Hindawi
Security and Communication Networks
Volume 2022, Article ID 1639203, 13 pages
https://doi.org/10.1155/2022/1639203

Research Article

A Low-Overhead and High-Precision Attack Traceback Scheme with Combination Bloom Filters

Jie Ma, Wei Su, Yikun Li, and Fangtao Yao

Beijing Jiaotong University, Beijing, China

Correspondence should be addressed to Jie Ma; 20111028@bjtu.edu.cn

Received 26 June 2022; Revised 2 September 2022; Accepted 24 September 2022; Published 13 October 2022

Academic Editor: Konstantinos Fysarakis

Distributed denial of service attacks seriously threatens the availability of highly resilient software-defined networking systems, such as data center networks. A traceback scheme is an effective means of mitigating attacks by identifying the location of the attacker and the attacking path. However, traditional traceback schemes suffer from low traceability success rates, high packet header overheads, and high communication traffic overheads, in addition to the fact that logically centralized traceability schemes make the control plane a prime target for attacks. To overcome the above challenges, we propose the low-overhead and high-precision traceback scheme, which is divided into two stages: packet marking and path reconstruction. The first stage of the traceback scheme utilizes programmable switches in the data plane to selectively mark the actual physical path information that the packet was forwarded on. The marking method is adaptive to the path length, which utilizes a combined Bloom filter so that the packet length does not grow with the length of the attacking path. The proposed probabilistic packet marking algorithm effectively reduces the number of packets collected to reconstruct the attacking path. The second stage of the traceback scheme utilizes the distributed victim host to reconstruct the attacking path without the controller and locate the source of the attacker. Theoretical analysis and experimental results show that the proposed scheme ensures the high accuracy of tracing and minimizes the traffic overhead and storage overhead required for the traceback process.

1. Introduction

Distributed denial of service attacks is the main means used by attackers and a major threat to the security of the Internet. The number of DDoS attacks has grown by 200% every year, causing losses of up to $100,000 an hour for the targeted service providers [1]. The problem of DDoS attacks is becoming so pervasive that researchers are motivated to find a reliable method of identifying the origin of attackers. Internet Protocol (IP) traceback is a method of reconstructing the attacking path and identifying the source of the attacker, with the message provided by the network devices where the attacking packets are forwarded. However, the effectiveness of traditional traceback schemes is affected by different problems, such as IP spoofing, packet length limits, and additional communication overhead [2–5]. Among them, IP spoofing means attackers often forge the source IP address of packets, resulting in low traceability accuracy. The number of marked path messages is limited by the length of the packet header field. Internet Control Message Protocol-based technologies need to send additional trace messages, imposing additional overhead on the communication channel. Furthermore, the operability of these schemes based on traditional IP network architectures is generally limited to the traditional network infrastructure, such as forwarding equipment with high storage space, computing power, and the ability to tag packets. The cost of replacing new hardware equipment is too high to implement, which limits the use of most IP tracking technologies.

Stanford University proposed the software-defined network (SDN) in 2008 [6, 7], which is a centralized, hierarchical architecture that separates forwarding and control. The SDN paradigm can be deployed in traditional SDN network architectures, such as highly resilient data center networks [8, 9], or new complex dynamic network architectures, such as SDN-based heterogeneous vehicular

networks [10–14]. SDN decouples the data plane and control plane of network devices, making the data plane devices dumb, and it places the control plane logically centralized at the network controller. The unique information architecture provides scholars with a new idea of IP traceback. SDN-based logging traceback schemes [15–17] utilize the control plane to centralize DDoS attacks and identify the attackers. However, there is a key challenge in this scenario, where the existing traceability tasks are overly focused on the control plane. It is a serious burden for the controllers to handle illegal data packets, store log information, and maintain the routing table for dynamic links. As a result, the control plane is more likely to be a prime target for attacks. Once attacks threaten the control plane, the stability of the entire network is greatly compromised, and the traceability scheme is virtually ineffective.

We propose a new traceback scheme aimed to overcome the above-mentioned challenges, including identifying the attack sources with high accuracy and reducing its over-reliance on the control plane for highly resilient software-defined network architectures such as data center networks. The overall traceback scheme is divided into two stages: packet marking and path reconstruction. The first stage of the traceback scheme utilizes the programmable switch in the data plane to collect the actual physical path information of packet forwarding and mark them as the tracing information, which can solve the problem of IP spoofing in the traditional tracing task. We consider that the packet field length is limited and that it does not grow with the attacking path length. For this reason, we develop a programmable field based on a combination of Bloom filters, which provides great flexibility for encoding various traceback information to be adaptive to the path length. The new algorithm for probabilistic packet marking proposed in this paper effectively reduces the number of packets collected to reconstruct the attacking path. The second stage of the traceback scheme is to reconstruct the attacking path on the victim host. Then, the victim host completes the task of locating the network attacker directly without the controller.

The following are the overall contributions of this paper:

(i) The proposed traceback scheme is a fine-grained traceability scheme with high accuracy and low traffic overhead, which can be deployed in the existing SDN frameworks.

(ii) The problem of packet header length being sensitive to path length is overcome by using a programmable combination of Bloom filters. Packets are designed to be adaptive to the attacking path length. Then, the packets can carry more marked path information while saving significant space in terms of memory space.

(iii) Compared to existing SDN-based traceback schemes, the dynamic probabilistic packet marking (PPM) methods reduce the expected number of packets required to reconstruct the attacking path, speed up traceability convergence time, and reduce traceability traffic overhead.

(iv) The design of the path identifier field improves the accuracy of the traceback scheme and supports the identification of multiple attackers outside the network.

(v) The experimental results show that our scheme is not overly focused on the SDN control plane, which makes it a more stable solution. The marking part is done with the help of the SDN data plane, and the path reconstruction part is done at the victim host.

The rest of the paper is organized as follows: Section 2 summarizes related work to the different attacking traceback schemes based on the traditional IP network and SDN. Section 3 describes our design principles, system architecture, and the details of our proposed scheme. Section 4 presents the implementation details of our scheme and evaluates its specific techniques. Finally, conclusions and future work are drawn in Section 5.

2. Related Work

As introduced previously, the first step in holding the attacker accountable is to identify the source of the attacking packets. Attack traceback is a means of defending against serious DDoS attacks. After calculating the attacking path followed by the attacking flow, the victim takes effective physical defensive measures. Traditional IP traceback schemes are well divided into six categories. (1) Link testing [18, 19] is often used to trace back the attacking path and the attacking source in real time. (2) The packet marking methods [2, 3, 20] are characterized by inserting the required traceability data into the IP packet to be traced, thus marking the packet as it travels through the forwarding network facilities (switches or routers) to the destination host. The packet marking schemes only use the information embedded in packets without generating additional traffic. The packet marking methods include PPM methods, deterministic packet marking (DPM) methods, and algebraically encoded packet marking methods. (3) ICMP-based technology [4] is a scheme in which routers generate ICMP messages to transmit packets with very low probability. The messages are sent to the attacking host or the victim host. (4) Logging-based schemes [5, 21, 22] utilize the packet digests or signatures stored on network devices to identify the hop-by-hop log information of the routers in the attacking path and then determine the closest router to the attacker. (5) Finding the source of forged IP datagrams in a large high-speed network is difficult due to the design of the IP protocol and the lack of sufficient capacity in most high-speed, high-capacity router implementations. An overlay network for IP traceback was proposed to solve this problem. CenterTrack [23] is a typical scheme to implement IP traceback with overlay networks and existing techniques (e.g., tunneling and input debugging). (6) Hybrid IP traceback schemes combine packet tagging and logging-based schemes. A high-precision single-packet IP traceback (HPSIPT) scheme [24] is a typical hybrid traceback scheme.

Table 1: Comprehensive comparative analysis of attack traceback techniques.

Category	Typical technologies	Storage overhead	Computing overhead	ISP cooperation	False positive ratio	False negative ratio	Traceback time
Link testing	Ingress filtering	Low	Low	Collaborative	High	Low	Long
Logging-based schemes	SPIE	High	Low	Collaborative	Low	Low	Middle
Packet marking methods	PPM	Low	High	Noncollaborative	High	Low	Long
ICMP-based technologies	iTrace	Low	Low	Collaborative	High	High	Long
Overlay network for IP traceback	CenterTrack	High	Low	Collaborative	Low	Low	Short
Compounded IP traceback	HPSIPT	Low	Low	Collaborative	Low	Low	Middle
SDN-based logging traceability techniques	SDN traceroute	High	Low	Collaborative	Middle	Middle	Middle
SDN-based logging traceability techniques	SDN selective logging method	Middle	Low	Collaborative	Middle	Middle	Middle
SDN-based traceability techniques	SMITE	Low	Middle	Collaborative	Low	Middle	Middle
SD-WAN scenarios traceback schemes	Cross-domain traceback scheme	Low	Middle	Collaborative	Low	Low	Middle

Hence, the traceback schemes discussed in this section exploit the features of the SDN paradigm to reconstruct the attacking path and identify the attacker source with the help of the SDN control plane.

Agarwal et al. [15] proposed the SDN traceroute that utilizes the forwarding mechanism of SDN to trace the forwarding path of packets through the network by sending probe messages with a specific tag and comparing the probe messages hop by hop to trace back to the source. Hadem et al. [17] introduced an intrusion detection system (IDS) using the Support Vector Machines (SVM) along with selective logging for IP traceback based on the SDN. Handigol et al. [25] proposed the Netshark tool, a tool similar to Wireshark that allows users to set filters on the entire packet history, recording their path and packet header information at each hop. Users can view packet attributes at a particular hop, thus enabling the tracing of attack packets. Ren et al. [26] proposed a global flow table algorithm based on SDN, which periodically traverses all switches to obtain the flow table through the controller interface, maintaining each flow in the SDN and enabling the tracing of abnormal traffic by analyzing the global flow table. Zhang et al. [27] suggested utilizing the controller language to complete packet traceback, predicting packet action expressions based on the current packet processing policy, and further calculating all preceding forwarding policies for any packet. Francois et al. [28] proposed an SDN-based anonymous IP traceback method, which models the SDN network topology with a directed graph and maintains the information of the flow table in this graph. Hadem et al. [29] proposed a traceback method based on SDN and multiprotocol label switching (MPLS).This method uses the MPLS technology to design a short path flag to represent the attack path information, which requires only tens of bits and maintains a mac table and arp table to record the attack path information. Sahay et al. [30] proposed a method by marking the packet header of each data stream in the network for identification purposes. When a network anomaly is detected, its source is identified by looking at the label of the anomalous stream. The tag consists of, for example, source IP information obtained by the switch. The reliability of the tag information is reduced when there is spoofing of the source IP address. Sahay et al. [31] addressed the problem of traceability tracing in a multicontroller scenario by dealing with traffic identification in ISP networks. Dayal and Srivastava [16] solved the problem of cross-domain and intradomain traceability for SD-WAN scenarios.

According to the above six common evaluation indexes, we compare the ten representative traceback schemes (ingress filtering [18], SPIE [22], PPM [20], iTrace [4], CenterTrack [23], HPSIPT [24], SDN traceroute [15], SDN selective logging method [17], SMITE [29], and cross-domain traceback scheme [16]). The comparison results are listed in Table 1. According to Table 1, there is no method whose six evaluation indexes are all the best or all the worst. Logging-based schemes and the SDN-based logging traceability techniques are storage overhead techniques, packet marking methods are the typical computing overhead techniques, and compounded IP traceback is a compromise between storage and computing overheads. SDN-based traceback schemes are suitable for new network architectures, while other traceability techniques are designed based on traditional network architectures. Where traceability tasks require the location of attacks across different autonomous domains, only packet marking methods do not require the support of network service providers (ISPs). This means that only one such technique is feasible for forensics in certain demanding network environments. The pros and cons of a method depend on the method itself and the requirements of end users.

As mentioned above, the traditional schemes are outstanding in their own respects; however, they have several drawbacks. Our proposed scheme significantly reduces the

reliance on the control plane. Traceback tasks that were overly concentrated in the control plane are broken down and decentralized to the data plane and victim hosts. Each terminal that accesses the network is used to prestore topology information, which is a virtual computation independent of the underlying SDN infrastructure, eliminating the additional information exchange and storage overhead required for network transmission. In addition, a new dynamic probabilistic marking method in the traceback scheme reconstructs the attacking path to minimize the traffic overhead instead of setting the same marking probability method for every switch in the network.

3. Proposed Scheme

The traceback scheme described in this paper consists of two parts: the marking and the path reconstruction processes. In the first part of the marking process, the programmable switch in the data plane marks path information related to its entire address into the attacking packet with a certain probability. The encoded path information consists of the physical information of the switch ID, switch order, and port number. In the second part of the trancing procedure, the marked path information carried by the packet is decoded in the victim host. When the victim host receives packets, it decodes packet headers to reconstruct paths using the path reconstruction algorithm. Then, the source of the attackers is accurately identified.

3.1. Basic Assumptions. The notations used in this paper are listed in Table 2. The proposed traceback scheme is based on the following assumptions:

(i) The probability of the attacking threat to the programmable switch in the data plane is almost zero.

(ii) There is a risk of malicious tampering with the marked path information, but the probability is almost zero.

(iii) Multiple attackers can simultaneously attack the same victim host.

(iv) The controller of the control plane is informed of the network topology and sends flow mod messages to install flow entries on the programmable switches. In this way, the control plane does not interfere with the switches to automatically perform packet content updates and the switches actively write the path information into the packet.

Table 2: Notations used in this paper.

Notations	Interpretation
m_1	The length of BF1
m_2	The length of BF2
k_1	The number of hash functions in BF1
k_2	The number of hash functions in BF2
b_1	The number of programmable switches inserted into BF1
b_2	The number of programmable switches inserted into BF2
m_{pi}	The marking probability of the switch
d	The path length of the switch from victim host (in hops)
N	The total number of switches in the network topology
n	The constant value is equal to N in most cases
i	The path length of the marked switch from the attack source (in hops)
i^*	The path length of the marked switch to the victim host (in hops)
c	The constant value

3.2. Problem Statement. An actual network topology is illustrated in Figure 1. It is observed that multiple attackers attack the victim host, host 5, from host 1 and host 2, where there are two possible attacking paths: ($S6 \longrightarrow S3 \longrightarrow S1 \longrightarrow S5 \longrightarrow S10$) and ($S2 \longrightarrow S3 \longrightarrow S1 \longrightarrow S5 \longrightarrow S10$).

The problem of finding the source of the attacker is to reconstruct the attacking path with the collected path information. In most cases, the packets that attack the victim host are sent by different hosts. In addition, the path between the same pair of the attacking and victim hosts changes in real time. In this process, the switch IDs on the forwarding path are linked together with their orders to constitute the corresponding path segment. Each forwarded packet carries the information of the switch ID, the switch order, the ingress port ID, and the egress port ID gathered into a path segment label, which consists of several path segment labels. We also introduce the path identifier in the packet header design to improve the accuracy of the traceback scheme, identify different attackers, and authentically reconstruct the fine-grained forwarding path. The switch ID, the switch order, the ingress port ID, and the egress port ID are the information on the real physical path written directly from the switch, avoiding the problem of low traceability success rate caused by DDoS attacks forging path information. The attacking path label is formally expressed as follows:

$$\text{Attacking path} = \begin{bmatrix} \text{path identifier} \\ \langle \text{SwitchID}_1, \text{Order}_1, \text{IngressPortID}_1, \text{EgressPortID}_1 \rangle \\ \cdots \\ \langle \text{SwitchID}_n, \text{Order}_n, \text{IngressPortID}_n, \text{EgressPortID}_n \rangle \end{bmatrix}^{\text{T}}, \quad (1)$$

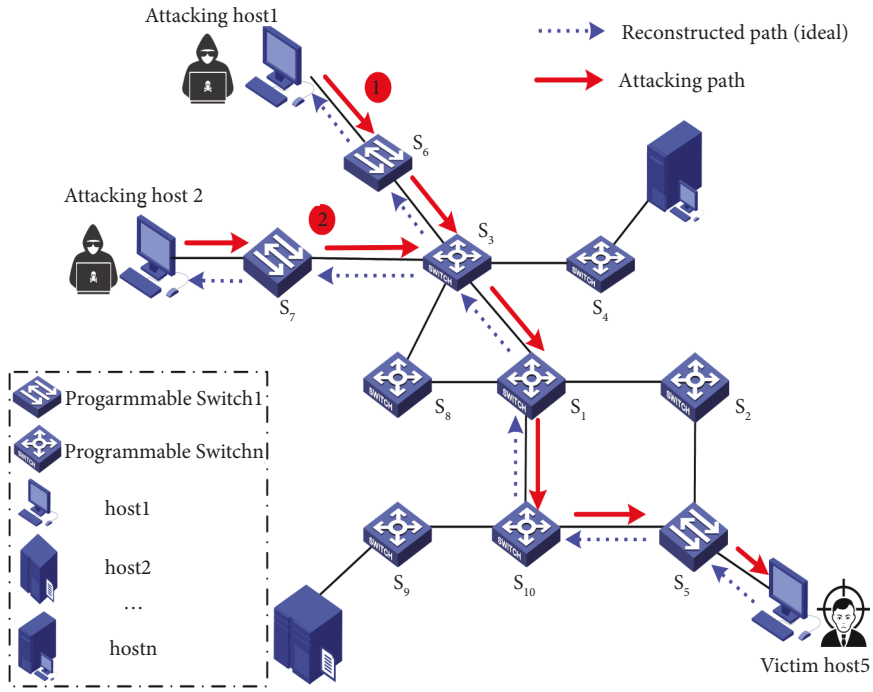

Figure 1: Traceback problem model.

The above equation represents the fine-grained attacking path of a packet forwarded with the assistance of the path identifier from the ingress port of the first switch to the egress port of the last switch.

3.3. Advanced Marking Methods. In the first part of the marking procedure, the packet header length is designed to utilize a programmable combinatorial Bloom filter that does not limit the packet header to populate complete traceability information due to changes in the actual topological path length for adaptive path length. The probabilistic packet marking algorithm can effectively reduce the number of marked packets when reconstructing the attacking path.

3.3.1. Marking Mechanism Based on a Combination of Bloom Filters. The Bloom filter is a random data structure with high spatial efficiency. During the marking procedure, the fast insertion and query functions provided by the combined Bloom filter are used to complete the tagging task for different path lengths, improving the coding efficiency and reducing the high cost associated with storing the entire attacking path in multiple packets.

We have transformed the path information storage and reduction problem into a set of key-value pair storage and query problems. The packet-encapsulated traceback information field contains four types of information about the switch ID, switch order, ingress port ID, and egress port ID. Then, the switch ID and the ingress and egress ports ID are abstracted as keys in the Bloom filter and the order as values. For example, in Figure 2, the information for attacking path 1 can be represented by {S6 (1) (4), 1}, {S3 (4) (2), 2}, {S1 (6) (1), 3}, {S5 (3) (2), 4}, \\{S10 (5) (1), 5}, where S6 is the first hop, S3 is the second hop, S1 is the third hop, S5 is the fourth hop, and S10 is the fifth hop. At each hop in the network, the key-value pairs of path information are encapsulated in packets as they propagate across the network. When the packet reaches the victim host, it needs to recover not only the key but also the value to reconstruct the entire attacking path. To recover the key values and maintain the space efficiency of the Bloom filter, a combination of Bloom filters, such as BF1, BF2, and BFT, was chosen to store the key-value pairs. The process of storing and parsing the path information is shown in Figure 2. A programmable switch within the network encodes the path information in the packet header, and the victim host decodes and reconstructs the path. By considering the size of the existing network and the number of bits in the packet header option field, BF1 is designed to have an m_1 bit length and k_1 different hash functions. BF2 has an m_2 bit length and k_2 different hash functions. The information written to BF1 is about all the programmable switches through which the packet forwards. The information written in BF2 is about the number of b_2 switch labels (IDs and ports) through which the packet passes on the attacking path. The sequential number (order) is inferred by *len*, written by the switch directly to the traceback information field in the packet as it travels across the network. The algorithm for path information encoding is given in Algorithm 1.

3.3.2. Dynamic Probabilistic Packet Marking Algorithm. A key problem here is how to determine which switches are selected to achieve the fewest packets to encode different parts of the path with a high probability. The victim host combines this different path information and decodes the complete path. In order to solve this problem, traditional

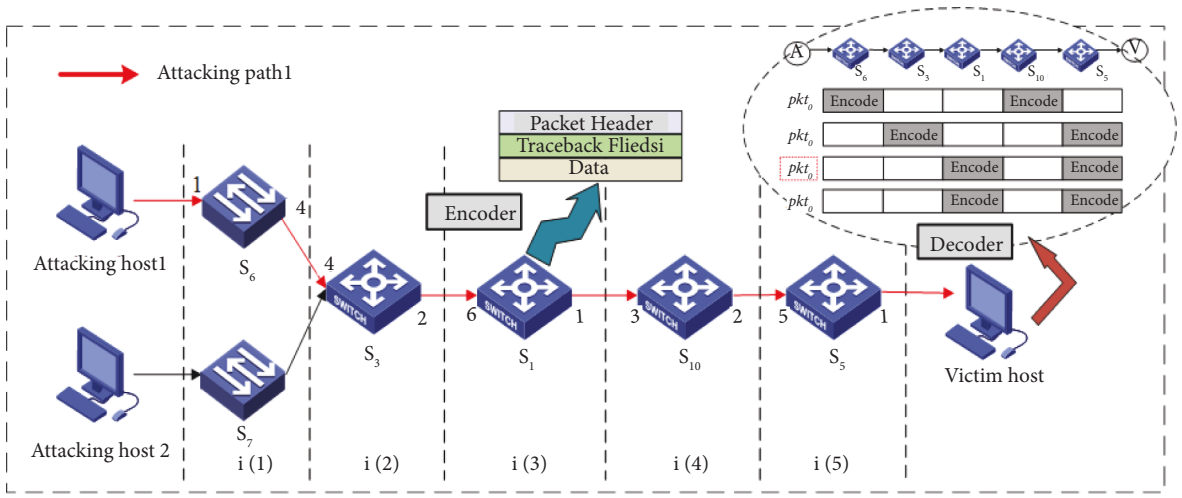

Figure 2: Traceback scheme model.

(1) **procedure**INFORMATION ENCODE (BF1, BF2, len)
(2) Key = SwitchID, IngressPortID, EgressPortID
(3) Value = len
(4) **For** $j = 1$ to k_1 **do**
(5) $i \leftarrow h_j$ (key, value)
(6) BF1 $(i) = 1$
(7) **End for**
(8) x = DynamicProbabilityMarking (switch)
(9) **If** value is in x **then**
(10) **For** $j = 1$ to k_2 **do**
(11) $i \leftarrow h_j$ (key)
(12) BF2 $(i) = 1$
(13) **End for**
(14) **End if**
(15) len←len + 1
(16) **End procedure**

Algorithm 1: Path information encoding

traceback schemes based on SDN, overly dependent on the control plane, choose to use enough packets to reconstruct the entire attacking path, which also causes a large number of data storage resources. We transform this problem into a dynamic probabilistic packet marking problem.

In this problem, the convergence time of the algorithm performed by the network infrastructure is related to the number of packets required by the host victim to reconstruct the complete attacking path. The recovery of the complete attacking path is called convergence, and the convergence time originally represents the speed of route tracing. Then, it is defined by the minimum number of packets required for convergence to measure the performance of the traceback scheme.

In the traditional deterministic marking method, each router explicitly marks data packets by appending its label to the IP packet field. However, the use of this method results in significant traffic overhead. Unlike traditional deterministic marking methods, the dynamic PPM methods can enhance the probability of a packet marked by setting a certain router marking probability and reserving space in the packet for a router label. In this paper, the PPM method is redefined that each packet in the SDN only needs to be written to the programmable switch label once to mark the attacking path without abnormal packet loss. Analyzing the problem from an attacker's perspective, the packets that are further away from the victim host are more likely to be dropped or overwritten in the log store. Therefore, in order to improve the marking rate of the access switch, the function is set in this paper to be the probability of a packet marked by the switch:

$$p_{\text{mark}_i}(d) = m_{pi} \prod_{j=i+1}^{d} \left(1 - m_{pj}\right). \tag{2}$$

The minimum number of packets to be obtained (desired value) is as follows:

$$E[P] = \frac{\ln(d)}{m_{pi} \prod_{j=i+1}^{d} \left(1 - m_{pj}\right).} \tag{3}$$

In our scheme, the marking probability of the programmable switch is set as follows:

$$m_p(i) = i^{-c}, \tag{4}$$

where c is a constant given according to the actual network topology. The proposed dynamic probabilistic packet marking method differs from the static probabilistic packet marking method (ST) and the traditional dynamic probabilistic packet marking method (DY), where the marking probability in the static probabilistic packet marking method is as follows:

$$m_{p-\text{ST}}(i) = n^{-1}. \tag{5}$$

The marked probability in the traditional dynamic probabilistic packet labeling method (DY):

$$m_{p-\text{DY}}(i^*) = (i^*)^{-1}. \tag{6}$$

According to equations (2) and (4), the value of d determines the value of the marking probability m_{pi} during the

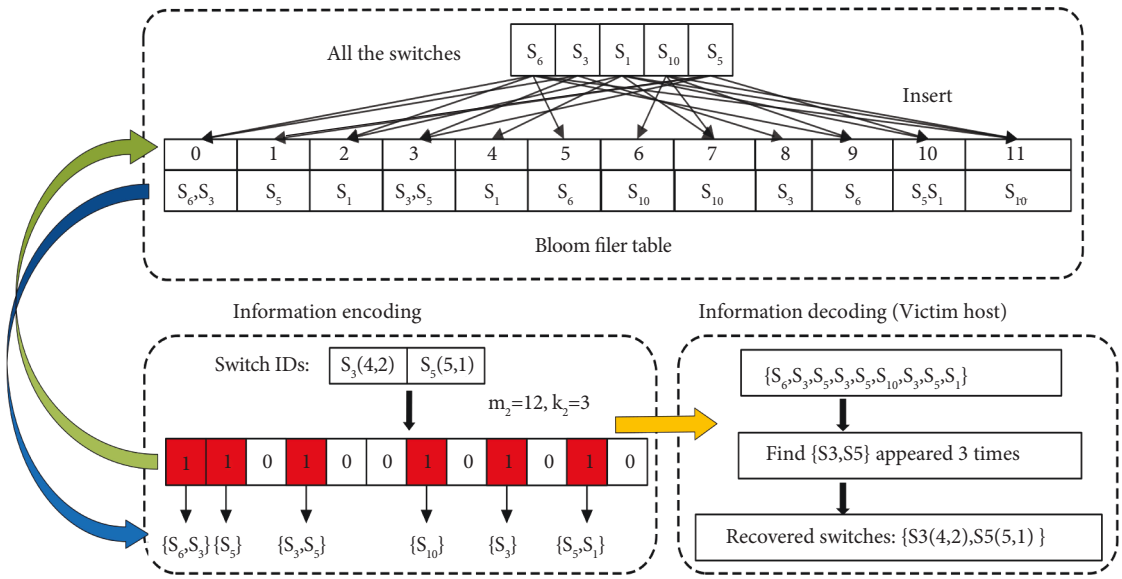

Figure 3: Utilization of BF and BFT to recover the information.

marking process, which means that the marking value increases as the value of d increases. This design of the probabilistic algorithm is superior to traditional marking methods because it increases the number of packets carrying marking information to the access switch, effectively reducing the number of packets stored on the victim host that are overwritten. To figure out the number of packets needed to reconstruct a meaningful attacking path, we need to get the expected value. The probability value of the packet with the farthest switch label known to be received is the lowest, and the convergence time of the marking process depends mainly on the distance of the tagged packet to the victim. The probabilistic marking problem discussed is a classic coupon collection problem. The coupon collection problem refers to repeatedly taking an object out of a set of objects and then finding the number of times that all the objects have to be taken out at least once. We use the probability property that the expectation of the sum of random variables is equal to the sum of the expectations of these random variables. The expectation of the coupon collection problem is thus obtained as $k \cdot (\ln(k))$. By doing so, we can conclude that the expected number of packets required to construct a meaningful attack graph is $E[P] < \ln(k) - k^n$.

3.4. Path Reconstruction Method. In the second part of the path reconstruction process, each programmable switch encodes the path information into the packet header, and the victim host decapsulates the packet to restore the information in the traceback information field carried by the packet. Then, the information is used to restore the attack path segments. The entire attacking path is reconstructed through several attacking path segments.

A Bloom filter table is preset in the network access terminal. The BFT design idea is to prewrite all switch information in the network topology into the BFT of the victim host, which greatly reduces the delay consumption in the decoding process. We set the parameters of the Bloom filter table and BFT2 to the same parameters. Each cell in the Bloom filter table stores a collection of all switch IDs and switch port numbers that have been hashed to this location. Insert All switch IDs and switch port numbers in the topology into the Bloom filter table and obtain a set containing the two pieces of information after searching the Bloom filter table. The victim host first recovers the ID and port number of the switch inserted into BF2. When the packet arrives at the victim host, check which locations in BF2 are set to 1. If the set of switch ID and switch port number appears k times, we believe that the two have been inserted into the Bloom filter table; that is, the packet is forwarded through the switch. As shown in Figure 3, positions "0," "1," "3," "6," "8," and "10" in the Bloom filter table are set to "1." After searching the Bloom filter table, the victim host gets a set composed of $\{S6, S3, S5, S3, S5, S10, S3, S5, S1\}$, where $\{S6, S3\}$ appears three times, which is the number of hash functions used in BF2 and the Bloom filter table. Therefore, we believe that $\{S6(1)(4), 1, S3(4)(2), 2\}$ is written in BF2. Select the $\{S6(1)(4), 1, S3(4)(2), 2\}$ collection to hash k times with the order information in the unpacked packet inside the victim host. The operation result is matched with the position of "1" in BF1, and the false positives caused by a collision are checked. If the position of "1" in BF1 is the same, the corresponding mapping relation of the attacking path segment can be established.

The victim host divides packets into different packet groups by path identifiers. Here, we use BF1 as the identifier for each path. Establish the mapping between BF1 and path ID and then the mapping between path ID and path reconstruction result. When the victim host receives the packet, it first extracts BF1 in the packet header and queries the first mapping. If an invalid path ID is returned, the packet passes through a new forwarding path. Therefore, we create a new path ID and then assign it to the packet. Instead, we use this valid path ID to query the second map to obtain

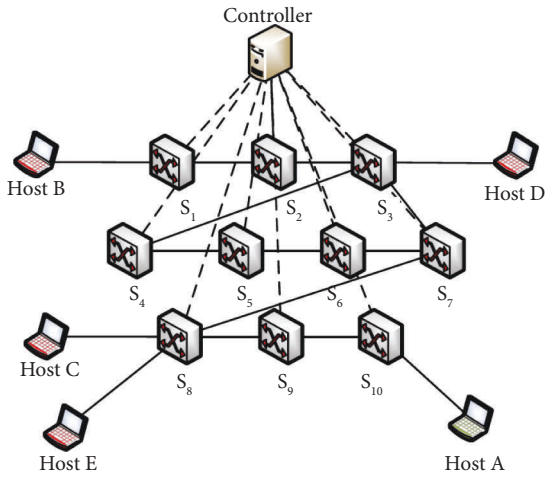

Figure 4: Topology of the simulation experiment.

Table 3: The setting of device ports and addresses.

Device	Networks port	Address	State
Controller	eth0	192.168.71.128	—
Switch 1	eth0, eth1, eth2	10.0.1.0	—
Switch 2	eth3, eth4, eth5	10.0.2.0	—
Switch 3	eth1, eth2, eth3, eth4, eth5	10.0.3.0	—
Switch 4	eth3, eth4, eth5	10.0.4.0	—
Switch 5	eth1, eth4, eth5	10.0.5.0	—
Switch 6	eth2, eth3, eth5	10.0.6.0	—
Switch 7	eth1, eth4, eth5	10.0.7.0	—
Switch 8	eth1, eth2, eth3, eth4, eth5	10.0.8.0	—
Switch 9	eth1, eth4, eth5	10.0.9.0	—
Switch 10	eth1, eth4, eth5	10.0.10.0	—
Host A	eth0	10.0.10.101	Normal
Host B	eth0	10.0.1.102	Malicious
Host C	eth0	10.0.8.103	Malicious
Host D	eth0	10.0.3.104	Malicious
Host E	eth0	10.0.8.105	Malicious

the previous path reconstruction results. The reconstructed result is updated with the decoded path segment. This way, decoding the desired number of packets in each path group restores the real attacking path.

4. Experimental Results and Analysis

4.1. Experimental Environment. The traceback scheme proposed in this paper was tested in the following simulation environment: the experimental environment is based on the Ubuntu operating system with an Intel(R) Core(™) i7-8700 CPU @ 3.20 GHz, using a VMware Workstation virtual machine with 8 GB RAM, 40 GB hard disk, and four core processors already configured. The network topology was simulated using the Mininet [32] simulation tool. Background traffic and DDoS attack traffic are simulated and generated by the Scapy library[33] and hping3[34] tool. Detailed data traffic was captured and analyzed using Wireshark software.

Mininet simulated a simple topology for the SDN network, as shown in Figure 4. The network topology consists of ten P4 programmable switches, five terminals, and a controller. We select Host B, Host C, Host D, and Host E as the attacking hosts and Host A as the victim host. The background traffic consisted of 75% TCP packets, 20% UDP packets, and 5% ICMP packets. The terminal of each host is opened on the Mininet side *via* the XTerm command. When the host terminal is turned on, a shell script is automatically started, which runs a background traffic generation code written in python to generate and send random packets to one of the hosts in the network to simulate the background traffic. The packet interval for DDoS traffic is set to 25 times the packet interval for background traffic. The relevant settings of the device are listed in Table 3.

In order to prove the authenticity and validity of the experimental topology of this scheme [35–38], we analyzed the topology of different existing networks, as shown in Figure 5, where Facebook Fabric Topology is the topology of the latest generation of the Facebook data center network. The Stanford Backbone network is the backbone topology of the Stanford University campus network. In contrast, the other three topologies are the average path length of traditional network IPv4 packets forwarded from the CAIDA host. Our topology is designed to meet the requirements of network topologies based on SDN architectures.

4.2. Effect of Bloom Filter Parameters. As each Bloom filter suffers from false positives, the false positives of the Bloom filter in our proposed scheme only exist when a wrong path may be constructed. By correctly setting the optimal BF1, the BF2 parameters reduce the minimum false alarm rate of the Bloom filter, improve the path restoration rate, and eliminate the excessive effect of introducing a bloom filter on the traceability success rate. We analyze how much information, such as switch ID, should be written to BF2 to ensure successful decoding by the victim host, depending on the false alarm rates of BF1 and BF2 in the scenario. The length of the Bloom filter depends on the available packet header space. We choose to use the 16-bit IP_ID and 12-bit VLAN_ID as the length of m_1 in BF1 and m_2 in BF2, respectively. Equations (7) and (8) represent the respective false alarm rates for BF1 and BF2:

$$P_{\mathrm{BF1}} = \left(1 - \left[1 - \frac{1}{m_1}\right]^{k_1 b_1}\right)^{k_1} \approx \left(1 - e^{-(k_1 b_1/m_1)}\right)^{k_1}, \quad (7)$$

$$P_{\mathrm{BF2}} = \left(1 - \left[1 - \frac{1}{m_2}\right]^{k_1 b_2}\right)^{k_2} \approx \left(1 - e^{-(k_2 b_2/m_2)}\right)^{k_2}. \quad (8)$$

According to the following equations [39], the optimal number of hash functions and the optimal number of switches written to the Bloom filter are determined to achieve the goal of the lowest false positive rate:

$$k_1 = \frac{m_1}{b_1}\ln 2, \quad (9)$$

$$k_2 = \frac{m_2}{b_2}\ln 2. \quad (10)$$

As the length of the attacking path from the victim increases, more packets are needed to reconstruct the path,

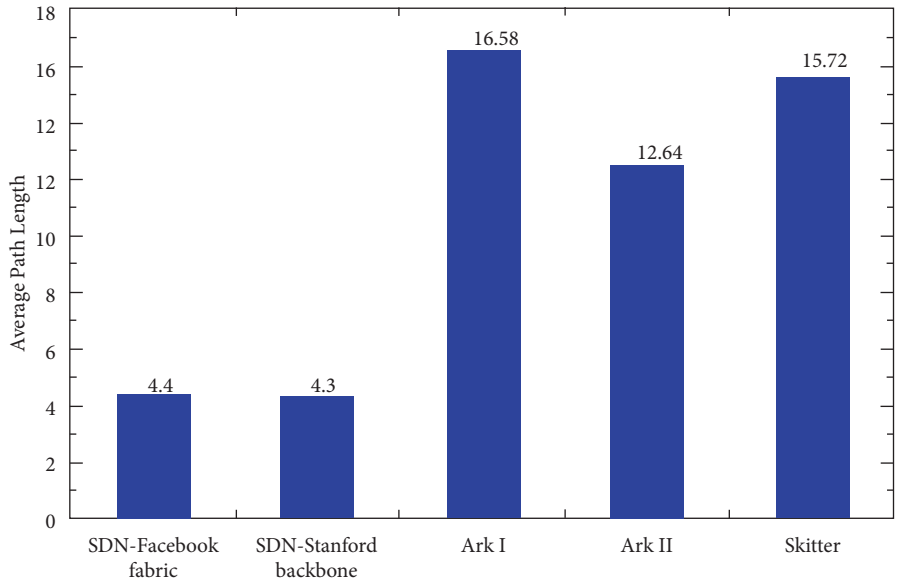

Figure 5: Path length distribution in different network topologies.

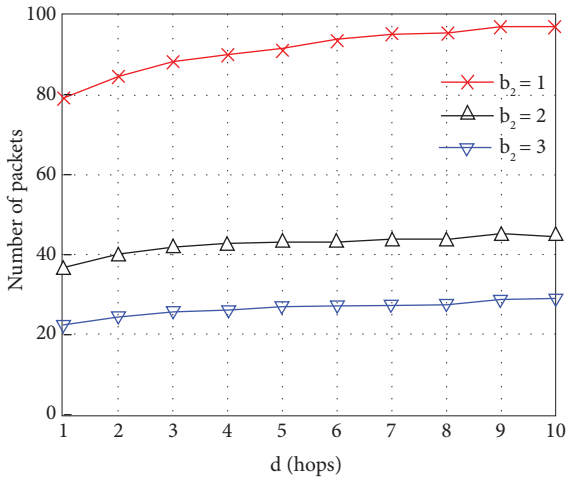

Figure 6: Number of packets needed in terms of d with different b_2.

as shown in Figure 6. We can observe that the optimal number of switches to insert into BF2 in order to ensure successful path recovery is either 1 or 2.

Here, the success rate of path information restoration can be defined by the following formula. When the victim host queries BF1, it generates a possible key-value pair. However, only one of these key-value pairs matches the one in BF1, which means that only this one has the probability of false positives, and the remaining key-value pairs have no false positives. Therefore, the failure probability of path reconstruction is shown as follows:

$$P_{\text{Failure}} = P_{\text{BF2}} * P_{\text{BF1}} * \left(1 - P_{\text{BF1}}\right)^{(b_2-1)}. \tag{11}$$

The success rate of path reconstruction is shown as follows:

$$\begin{aligned} P_{\text{Success}} &= 1 - P_{\text{Failure}} \\ &= 1 - \left[1 - \left(1 - \frac{1}{m_2}\right)^{k_2 b_2}\right]^{k_2} * \left[1 - \left(1 - \frac{1}{m_1}\right)^{k_1 b_1}\right]^{k_1} \\ &\quad * \left\{1 - \left[1 - \left(1 - \frac{1}{m_1}\right)^{k_1 b_1}\right]^{k_1}\right\}^{(b_2-1)}. \end{aligned} \tag{12}$$

When there are N switches in the network topology, the success rate of path information restoration can be expressed as follows:

$$P_{\text{Success}} = \left(1 - P_{\text{Failure}}\right)^{(N-b_2)}. \tag{13}$$

The above equation is consistent with the binomial distribution $e \sim B(N - b_2, P_{\text{Failure}})$. We plot the change in the success rate of path information restoration when the reduction is small, as shown in Figure 7. Note that when $P_{\text{Failure}} \leq 0.01$, the success probability is greater than 0.99. Furthermore, when $P_{\text{Failure}} \leq 0.001$, the success probability is greater than 0.9999.

The traceability scheme proposed in this paper is evaluated and analyzed by four evaluation indexes: success rate of traceability, traffic overhead, storage overhead, and controller memory occupancy.

4.3. Accuracy. The success rate of the traceability of the proposed scheme refers to the accuracy of identifying the switch closest to the attack source in the case of IP spoofing or multiple attack sources simultaneously attacking the victim host:

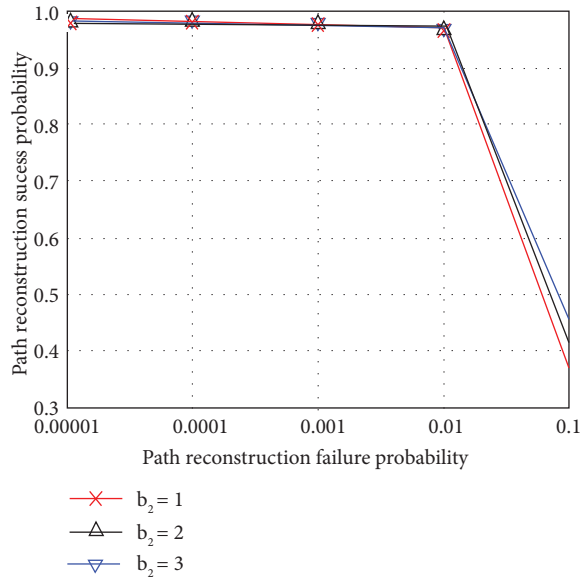

Figure 7: Attacking path reconstruction success probability with respect to path reconstruction failure probability.

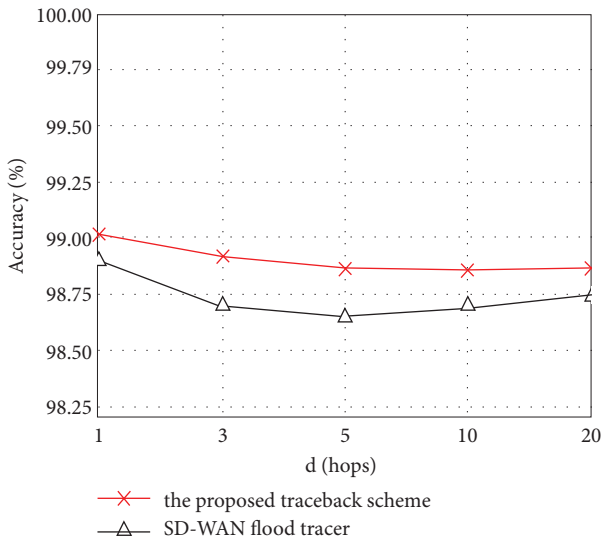

Figure 8: Comparison with the latest traceback scheme [16] in terms of traceability accuracy as the path length of the switch from victim host increases.

$$\text{Accuracy} = \frac{\text{Traceback}_{\text{correct}}(T_1)}{\text{Traceback}_{\text{total}}(T_1)} * 100\%. \quad (14)$$

$\text{Traceback}_{\text{event\_correct}}(T_1)$ refers to the number of times that the switch closest to the attacker is traced within a period of time. $\text{Traceback}_{\text{event\_total}}(T_1)$ is the total number of traceability performed within the preceding period. Under the above experimental conditions, the traceability accuracy of the proposed traceability scheme can reach about 98.93%, higher than that of the existing SDN technology-based traceability scheme, as shown in Figure 8. In the case of multiple-attack-source IP address spoofing, the attack sources located on attack paths of different lengths

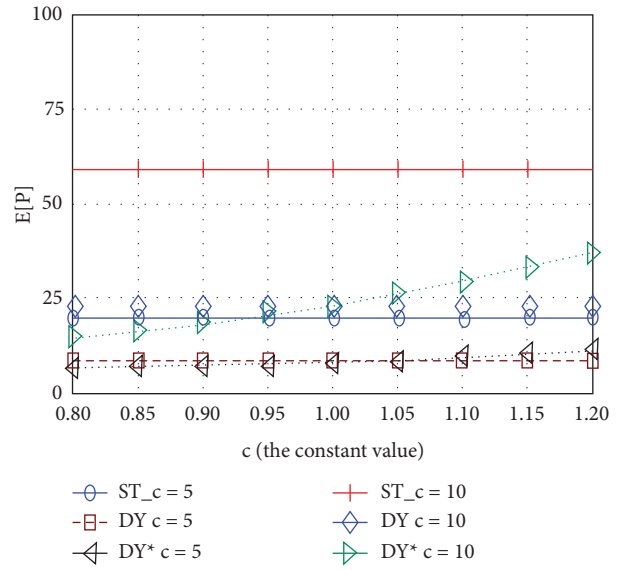

Figure 9: Trend of E[P] *versus* the constant value for static, dynamic, and the proposed marking methods.

can be traced accurately. The success rate of tracing is affected by actual operations, such as packet loss in links, packet fragmentation, repeated marking of packets, and overwriting of log records.

4.4. Traffic Overhead. Figure 9 compares trends in the packet expectations required to reconstruct meaningful attacking paths for different marking methods. When the value range of parameter C is set to [0.8, 1], the rule that each programmable switch has at least one flag is met. For different path lengths D, the new dynamic probabilistic packet marking scheme proposed by us has a better effect and a smaller expected value, which is superior to the traditional static and dynamic probabilistic packet marking methods.

When $c = 0.9$, we select three different attack path lengths. The ratio between the number of packets required by the recovery path and the success rate of attacking path reconstruction is shown in Figure 10. As expected, the longer the reconstituted attacking path length, the more packets are required. The proposed scheme to find the number of packets required by the attack source is far less than 100. The traffic cost is lower than other schemes.

4.5. Storage Requirement. If the number of packets required for the path reconstruction is n, the storage size S_1 required for logging our proposed mechanism is the amount of memory required for storing switch ID, switch order, ingress port, and egress port fields in addition to the fields normally required for traceback scheme as follows:

$$\begin{aligned} S_1 &= n_1 * [2 * \text{size}(\text{switch ID}) + 2 * \text{size}(\text{switch order}) \\ &\quad + 2 * \text{size}(\text{ingress port ID}) + 2 * \text{size}(\text{egress port ID})] \\ &= n_1 * (m_1 + m_2 + 8) = 36n_1 \text{ bits}. \end{aligned} \quad (15)$$

The required storage size S_2 for the same above-mentioned fields carried by the packets in the traditional scheme is shown as follows:

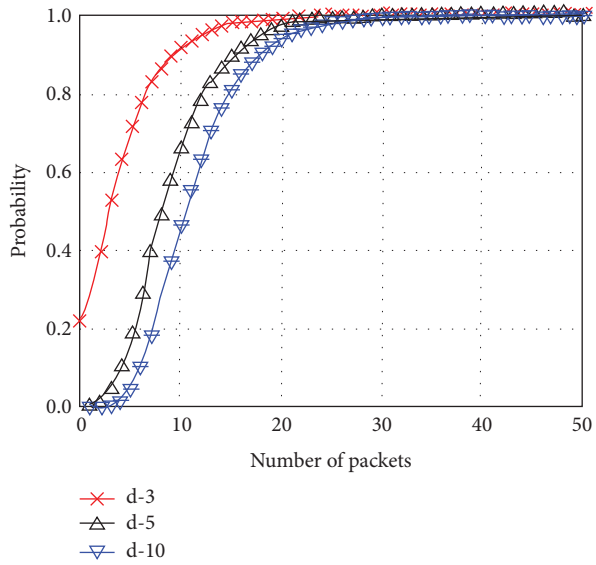

Figure 10: Variation of path reconstruction success probability as the number of packets increases.

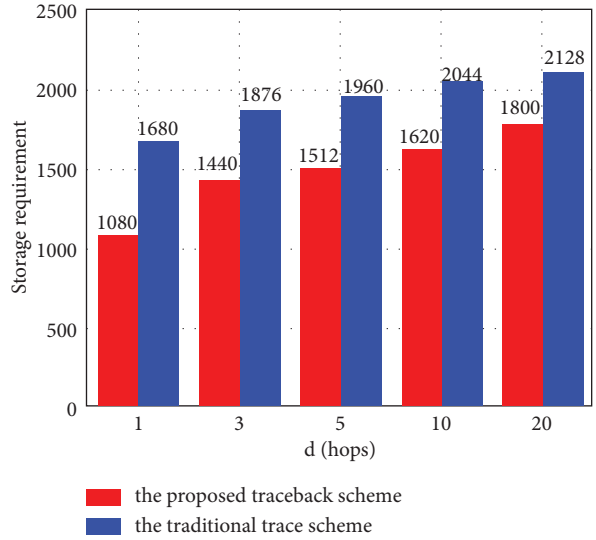

Figure 11: Variation of memory storage between our scheme and the traditional scheme as the length of the attack path increases.

$$\begin{aligned} S_2 &= n_2 * 2 * [\text{size}(\text{switch ID}) + \text{size}(\text{switch order}) \\ &\quad + \text{size}(\text{ingress port ID}) + \text{size}(\text{egress port ID})] \\ &= n_2 * 2 * (8 + 4 + 8 + 8) \\ &= 56 n_2 \text{bits}. \end{aligned} \tag{16}$$

Thus, the memory storage reduction in our proposed work taking the above scenario is shown in Figure 11, observing that, as the length of the attacking path increases in the topology, the storage overhead of the traceback information field in the proposed scheme is less than that of the same information field in the traditional traceback scheme.

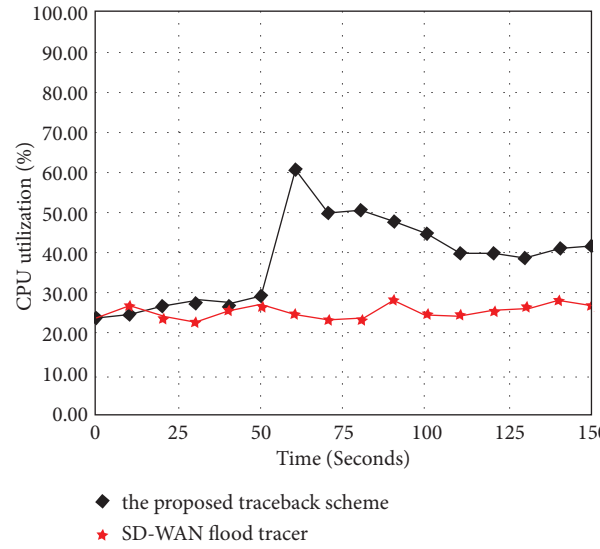

Figure 12: Comparison with the latest traceback scheme [16] in terms of controller CPU utilization.

4.6. Controller CPU Utilization. The controller CPU utilization indicates the CPU utilization of the ONOS controller during the tracing scheme and other tracing schemes. Figure 12 shows that the attack traffic is detected at T1, and the CPU usage of the controller changes with the tracing time after different tracing programs are started. The traceability scheme proposed in this paper does not depend on the controller. After the traceability task is started, the memory occupation of the controller will not surge over time. The stable fluctuation phenomenon indicates that the controller in the network only interacts with the data plane switch normally at this time.

5. Conclusions and Future Work

This paper presents a scheme for attacking path reconstruction and identification of attack sources based on a combined programmable Bloom filter and a new dynamic probabilistic packet marking algorithm. The method adaptively encodes the path information in two Bloom filters encapsulated in the packet header. The path information is converted into a set of key-value pairs where the switch ID, ingress port, and egress port are abstracted as keys in the Bloom filters and the order is abstracted as a value. The new dynamic probabilistic packet marking algorithm effectively reduces the number of packets collected to reconstruct the attacking pat. An experimental study implementing the traceback scheme in a network emulation system shows significant improvement in the performance of the low-overhead and high-precision traceback scheme compared to other reported works.

In future work, we plan to develop an improved packet traceability algorithm and combine this traceability algorithm with a large-scale autonomous domain incentive deployment algorithm to propose a traceability strategy in a federated model to perform cross-domain traceability more efficiently and reduce maintenance costs across autonomous domains.

Data Availability

The data used in this paper are available from the corresponding author upon reasonable request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Key RD Program of China (2022YFB2901600).

References

[1] S. Mohammed, R. Hussain, O. Senko et al., "A new machine learning-based collaborative ddos mitigation mechanism in software-defined network," in *Proceedings of the 2018 14th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pp. 1–8, Limassol, Cyprus, October 2018.

[2] A. Belenky and N. Ansari, "Tracing multiple attackers with deterministic packet marking (dpm)," in *Proceedings of the IEEE Pacific Rim Conference on Communications Computers and Signal Processing PACRIM 2003 (Cat. No.03CH37490)*, pp. 49–52, Victoria, BC, Canada, August 2003.

[3] A. Saini, C. Ramakrishna, and S. Kumar, "A hybrid optimization algorithm based on ant colony and particle swarm algorithm to address ip traceback problem," in *Cognitive Informatics and Soft Computing*, P. K. Mallick, V. E. Balas, A. K. Bhoi, and A. F. Zobaa, Eds., pp. 429–439, Springer, Singapore, 2019.

[4] A. Mankin, D. Massey, C.-L. Wu, S. F. Wu, and L. Zhang, "On design and evaluation of "intention-driven" icmp traceback," in *Proceedings of the Tenth International Conference on Computer Communications and Networks (Cat. No.01EX495)*, pp. 159–165, Scottsdale, AZ, USA, October 2001.

[5] K. Boudaoud and F. LeBorgne, "Towards an efficient implementation of traceback mechanisms in autonomous systems," in *Proceedings of the NOMS 2008 - 2008 IEEE Network Operations and Management Symposium*, pp. 1015–1018, Salvador, Brazil, April 2008.

[6] N. McKeown, T. Anderson, H. Balakrishnan et al., "Openflow: enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*vol. 38, no. 2, pp. 69–74, mar 2008.

[7] H. Kim and N. Feamster, "Improving network management with software defined networking," *IEEE Communications Magazine*, vol. 51, no. 2, pp. 114–119, 2013.

[8] M. Zhang, G. Li, L. Xu et al., "Control plane reflection attacks and defenses in software-defined networks," *IEEE/ACM Transactions on Networking*, vol. 29, no. 2, pp. 623–636, 2021.

[9] T. Jing, S. Liu, C. Liu, X. Bu, and G. Zhang, "A sdn-based wired/wireless hybrid network architecture of cloud data center," in *Proceedings of the 2021 IEEE 5th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, vol. 5, pp. 232–236, Xi'an, China, October 2021.

[10] J. Li, W. Shi, H. Wu, S. Zhang, and X. Shen, "Cost-aware dynamic sfc mapping and scheduling in sdn/nfv-enabled space–air–ground-integrated networks for internet of vehicles," *IEEE Internet of Things Journal*, vol. 9, no. 8, pp. 5824–5838, 2022.

[11] Q. Wu, M. Zhang, C. Dong et al., "Routing protocol for heterogeneous FANETs with mobility prediction," *China Communications*, vol. 19, no. 1, pp. 186–201, 2022.

[12] A. Mahmood, Q. Z. Sheng, S. Ali Siddiqui et al., "When trust meets the internet of vehicles: Opportunities, challenges, and future prospects," in *Proceedings of the 2021 IEEE 7th International Conference on Collaboration and Internet Computing (CIC)*, pp. 60–67, Atlanta, GA, USA, December 2021.

[13] W. E. Zhang, Q. Z. Sheng, A. Mahmood et al., "The 10 research topics in the internet of things," in *Proceedings of the 2020 IEEE 6th International Conference on Collaboration and Internet Computing (CIC)*, pp. 34–43, Atlanta, GA, USA, December 2020.

[14] A. Mahmood, "Towards software defined heterogeneous vehicular networks for intelligent transportation systems," in *Proceedings of the 2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pp. 441-442, Kyoto, Japan, March 2019.

[15] K. Agarwal, E. Rozner, C. Dixon, and J. Carter, "Sdn traceroute: tracing sdn forwarding without changing network behavior," in *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking, HotSDN '14*, pp. 145–150, Chicago Illinois USA, August 2014.

[16] N. Dayal and S. Srivastava, "Sd-wan flood tracer: tracking the entry points of ddos attack flows in wan," *Computer Networks*, vol. 186, Article ID 107813.

[17] P. Hadem, D. K. Saikia, and S. Moulik, "An sdn-based intrusion detection system using svm with selective logging for ip traceback," *Computer Networks*, vol. 191, Article ID 108015.

[18] P. Ferguson and D. Senie, "Rfc2827: network ingress filtering: defeating denial of service attacks which employ ip source address spoofing," 2000.

[19] H. Burch, "Tracing anonymous packets to their approximate source," in *Proceedings of the 14th USENIX Conference on System Administration, LISA '00*, pp. 319–328, New Orleans, Louisiana, USA, December 2000.

[20] H. Patel and D. C. Jinwala, "Lpm: a lightweight authenticated packet marking approach for ip traceback," *Computer Networks*, vol. 140, pp. 41–50, 2018.

[21] S. Malliga, C. S. Kanimozhiselvi, and S. V. Kogilavani, "Low storage and traceback overhead IP traceback system," *Journal of Information Science and Engineering*, vol. 32, no. 1, pp. 27–45, 2016.

[22] A. C. Snoeren, C. Partridge, S. T. Kent, W. T. Strayer, and S. M. Snoeren, "Single-packet ip traceback 1 single-packet ip traceback," 2008.

[23] R. Stone, "Centertrack: an ip overlay network for tracking dos floods," in *Proceedings of the 9th Conference on USENIX Security Symposium - Volume 9, SSYM'00*, Denver, Colorado, August 2000.

[24] V. Murugesan, M. S. Selvaraj, and M.-H. Yang, "Hpsipt: a high-precision single-packet ip traceback scheme," *Computer Networks*, vol. 143, pp. 275–288, 2018.

[25] N. Handigol, B. Heller, V. Jeyakumar, D. Mazières, and N. McKeown, "I know what your packet did last hop: using packet histories to troubleshoot networks," in *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation, NSDI'14*, pp. 71–85, Seattle, WA, April 2014.

[26] Q. Ren, X. Qiu, P. Chen, and X. D. Liang, "The global flow table based on the software-defined networking," in *Proceedings of the 2015 IEEE International Conference on Communication Problem-Solving (ICCP)*, pp. 264–267, Guilin, China, Ocotber 2015.

[27] H. Zhang, R. Joshua, and R. Jennifer, "Packet traceback for software-defined networks," pp. 1–7, Department of Computer Sciences, Princeton, 2015.
[28] J. Francois and O. Festor, "Anomaly traceback using software defined networking," in *Proceedings of the 2014 IEEE International Workshop on Information Forensics and Security (WIFS)*, pp. 203–208, Atlanta, GA, USA, December 2014.
[29] P. Hadem and D. K. Saikia, "Smite: an sdn and mpls integrated traceback mechanism," Association for Computing Machinery, New York, NY, USA, 2017.
[30] R. Sahay, G. Blanc, Z. Zhang, and H. Debar, "Towards autonomic ddos mitigation using software defined networking," in *Proceedings of the 2015 Network and Distributed System Security (NDSS) Symposium*, San Diego, California, February 2015.
[31] R. Sahay, G. Blanc, Z. Zhang, and H. Debar, "Aroma: an sdn based autonomic ddos mitigation framework," *Computers & Security*, vol. 70, pp. 482–499, 2017.
[32] "Mininet: an instant virtual network on your laptop (or other pc)," 2018, http://mininet.org.
[33] "Scapy v2.1.1-dev documentation," 2010, https://scapy.readthedocs.io/en/latest/introduction.html.
[34] "Hping3(8)-linux man page," 2010, https://linux.die.net/man/8/hping3.
[35] U. S. Caida, "Caida's ark project," 2013, https://catalog.caida.org/software/archipelago.
[36] U. S. Caida, "Caida's skitter project," 2012, http://www.caida.org/tools/skitter.
[37] "Introducing data center fabric, the next-generation facebook data center network," https://code.facebook.com/posts/360346274145943/.
[38] S. Radu, M. Popovici, L. Negreanu, and C. Raiciu, "Symnet: scalable symbolic execution for modern networks," in *Proceedings of the 2016 ACM SIGCOMM Conference, SIGCOMM '16*, pp. 314–327, Association for Computing Machinery, Florianopolis Brazil, August 2016.
[39] A. Broder and M. Mitzenmacher, "Network applications of bloom filters: a survey," *Internet Mathematics*, vol. 1, no. 4, 2002.