 60% probability is L_3 (general). The semicolon indicates the membership degree of the fuzzy single value.

(2) Fuzzy inference and rule-base

The fuzzy inference module takes the given fuzzy variables as input, activates the relevant rules from the rule-base, and obtains fuzzy inference results. Therefore, the rule-base becomes the most important module in the fuzzy inference system and directly affects the inference results. Based on Mamdani, the rule-base denoted as $Rules = \{rule_l\}$, where $rule_l$ is a rule that represents the mapping relationship between a set of fuzzy inputs and a fuzzy output. The rule is expressed in the form of "if *x* then *y*", such as "$rule_l$: if *NS* is L_1 and *NF* is L_3 and *CP* is L_3 and *CL* is L_2, then *SA* is OL_2". *SA* is a fuzzy output variable, which represents the network security status, including three fuzzy levels: OL_1, OL_2 and OL_3. The corresponding values are 0.2, 0.5, and 0.8, meaning that they are unlikely to be attacked, may be attacked, and are likely to be attacked.

The completeness and consistency check are required before the rule-base is enabled. Completeness means that at least one rule responses to any fuzzy input, which is a necessary condition for the rule-base to work properly. Consistency means that fuzzy rules with similar inputs should have similar outputs. Therefore, according to the completeness, there are at least $5^4 = 625$ rules ("5" is the number of fuzzy levels, "4" is the number of input variables) in *Rules*. Since the fuzzification module adopts the triangular membership function, each input parameter will be blurred into two fuzzy levels (it may be ignored when the membership degree is 0). Thus, the fuzzy inference module would activate the relevant $2^4 = 16$ rules in the rule-base in a given network state (4 input parameters), which is fuzzy output.

(3) Defuzzification

The defuzzification module adopts the gravity method to precisely process the fuzzy results. The 16 rules of fuzzy inference output are recorded as *rule*$_1$ to *rule*$_{16}$, and the membership degrees of *NS*, *NF*, *CP*, and *CL* in *rule*$_l$ are denoted as $\mu_{l,NS}$, $\mu_{l,NF}$, $\mu_{l,CP}$, and $\mu_{l,CL}$. Therefore, the Zadeh method with Cartesian product is used to convert the multidimensional fuzzy inference model into a simple one. The formula for calculating the network security state $O_{(SA)}$ with the gravity method is shown in equation (4).

$$O_{(SA)} = \frac{\sum_{l=1}^{16}\left(SA_l \times \mu_{(SA_l)}\right)}{\sum_{l=1}^{16}\mu_{(SA_l)}} \tag{4}$$

Where $\mu_{(SA_l)} = \mu_{l,NS} \times \mu_{l,NF} \times \mu_{l,CP} \times \mu_{l,CL}$, the value of SA_l is in $\{0.2, 0.5, 0.8\}$, and the specific value is determined by the rule. Obviously, $O_{(SA)} \in [0,1]$ is known from the domain of independent variables.

5.3 Probabilistic Flow Suppression

Probabilistic flow suppression component takes a non-discriminatory way to restrain the Packet-in flows. The suppression mode includes zero-probability, low-probability and penalty-probability. Which one to use is related to the network security status. The sensor node performs a probabilistic suppression through random number generation. After a flow fails to match, if the random number ([0,1]) generated temporarily by the node is larger than the probability, the packet header is further encapsulated into a Packet-in flow and forwarded by the control network. Otherwise, the mismatched flow is directly discarded.

Based on the above analysis, the designed inhibition probability is shown in equation (5). The α and β are the critical points of inhibition probability, and obviously $0 \le \alpha \le \beta \le 1$.

$$\lambda = \begin{cases} 0 & 0 \le O_{(SA)} < \alpha \\ O_{(SA)} - \alpha & \alpha \le O_{(SA)} \le \beta \\ p \ln O_{(SA)} + 1 & \beta < O_{(SA)} \le 1 \end{cases} \tag{5}$$

When $0 \le O_{(SA)} < \alpha$, it indicates that the network is currently in a state of “unlikely to be attacked”. Therefore, the zero-probability mode is adopted, that is, the node sends all the Packet-in flows to the control plane.

If $\alpha \le O_{(SA)} \le \beta$, the network is in a state of “may be attacked”. In this case, the low-probability mode should be taken, that is, the suppression probability is designed to be a linear increase mode with a maximum value of $\lambda_\beta = \beta\text{-}\alpha$.

Once $\beta < O_{(SA)} \le 1$, the network is in a state of “are likely to be attacked”. Therefore, the penalty-probability mode is used, and the probability is designed as a logarithmic growth mode $p \ln O_{(SA)} + 1$, thereby rapidly reducing Packet-in flow in the network. To ensure that the suppression probability at this stage is not negative, $\beta \ge e^{-1}$ should be satisfied.

The parameter *p* is the penalty factor and is characterized by the product of the node's average residual energy level (*RE*) and the network attack tolerance (*AT*), that is,

$p = RE \times AT$. The value of RE is the ratio of the current average residual energy of the network nodes to the initial energy, so $RE \in [0,1]$, and the energy factor is designed to extend network life. The AT ($AT \in [0,1]$) is determined by the administrator based on the application type, deployment environment, and other factors. The smaller the value, the lower the network's tolerance for attacks. Therefore, p dynamically changes with the network operating conditions, and $p \in [0,1]$. Suppression probability curve λ is shown in **Fig. 4**, where α=0.2, β=0.5 and p is taken as 0.2, 0.5, and 1 respectively.

Obviously, the punitive inhibition probability is related to α, β, and p. When $\beta - \alpha$ is small, it means that the low-probability suppression range is narrow. At this time, the administrator is more aggressive. On the contrary, it indicates that the administrator is more moderate and tends to suppress the Packet-in flow in the network with a slowly increasing probability. In addition, when in the penalty-type suppression interval, the smaller p is, the stronger the penalty is, and the higher the suppression probability is. In the extreme case of $p = 0$, if $O_{(SA)} > \beta$, then $\lambda = 1$, the network discards all Packet-in flows.

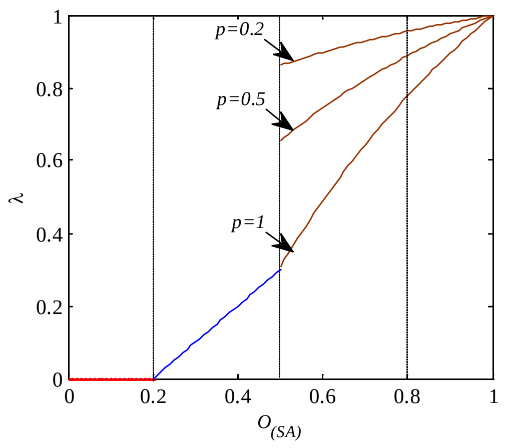

Fig. 4. The case of suppression probability curve

It should be noted that since it is costly to accurately identify the attack flow that can be arbitrarily forged, the component adopts an indiscriminate suppression manner that does not distinguish between a legal Packet-in flow and an illegal Packet-in flow (caused by the attack flow). Therefore, it is unavoidable that some Packet-in flows are handled incorrectly (a legal Packet-in flow is discarded or an illegal Packet-in flow is sent). However, the discarded legal Packet-in flow is usually generated again, which can reduce the negative impact of the mechanism on the network performance to some extent. Therefore, the probability φ of the legal Packet-in flow is erroneously suppressed as shown in equation (6), θ is the number of regeneration times of legal Packet-in flows.

$$\varphi=\lambda^{\theta+1} \tag{6}$$

6. Experiment and Evaluation

SDN-WISE [23] is a fully open source SDWSN architecture with source code (java) located at “https://github.com/sdnwiselab/sdn-wise-java”. The WISE flow table forwarding structure is designed at the network layer. This experiment is based on SDN-WISE and is mainly implemented based on the core code of its protocol architecture, flow table forwarding structure, and network global view. We focused on writing component codes such as control network construction, fuzzy attack detection, and probabilistic flow suppression in the controller to implement the FuzzyGuard prototype. The experimental host uses ThinkPad T470 laptop (Windows 10 (64-bits) OS, Intel Core I5-7200U CPU, 8G memory, 128G SSD), and runs Ubuntu_12.04 (32-bits) OS on VMWare Player 12. Furthermore, the Contiki 3.0 is installed on Ubuntu, and the experiment is carried out on the COOJA simulation platform.

6.1 Experimental Deployment

In the experimental deployment, a single controller is deployed on the control plane, and 100 sensor nodes (including 1 Sink and 5 malicious nodes) are deployed in the 200×200 m^2 area on the data plane. The control node is selected by the control network construction component after the network deployment. The legal data flows generated by the data node are sent to Sink, and the control node does not respond to the legitimate data flow. A malicious node could attack all neighbor data nodes within the transmission radius. The header of attack packet is generated on demand, and the payload data is randomly generated. The attack packets are finally encapsulated and injected into the network in a broadcast manner so that the data nodes cannot match and send Packet-in flows to the controller. To ensure the efficient injection of attack flows, the network does not deploy security mechanisms such as authentication.

To facilitate comparison with [19] and [20], the attack rate is designed to be R15, R25, and R50, which means that the ratio of the number of attack packets injected into the network to the number of normal packets in T time is 0.15, 0.25, and 0.50, respectively. The task of attack packet injection is equally shared by all malicious nodes. In addition, the attack rate was recorded as R00 when the network is not attacked. The relevant parameter settings are shown in **Table 1**. To make use of fuzzy rules, the cardinal number of security parameters *NS*, *NF*, *CP* and *CL*, that is, the maximum storage load of node, the maximum rule number of nodes, the maximum Packet-in flow of control network and the maximum load of the control plane in *T*, are set to 512Bytes, 32, 128 and 64, respectively.

Table 1. The related parameter settings

Parameters	**Values**	**Parameters**	**Values**
Transmission radius	50 m	Legal flow generation rate	30 packets/min
Interference radius	0 m	Legal flow load length	32 Bytes
Link packet transmission success rate	100%	Attack flow load length	1 Byte ~ 64 Bytes
Link packet reception success rate	100%	Time threshold *T*	10 s
Node initial energy	9×10^6 mC	The cardinal number of *NS*	512 Bytes
Energy consumption (sending 1 bit of data)	2.7×10^{-3} mC	The cardinal number of *NF*	32
Energy consumption (receiving 1 bit of data)	9.4×10^{-4} mC	The cardinal number of *CP*	128
Energy consumption (activated for 1 s)	6.8 mC	The cardinal number of *CL*	64

6.2 Experimental Results

Based on the above experimental deployment, validated experiments were designed for the three components of FuzzyGuard, namely data flow protection, DDoS attack detection and Packet-in flow suppression. The malicious node launches attacks at the rate of R15, R25, and R50 from the moment of network operation. The attack time for each rate is *3T*, and the interval time is *T*. All experiments were performed 10 times and the results were averaged.

(1) Data flow protection

It is an important task for control network construction component to make sure that the legitimate data flows are still efficiently forwarded when the network is under attack. The experiment results of data flow protection are shown in **Fig. 5**, where (a) is the average transmission delay (end-to-end) comparison of data packets, and (b) is the average delivery ratio comparison of data packets.

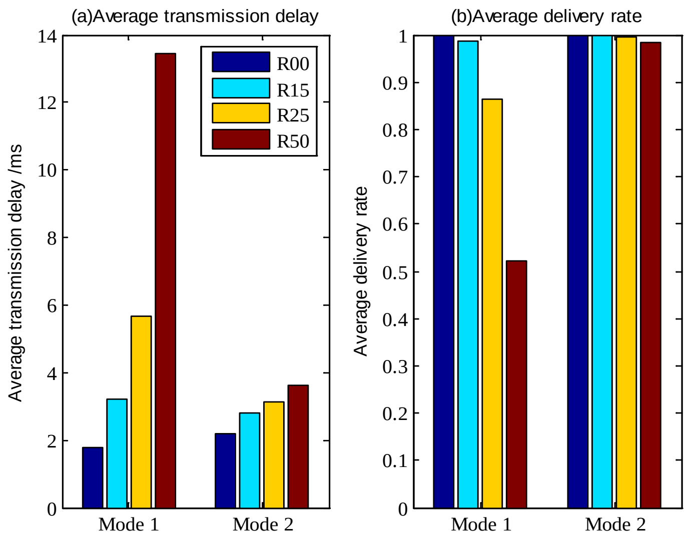

Fig. 5. The experimental results of data flow protection

As shown in **Fig. 5(a)**, the average transmission delay without the control network (**Mode 1**) is significantly longer than that for the case with the control network (**Mode 2**). The faster the attack rate of malicious node, the more obvious the gap between the two. When the attack rate is R50, the average transmission delay of **Mode 1** is as high as 13.45ms, while **Mode 2** is only 3.61ms, indicating that the control network could effectively guarantee the quality of service under attack. At the same time, as the attack rate increases, the average transmission delay in both cases shows an increasing trend, which means that saturation attacks have a greater impact on network performance. In addition, under the attack rate R00, the average transmission delay of **Mode 1** (1.80ms) is slightly shorter than **Mode 2** (2.17ms). The reason is that the control network occupies 9 data nodes (redefined as control nodes).

In **Fig. 5(b)**, when the attack rate reaches R50, the packet loss rate is only 1.7% in **Mode 2**, while it is 47.7% in **Mode 1.** More unfortunately, the network began to lose packets (1.3%) after the malicious node launched the attack (R15) in **Mode 1**. The reason for packet loss is that the rule space and storage space of some data nodes close to Sink overflow, resulting in subsequent data flows being discarded. It can be seen that building a control network could effectively improve the delivery rate of network packets under attack, indicating that the use of independent routing control flow has the effect of protecting data flow forwarding.

(2) DDoS attack detection

Fuzzy attack detection component takes parameters *NS*, *NF*, *CP* and *CL* as inputs to the fuzzy inference system to obtain the probability of the network being attacked. [19] detects DDoS attacks based on the changes in destination address entropy (the entropy threshold is set to 0.05). In [20], source address entropy, destination address entropy and flow request rate are inputs, and fuzzy synthetic evaluation decision-making model is used to obtain the probability that the current controller is attacked. Therefore, under the experimental deployment and setting conditions, the malicious attack modes of **Type1** and **Type2** are specially designed. The source address and destination address of attack packets in **Type1** mode are selected from the legal node addresses and distributed evenly. **Type2** mode uses randomly forged source and destination addresses to generate attack packets. The comparison results of DDoS attack detection between this paper and [19][20] are shown in **Fig. 6**. X-axis is an experimental time series in units of *T* and Y-axis is the detection output. The experimental time range from R15 to R50 and end the attack.

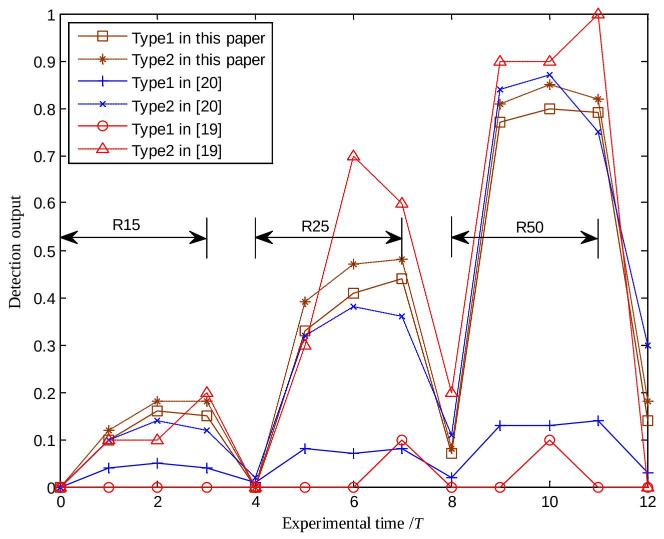

Fig. 6. The experimental results of DDoS attack detection

From **Fig. 6**, we can see that for the **Type1** mode with well-designed packets addresses, the attack detection result of [19] is only 0.1 at 7*T* and 10*T* (representing that only 1 attack was detected in 10 experiments). However, [20] also outputs a lower probability of attack on the network, and only gives an attack probability of no more than 15% when it is at the R50 attack rate. The root cause of the above misjudgment is that the address entropy is used as the detection element, and the source address and destination address of the attack packet could be arbitrarily set. Therefore, the malicious node could maintain the address entropy unchanged after being accurately designed, thereby distorting the detection methods. However, the fuzzy output of [20] still slowly increases with the speed of the attack. At this time, the flow request rate (similar to the *CP* parameter in this paper) is still in effect, so that the attack can be partially detected.

For the **Type2** mode randomly configured for the attack flow address, [19] could basically detect attacks when the rate is fast. However, [20] is similar to this paper in the fuzzy detection result and both follow the attack rate well to output the security state of the network. The two outputs are approximately 15%, 40%, and 82% at R15, R25, and R50 attack rates, respectively.

However, usually the administrator cannot know the attack mode taken by the attacker in advance. Therefore, outputting similar detection results for the same attack rate under different attack modes is extremely important to avoid misjudgment. However, in [20], the difference between the two detection results under the R50 attack rate exceeds 70%, and it is very likely that the **Type1** attack mode with R50 rate is misjudged as **Type2** attack mode with R15 rate. [19] is even more extreme and only can recognize **Type2** attack pattern.

In contrast, the difference between the two detection results in this paper is no more than 6%. The reason is that the fuzzy detection component of this paper uses the security parameter not controllable by the attacker as the detection element, eliminating the risk of being manipulated by the attacker. Therefore, this paper could output similar detection results for two attack modes and effectively reflect the actual attack conditions. For example, when the attack rate is R50, the probability of the network being attacked is floating between 76% and 84%. The relevant activation rules at this time are shown in **Table 2** (*IR* represents the result of rule inference). It is worth mentioning that, the reason that the detection result of the **Type1** mode is slightly lower than that of the **Type2** mode is that the legitimate node address is limited, and there are a small number of attack flows that are matched by the data node as legitimate flows. In summary, compared with [19] and [20], the fuzzy attack detection in this paper have no misjudgment and better detection results.

Table 2. The related activation rules (R50 attack rate)

NS	*NF*	*CP*	*CL*	*IR*	*NS*	*NF*	*CP*	*CL*	*IR*
L_3	L_3	L_3	L_3	OL_2	L_4	L_3	L_3	L_3	OL_2
L_3	L_3	L_3	L_4	OL_2	L_4	L_3	L_3	L_4	OL_3
L_3	L_3	L_4	L_3	OL_3	L_4	L_3	L_4	L_3	OL_3
L_3	L_3	L_4	L_4	OL_3	L_4	L_3	L_4	L_4	OL_3
L_3	L_4	L_3	L_3	OL_2	L_4	L_4	L_3	L_3	OL_3
L_3	L_4	L_3	L_4	OL_3	L_4	L_4	L_3	L_4	OL_3
L_3	L_4	L_4	L_3	OL_3	L_4	L_4	L_4	L_3	OL_3
L_3	L_4	L_4	L_4	OL_3	L_4	L_4	L_4	L_4	OL_3

(3) Packet-in Flow suppression

Packet-in flow suppression is an important means to slow the saturation attack. In order to analyze the probability that a legal Packet-in flow is erroneously suppressed, a legal node is specifically designed to generate a legal mismatched flow at an R10 rate (legal mismatched packets occupy 10% of the number of data packets). At the same time, in order to test the inhibitory effect of Packet-in flow under different suppression probabilities, and taking into account the fuzzy detection results, the critical points α and β of the inhibition probability are set to $\alpha=0.2, \beta=0.5$ and $\alpha=0.1, \beta=0.9$, and the attack tolerance *AT* space is set to {0.3, 0.7}, the regeneration times $\theta=3$. In addition, the attack flow adopts **Type2** mode. Based on this, by observing the suppression of legal Packet-in flows and illegal Packet-in flows under different attack rates, the experimental results of flow suppression are shown in **Fig. 7**. Similar to **Fig. 6**, X-axis still records the time range (in *T*) that the network is attacked, and Y-axis is the number of suppressed Packet-in flows.

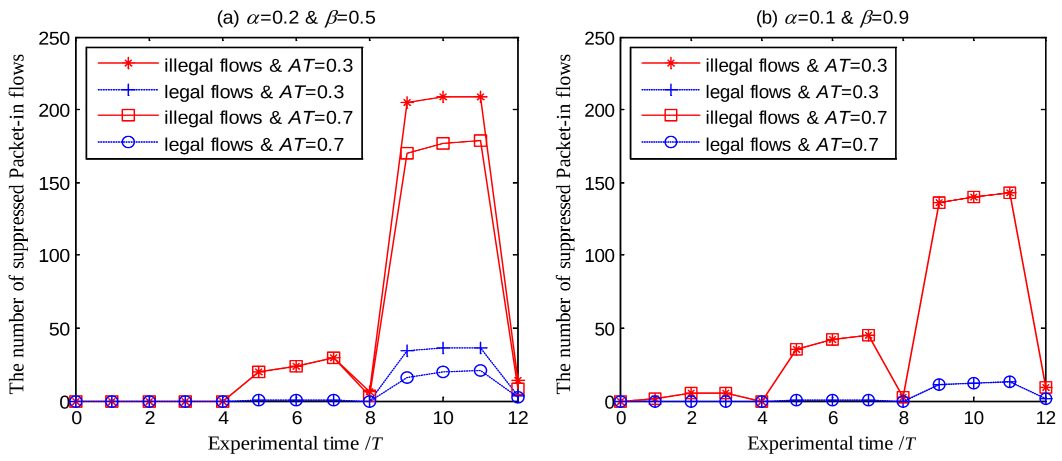

Fig. 7. The experimental results of Packet-in flow suppression

When a malicious node launches an attack at an R15 rate (0*T* to 3*T*), it can be seen from **Fig. 6** that the fuzzy attack detection result is between 0.1 and 0.2. Therefore, when α=0.2, the network adopts the zero-probability suppression mode (**Fig. 7(a)**), all Packet-in flows are transmitted to the control plane, and when α=0.1, the network adopts the low-probability suppression mode (**Fig. 7(b)**). The number of illegal Packet-in flows that are suppressed in each *T* is a maximum of 7, and the number of discarded legal Packet-in flows is 0.

For the R25 attack rate (4*T* to 7*T*), the detection result of the fuzzy attack is less than 0.5. At this time, both of **Fig. 7(a)** and **Fig. 7(b)** follow the principle of low-probability suppression. However, due to the different values of α, the actual application of the two has a difference in suppression probability of 0.1, resulting in the amount of illegal Packet-in flow being suppressed in **Fig. 7(b)** to be slightly larger than **Fig. 7(a)**. At the same time, due to the low suppression probability, the legal Packet-in flow is regenerated at most 3 times and is successfully transmitted to the control plane.

When the attack rate is R50 (8*T* to 11*T*), the attack detection result is as high as 0.8, which is significantly larger than the value in **Fig. 7(a)**. Therefore, the penalty-probability suppression mode is adopted, and the Packet-in flow that is suppressed in the network is significantly increased, indicating that the penalty-probability suppression could quickly reduce the level of packet-in flow in the network. At this time, when the attack tolerance is low (*AT* = 0.3), the number of illegal Packet-in flow suppression exceeds 200, which is approximately 40 (about 20%) more than the case of higher attack tolerance (*AT* = 0.7). However, there is a difference of 40% in the amount of legal Packet-in flows suppression. Therefore, a reasonable adjustment of *AT* has a greater impact on the network performance.

In addition, as can be seen from **Fig. 4**, when the fuzzy attack detection result is close to β, the adjustment effect of attack tolerance parameter is more obvious. In **Fig. 7(b)**, the inhibition curve has nothing to do with *AT*. The reason is that $\beta = 0.9$ at this time, which is larger than the detection result. Therefore, the network still adopts a linearly-increasing low-suppression mode, so the attack tolerance parameter does not work.

It should be noted that the average residual energy *RE* is continuously decreasing when calculating the penalty-probability, so the penalty factor is also dynamically changing. The penalty factors calculated at times 9*T*, 10*T*, and 11*T* in **Fig. 7(a)** are 0.234, 0.219, and 0.204, and the probabilities are 0.95, 0.96, and 0.97, respectively. In summary, the probabilistic flow suppression component could dynamically adjust the suppression probability according to the degree of attack of the network, effectively restrain the illegal Packet-in flow in the network and could reasonably reduce the amount of discarded legal Packet-in flow by adjusting the attack tolerance, thereby effectively resisting saturation attack.

7. Conclusion

For the control plane saturation attack in SDWSN, this paper proposes a DDoS attack prevention extension FuzzyGuard including components: control network construction, fuzzy attack detection and probabilistic flow suppression. By routing the control flow independently, the control network could protect the forwarding of legitimate data flows while merging attack flows. On this basis, the fuzzy attack detection component obtains the attack probability through fuzzy inference. Probabilistic flow suppression component adopts different suppression modes based on the above probability to deal with Packet-in flows in the network. It is worth mentioning that prior to the introduction of FuzzyGuard, an easy-to-implement attack method is given and a threat analysis is conducted. Finally, based on SDN-WISE, a prototype of FuzzyGuard is implemented and a simulation network is deployed. The results show that FuzzyGuard could effectively detect saturation attack in different modes while protecting the normal forwarding of data flows under attack. In addition, it is able to reasonably adjust the Packet-in flows in the network, prevent rapid saturation of the control plane, and effectively enhance the security performance of SDWSN.

References

[1] L. M. Borges, F. J. Velez and A. S. Lebres, “Survey on the characterization and classification of wireless sensor network applications,” *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 1860-1890, 2014. Article (CrossRef Link).

[2] M. Huang, B. Yu and S. Li, “PUF-assisted group key distribution scheme for software-defined wireless sensor networks,” *IEEE Communications Letters*, vol. 22, no. 2, pp. 404-407, 2018. Article (CrossRef Link).

[3] Á. L. V. Caraguay, A. B. Peral, L. I. B. López and L. J. G. Villalba, “SDN: Evolution and opportunities in the development IoT applications,” *International Journal of Distributed Sensor Networks*, vol. 10, no. 5, pp. 735-142, 2014. Article (CrossRef Link).

[4] N. McKeown, “Software-defined networking,” *INFOCOM keynote talk*, vol. 17, no. 2, pp. 30-32, 2009. Article (CrossRef Link).

[5] T. Luo, H. P. Tan and T. Q. Quek, “Sensor openflow: Enabling software-defined wireless sensor networks,” *IEEE Communications Letters*, vol. 16, no. 11, pp. 1896-1899, 2012. Article (CrossRef Link).

[6] G. Li, S. Guo, Y. Yang and Y. Yang, “Traffic load minimization in software defined wireless sensor networks,” *IEEE Internet of Things Journal*, vol. 5, no. 3, pp.1370-1378, 2018. Article (CrossRef Link).

[7] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford and J. Turner, “OpenFlow: Enabling innovation in campus networks,” *Acm Sigcomm Computer Communication Review*, vol. 38, no. 2, pp. 69-74, 2008. Article (CrossRef Link).

[8] D. Kotani and Y. Okabe, “A packet-in message filtering mechanism for protection of control plane in OpenFlow switches,” *IEICE TRANSACTIONS on Information and Systems*, vol. 99, no. 3, pp. 695-707, 2016. Article (CrossRef Link).

[9] S. Shin, V. Yegneswaran, P. Porras and G. Gu, "AVANT-GUARD: Scalable and vigilant switch flow management in software-defined networks," in *Proc. of the 2013 ACM SIGSAC conference on Computer & communications security*, pp. 413-424, November 4-8, 2013. Article (CrossRef Link).

[10] R. Braga, E. Mota and A. Passito, "Lightweight DDoS flooding attack detection using NOX/OpenFlow," in *Proc. of the 35th conference on local computer networks (LCN)*, pp. 408-415, October 11-14, 2010. Article (CrossRef Link).

[11] H. Wang, L. Xu and G. Gu, "Floodguard: A dos attack prevention extension in software-defined networks," in *Proc. of the 45th annual IEEE/IFIP conference on dependable systems and networks (DSN)*, pp. 239-250, June 22-25, 2015. Article (CrossRef Link).

[12] N. Abdolmaleki, M. Ahmadi, H. T. Malazi and S. Milardo, "Fuzzy topology discovery protocol for SDN-based wireless sensor networks," *Simulation Modelling Practice and Theory*, vol. 79, pp. 54-68, 2017. Article (CrossRef Link).

[13] T. Huang, S. Yan, F. Yang and J. Liu, "Multi-domain SDN survivability for agricultural wireless sensor networks," *Sensors*, vol. 16, no. 11, pp. 1861-1874, 2016. Article (CrossRef Link).

[14] Y. Bangash, L. Zeng, S. Deng and D. Feng, "LPSDN: Sink-node location privacy in wsns via SDN approach," in *Proc. of the IEEE conference on networking, architecture and storage (NAS)*, pp. 1-10, August 8-10, 2016. Article (CrossRef Link).

[15] M. Ambrosin, M. Conti, F. De Gaspari and R. Poovendran, "Lineswitch: Efficiently managing switch flow in software-defined networking while effectively tackling dos attacks," in *Proc. of the 10th ACM symposium on information, computer and communications security*, pp. 639-644, April 14-17, 2015. Article (CrossRef Link).

[16] T. Wang and H. Chen, "SGuard: A lightweight SDN safe-guard architecture for DoS attacks," *China Communications*, vol. 14, no. 6, pp.113-125, 2017. Article (CrossRef Link).

[17] X. Wang, M. Chen, C. Xing and T. Zhang, "Defending DDoS attacks in software-defined networking based on legitimate source and destination IP address database," *IEICE TRANSACTIONS on Information and Systems*, vol. 99, no. 4, pp. 850-859, 2016. Article (CrossRef Link).

[18] A. Saied, R. E. Overill and T. Radzik, "Detection of known and unknown DDoS attacks using artificial neural networks," *Neurocomputing*, vol. 172, pp. 385-393, 2016. Article (CrossRef Link).

[19] S. M. Mousavi and M. St-Hilaire, "Early detection of DDoS attacks against SDN controllers," in *Proc. of the International conference on computing, networking and communications (ICNC)*, pp. 77-81, February 16-19, 2015. Article (CrossRef Link).

[20] Q. Yan, Q. Gong and F. Deng, "Detection of DDoS attacks against wireless SDN controllers based on the fuzzy synthetic evaluation decision-making model," *Ad Hoc Sensor Wireless Networks*, vol. 33, pp. 275-299, 2016. Article (CrossRef Link).

[21] H. Dai, Q. Wang, D. Li, R. C. W. Wong, "On eavesdropping attacks in wireless sensor networks with directional antennas," *International Journal of Distributed Sensor Networks*, vol. 9, no. 8, pp. 760834, 2013. Article (CrossRef Link).

[22] S. H. Jokhio, I. A. Jokhio and A. H. Kemp, "Node capture attack detection and defence in wireless sensor networks," *IET Wireless Sensor Systems*, vol. 2, no. 3, pp. 161-169, 2012. Article (CrossRef Link).

[23] L. Galluccio, S. Milardo, G. Morabito and S. Palazzo, "SDN-WISE: Design, prototyping and experimentation of a stateful SDN solution for WIreless SEnsor networks," in *Proc. of the Computer Communications (INFOCOM)*, pp. 513-521, April 26-May 1, 2015. Article (CrossRef Link).

Meigen Huang was born in Hunan, P. R. China, in December 1990. He received his B.S. degree and M.S. degree in the Zhengzhou Information Science and Technology Institute in 2013 and 2016, respectively. He is currently studying the Ph.D. degree in the Department of Computer Science and Information Engineering at Zhengzhou Information Science and Technology Institute. His research interests are wireless sensor networks, software-defined networking and network security.

Bin Yu received his B.S. degree in Dept. of Electronic Engineering from the University of Shanghai Jiaotong in 1986, the M.S. degree from South China University of Technology in 1991 and the Ph.D. degree in 1999. Currently, he is a professor of the Department of Computer Science and Information Engineering at Zhengzhou Information Science and Technology Institute, P. R. China. His research interests include the design and analysis of algorithms, wireless sensor networks, network security and visual cryptography.