Pisces: In-Path Distributed Denial-of-Service Defense via Efficient Authentication Code Embedded in IP Address

Yi Zhao, *Member, IEEE*, Bingyang Liu, Weiyu Jiang, Ke Xu, *Fellow, IEEE*, Qi Li, *Senior Member, IEEE*, Chuang Wang, Zongxin Dou, and Yanjun Liu

***Abstract*—High-volume brute-force distributed denial-of-service (DDoS) attack is among the top threats on the Internet. Existing widely deployed methods (e.g., BGP blackhole and scrubbing center) have difficulty achieving legitimate traffic friendliness, low cost, low latency, and high accuracy. We present an in-path DDoS defense mechanism, namely *Pisces*. Without requiring modifications to existing IP protocols, *Pisces* embeds authentication information into the IP address. Simultaneously, we design a QUIC-based extension to distribute authentication information. *Pisces* incorporates a translator module and a filter module, which accurately identifies malicious and legitimate traffic. These multi-dimensional compatibility advantages make it easy to deploy in the real world. We implement *Pisces* on a high-end commercial router with service processing units. Even without hardware acceleration, a single CPU can achieve 20 Gbps throughput and the performance can scale linearly with the number of CPUs. The additional latency for the victim-related traffic and other traffic is around 27 us and 0.5 us, respectively, whose cost is far less than the scrubbing center. Remarkably, *Pisces* without false positives can provide high-quality datasets for intelligent approaches and form a prominent complementary effect.**

***Index Terms*—Authentication code, DDoS defense, filter, IP address, translator.**

I. Introduction

DISTRIBUTED denial-of-service (DDoS) attack [1], [2], [3], as one of the most representative cyber attacks, achieved 4.8 million attacks in the first half of 2020, up 15 percent from 2019 [4]. Moreover, *Cloudflare*'s DDoS threat report for 2023 Q4 [5] points out that the annualized growth rate of network-layer DDoS attacks is as high as 117%. These DDoS attacks typically force huge amounts of traffic to overwhelm the bottleneck links close to the victim (i.e., service provider). Once attacked, the victim (e.g., financial service, online gaming, and e-commerce platform) will lose the ability to serve externally, causing huge economic losses or financial impact. For example, *Kaspersky Lab* [6] reported that DDoS attack can cost enterprises over $\$2,000,000$, per attack on average. Accordingly, how to effectively defend against DDoS attacks has spawned a huge commercial market. Accordign to *The Business Research Company*'s forecast, for example, the DDoS protection and mitigation market size is expected to reach $8.66 billion in 2028, with a compound annual growth rate of 15.9% [7].

To exhaust the limited resources of the victim, the existing DDoS malicious attack traffic mainly originates from direct flooding DDoS attacks (e.g., SYN Flooding DDoS [8] and UDP Flooding DDoS [9], [10]) and Distributed Reflective Denial-of-Service attacks (DRDoS attacks,[1] a.k.a., amplification attacks) [11]. Regarding flooding attacks,[2] the attacker operates a huge number of botnet hosts to continuously send requests to the targeted victim, thus exhausting the capacity of the victim's network. Considering the same essence but different implementation details, DRDoS can be regarded as a more advanced attack. Via source address spoofing, one or more adversaries send relatively small requests to public hosts (e.g., NTP servers and DNS servers). These requests can exploit the vulnerability of typically UDP-based protocols, e.g., lacking source address verification. Subsequently, public hosts reply much larger responses to the spoofed IP address. In other words, the victim of attack receives the reflected replies.

In terms of mitigating the threat of DDoS attacks, both academia and industry have made efforts to offer promising defense mechanisms. For example, Mishra et al. [12] innovatively exploit variations in entropy between illegal and legitimate traffic to achieve efficient DDoS attack detection. Capability-based approaches [13], [14] are proposed by the academic community.

Received 31 August 2023; revised 3 September 2024; accepted 19 December 2024. Date of publication 15 January 2025; date of current version 11 July 2025. This work was supported in part by the National Science Foundation for Distinguished Young Scholars of China under Grant 62425201 and Grant 61825204, in part by the National Natural Science Foundation of China under Grant 62472036, Grant 62202258, Grant 62132011, and Grant 61932016. *(Corresponding authors: Ke Xu; Bingyang Liu.)*

Yi Zhao is with the School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing 100081, China (e-mail: zhaoyi@bit.edu.cn).

Bingyang Liu and Weiyu Jiang are with the Huawei, Beijing 100095, China (e-mail: liubingyang@huawei.com; jiangweiyu1@huawei.com).

Ke Xu is with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China, and also with Zhongguancun Laboratory, Beijing 100094, China (e-mail: xuke@tsinghua.edu.cn).

Qi Li is with the Institute for Network Sciences and Cyberspace, Tsinghua University, Beijing 100084, China (e-mail: qli01@tsinghua.edu.cn).

Chuang Wang, Zongxin Dou, and Yanjun Liu are with the Huawei, Beijing 100095, China (e-mail: wangchuang@huawei.com; douzongxin@huawei.com; jason.liuyanjun@huawei.com).

Digital Object Identifier 10.1109/TDSC.2025.3530140

[1] In this paper, we will utilize the terms DRDoS attacks and amplification attacks interchangeably.

[2] Note that in addition to direct flooding DDoS attacks, amplification attacks can also cause similar phenomenon. Without extra explanation, in this paper, flooding attacks refer to direct flooding DDoS attacks.

However, before establishing the capability-protected channel, such methods require a capability-setup channel. This premise can also be attacked by DDoS attackers. In terms of intelligent detection, Mirnajafizadeh et al. [15] design a distributed and in-network data collection to enhance network attack detection. However, academic proposals (e.g., neural network-based intelligent defense [16]) require high-cost hardware or software updates and upgrades, it is difficult to enforce the deployment of these proposals in the Internet [17]. Similarly, Tran et al. [18] also point out that existing rerouting approaches cannot solve DDoS problem. Considering that individual victims cannot mitigate DDoS attacks, Ramanathan et al. [19] innovatively propose the security service for collaborative mitigation, enabling victims to request upstream ISPs (including both immediate and remote ISPs) to monitor and filter DDoS attacks. Moreover, these ISPs can authenticate the request without having prior trust with the victim.

In fact, the defense approaches actually deployed on the Internet are mainly *BGP blackhole* [20] and *traffic scrubbing* [21]. Regarding *BGP blackhole*, it discards malicious traffic and legitimate traffic indiscriminately, and the victim still cannot provide normal services. Different from *BGP blackhole*, *traffic scrubbing* requires relevant ISPs to redirect traffic to third-party security service providers (e.g., *Akamai* and *Cloudflare*). They utilize professional equipment and proprietary techniques to strip malicious traffic from legitimate traffic, and then re-inject the remaining traffic back to the relevant ISPs. Although *traffic scrubbing* can ensure that legitimate traffic is served, it introduces high currency expenses [22], [23] and obvious processing latency [24]. In summary, the existing DDoS defense approaches cannot meet the requirements of low cost, high accuracy, and low latency.

To solve the above challenges, we design and implement an in-path DDoS defense mechanism, namely *Pisces*,[3] which enables the router to defend against malicious attack traffic in real time, without affecting the normal forwarding of legitimate traffic from historical clients.[4] Meanwhile, in terms of deploying our proposed defense mechanisms in practice, *Pisces* does not require the cooperation of other participants (e.g., additional ASes, public reflectors, and third-party security service providers), nor parsing of application-layer data. Correspondingly, the characteristics of independent deployment, low latency and low cost further facilitate the widespread deployment of *Pisces*.

More specifically, our proposed *Pisces* is mainly composed of two modules, i.e., *translator* module and *filter* module. Regarding the *translator* module, it only processes the header of IP packets. Through embedding authentication code into IP addresses, *translator* can efficiently perform address transform and restoration. Subsequently, these embedded authentication code will serve as an indispensable input to the *filter*. In cooperation with the pre-configured encryption module, *translator* and *filter* deployed on the same or different routers can synergistically distinguish malicious traffic from legitimate traffic, while accurately blocking malicious traffic.

Our contributions can be summarized as follows:

- We propose a novel in-path DDoS defense mechanism based on embedding authentication information in IP addresses. Routers can efficiently process the authentication information without flow states. The mechanism is inherently free of false positive, and has features like low latency, low cost, and easy for incremental deployment.
- We implement the newly proposed *Pisces* on a high-end and high-throughput commercial router, which is typically deployed in IP backbone, WAN, MAN, etc., where anti-DDoS approaches are urgently required.
- Extensive experimental results demonstrate that the processing latency is merely around 27 us. Meanwhile, *Pisces* protects target services' FCT unchanged as the attack volume increases.

The organization of the rest of this paper is as follows. Section II discusses related work from three perspectives: industral approaches, academic proposals, and emerging methods. Section III presents the system design, including the system overview, the design details of two key modules (i.e., *translator* module and *filter* module), as well as QUIC-based authentication code distribution. Regarding the evaluation, Section IV first clarifies the hardware implementation details, and then gives the experimental configuration and performance analysis. Subsequently, Section V discusses several potential scenarios. Finally, Section VI concludes the paper.

II. Related Work

Due to the low-cost and convenient implementation, DDoS attacks frequently occur, causing serious security problems and huge economic losses. Both the industry and academia are focusing on DDoS attacks in the network security field, and various defense mechanisms have been proposed. In this section, we discuss existing defense mechanisms from industrial, academic, and emerging perspectives.

Industrial approaches: Industry prefers practical and economical approaches. The most common commercial anti-DDoS solutions include *BGP blackhole* [20], [25], [26] and *traffic scrubbing* [21]. Using *BGP blackhole* [25], [26], the network can announce victim IP addresses or prefixes in BGP, so all the traffic to the target is dropped. The problem is that legitimate traffic is also dropped, which is unacceptable in a lot of scenarios. On the contrary, *traffic scrubbing* [21], [27] can distinguish legitimate and malicious traffic. However, in addition to requiring expensive equipment, when *traffic scrubbing* redirects traffic to scrubbing devices, it will inevitably increase additional transmission and processing delay, and may cause privacy leaks [24], [28]. This is also unacceptable for some important services. In summary, these methods widely deployed in the industrial field still expose significant defects in terms of

[3]Similar to the twelfth astrological sign in the Zodiac, i.e., Pisces (a.k.a.,)-(), the defense mechanism we propose requires two modules (i.e., *filter* and *translator*) to cooperate with each other to achieve defense. Therefore, in our project, the referred defense mechanism is named *Pisces*.

[4]The term 'historical clients' refers to clients who have used the protected service before the defense mechanism (i.e., *Pisces*) is activated. Correspondingly, historical clients and new clients are a pair of terms with opposite concepts. Therefore, our *Pisces* will have some negative impact on new clients, which is our potential shortcoming. Section V will further discuss that this negative impact is acceptable, and give some possible compensation plans.

legitimate traffic friendliness, economic cost, detection efficiency, and privacy.

Academic proposals: Regarding DDoS defense, a mainstream method in academia is based on filtering, such as IP spoofing prevention [29], [30], [31], packet marking [32], [33], hop-count [34], and push-back [35]. These rule-based methods can effectively filter known attack traffic, but cannot cope with complex unknown or encrypted attack traffic. To break through the limitations of fixed filtering granularity, Li et al. [36] design an adaptive DDoS-filtering rule generation method, which is compatible with both effective filtering granularity and victim-first properties to achieve optimal filtering effect. Since the DDoS traffic characteristics of different IoT devices are different, Cvitić et al. [37] innovatively propose a boosting method of logistic model trees, which can configure different detection models for different devices. To accurately classify DDoS attacks, Mishra et al. [38] design a majority vote-based ensemble of classifiers, which involves multiple methods, e.g., random forest, decision tree, and naive bayes.

In addition, overlay-based approaches (e.g., SOS [39] and Phalanx [40]) can defend DDoS attacks via large scale overlay networks. Andersen et al. [41] provides accountability for the network architecture to improve security. However, the change from the prefix address to the flat address is not compatible with existing network operation. Instead of identifying features of malicious traffic, Portcullis [42] mitigates DDoS attacks on initial connections by requiring clients to solve mathematical problems. This method can consume more resources for attackers and increase the difficulty to launch an attack, but it may also bring unacceptable latency for establishing legal connections. Many existing solutions are popular when they can be directly deployed at the ingress or egress of a victim network. However, when the network is congested, the professional hardware devices deployed on the victim network may also have no enough resources to effectively identify and filter malicious traffic.

Emerging methods: Compared with proprietary hardware and virtual appliances, defending against DDoS attacks through programmable switches shows outstanding advantages in cost, performance, and flexibility [43], [44], [45]. For example, Zhang et al. [43] design the modular abstraction for defense policies, the resource orchestration mechanism for policy primitives mapping, and the runtime management mechanism for dynamic attacks. By integrating these innovations with programmable switches, victims can flexibly defend against a variety of DDoS attacks and their variants, without requiring any hardware updates. Also independent of additional hardware data plane support, Liu et al. [44] propose switch-native line-speed DDoS attack detection and mitigation method.

With the explosive development of artificial intelligence (AI) technologies [16], [46], [47], related DDoS defense solutions are also emerging. In terms of empowering individual devices with intelligent capabilities, decision trees are deployed on the data plane through transformation [48], [49], thereby achieving line-speed traffic classification. Yan et al. [50] propose deploying neural networks on the data plane to enhance intelligent capabilities. Moreover, to coordinate the intelligence capabilities of multiple individuals, Zhao et al. [51] proposed a collaboration-enabled intelligent Internet architecture that can fully utilize the data and resources possessed by different devices to detect malicious traffic. Fu et al. [52] exploit packet length patterns to improve detection efficiency. Feng et al. [53] propose a reinforcement-learning-based approach to achieve application-layer DDoS attack defense. However, AI-based defense mechanisms are limited by training data and the lack of high-performance computing devices, which restricts such methods from being directly deployed on routers and switches.

In this paper, therefore, we propose an in-path DDoS defense mechanism (i.e., *Pisces*). The router can accurately and efficiently identify and filter malicious traffic inside the router, so the cost, latency and privacy issues can be avoided. In addition, *Pisces* can be categorized into the capability-based approaches [13], [14]. Capability-based approaches require that capability is authorized before data can be sent. The uniqueness of *Pisces* is that by embedding authorization implicitly in IP addresses, endpoints and irrelevant routers en-route can be agnostic, which makes the newly proposed approach easier to deploy. Moreover, traffic data classified by *Pisces* can enrich the training data required by existing intelligent methods.

III. System Design

To enable the router to effectively defend against various DDoS attacks without interfering with normal services, we propose the in-path DDoS defense mechanism, i.e., *Pisces*. In this section, we first systematically provide the basic framework of the entire defense mechanism. Subsequently, we introduce in detail two key modules (i.e., *translator* module and *filter* module) to support the efficient real-time running of the entire global system. Finally, we propose QUIC-based authentication code distribution mechanism to further expand defensible scenarios.

A. System Overview

Regarding the DDoS attack and defense scenario involved in this paper, there are mainly three types of participants, namely the malicious attacker, the legitimate user, and the protected victim. As for these three types of participants, their corresponding individual instances are separately represented as a_i, u_j, and v_k, where i, j, and k are the indexes of a specific individual, respectively. In addition, in amplification attacks, there will be some public hosts (e.g., Memcached servers, Network Time Protocol (NTP) servers, and Domain Name System (DNS) servers) acting as reflectors, and we utilize r_l to represent the reflector whose index is l. For example, Xu et al. [54] deftly coordinate a large number of vulnerable DNS resolvers that have small amplification capabilities into a powerful DoS amplifier, thereby achieving a more powerful DRDoS attack.

For the sake of clarity, Table I lists the major notions in our defense system. It is worth noting that the specific meaning of the symbol also depends on its superscript and subscript. For example, $AuCode_{PSK}$ and $AuCode_{PrA}$ respectively refer to the authentication codes distributed through two different approaches, and a more detailed explanation will be introduced in Section III-D.

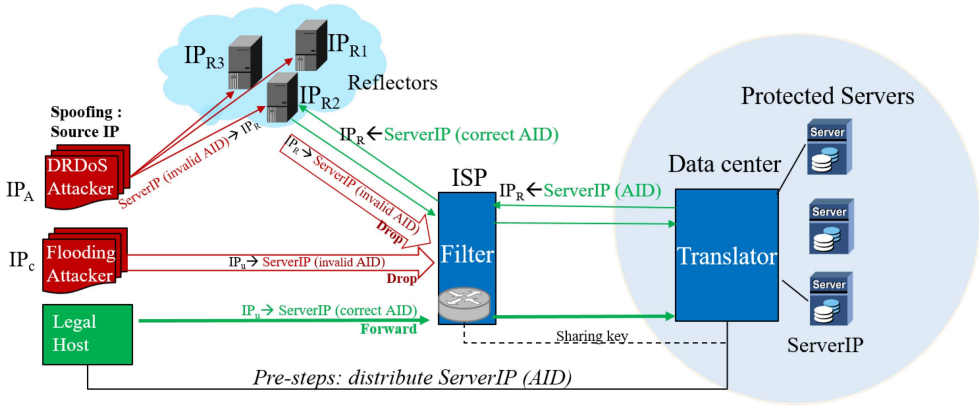


Fig. 1. Basic framework of our defense system (i.e., *Pisces*) and the workflow of various traffic.

TABLE I
MAJOR NOTATION LIST

Symbol	Description
a_i	A malicious attacker with index i
u_j	A legitimate user with index j
v_k	A protected victim with index k
r_l	A publicly available reflector with index l
$\mathcal{T}$	Address transform function
$\mathcal{R}$	Address restoration function
$\mathcal{E}$	Encryption function
$\mathcal{D}$	Decryption function

As formerly notified, DRDoS and flooding are the two most mainstream types of DDoS attacks, and they also account for the vast majority of DDoS attack traffic. To effectively defend against these two types of attacks, while having low latency, low cost, and individually deployable features, we propose and implement *Pisces*, of which the basic framework and the workflow of various traffic are illustrated in Fig. 1.

Whether it is DRDoS or flooding, the essence of implementing DDoS attacks is to force huge amounts of traffic to exhaust the limited resources owned by a particular service (i.e., victim). Unlike regular expression denial-of-service (ReDoS) attacks (which use complex algorithms to trigger super-line worst-case matching times) [55], [56], DDoS attacks have obvious distributed characteristics and are difficult for the victim to mitigate independently. However, existing routers cannot utilize the network-layer information to efficiently filter malicious traffic on the forwarding plane. The root cause is that IP address lacks authentic verifiable characteristics. Motivated by this root cause, our core idea is to embed authentication code into the IP address without requiring changes to the existing protocol stack. With authentication code embedded in the IP address, routers equipped with *Pisces* on the path can accurately distinguish malicious traffic from legitimate traffic, ensuring the direct forwarding of legitimate traffic without requiring expensive third-party scrubbing services. Note that different from the existing address verification technologies, we do not require cooperation between ASes.

There are two main reasons for abandoning the use of a header extension. First, different network devices have different compatibility for header extensions. Once a network device cannot recognize the extension header, the corresponding information (i.e., authentication code) is discarded. Instead, we choose an IP address that is compatible with any network device. Second, for DRDoS attacks, if the authentication code is stored in the extension header, the returned message will lose the verifiable information. Instead, our *Pisces* put the authentication code in the source address. In the returned message, the source address will become the destination address, and verifiable information will continue to be maintained.

Considering that IPv6 address has sufficient address bits, we implement the newly proposed *Pisces* in the IPv6 protocol. As illustrated in Fig. 1, the *translator* exploits the redundant space in the IP address, embedding authentication code into the IP address. Subsequently, the *filter* decrypts relevant authentication code to prevent malicious traffic. More specifically, Fig. 1 involves the following four scenarios.

1) Legitimate Packets for Public Services During DRDoS Attacks: Once attacked, the victim will request the nearest neighbor upstream ISP service provider to activate the defense mechanism. For public service (e.g., DNS) request packet originating from the victim, i.e., $IP_v \rightarrow IP_r$, once it reaches the router of the nearest neighboring upstream ISP, *translator* will perform TRANSFORM operation, embedding authentication code into the source address, denoted by (1).

$$IP_{v_i^t} = \mathcal{T}\left(IP_{v_i^0}, timestamp'; key, rule\right) \tag{1}$$

where $t >= 0$ and $timestamp' = timestamp/10$. key is the key required by the submodule (e.g., using HMAC-SHA1 to generate Code in Fig. 2) when performing TRANSFORM operation. $relu$ refers to the optional configuration of TRANSFORM operation. For example, in the experiment, our $rule$ is refined to use HMAC-SHA1 to generate $CODE$, and fill the first 32 bits of HMAC-SHA1 output into the $CODE$ part in Fig. 2. Note that the part on the left of the semicolon is the variable during (1) execution, which can change at any time. The part on the right of the semicolon is (1) configuration, which remains unchanged during the function execution. And the comma is used to distinguish different elements of the same type.

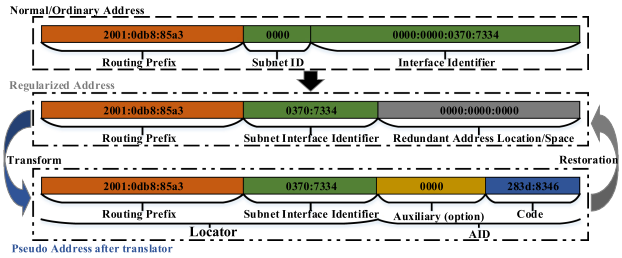


Fig. 2. An example of IP address configuration, transform, and restoration.

In other words, the source-destination address pair of the packet is updated to $IP_{v_i^t} \rightarrow IP_r$. The corresponding reflector does not require any additional modification, and directly responds to the request, i.e., $IP_r \rightarrow IP_{v_i^t}$. Once it reaches the victim's upstream ISP,[5] the filter will routinely forward the reply packet after the decryption verification is passed. Before leaving the upstream ISP, *translator* performs RESTORATION operation to adjust the destination IP address, denoted by (2).

$$IP_v = \mathcal{R}\left(IP_{v_i^t}, timestamp'; rule\right) \tag{2}$$

Thus the reply packet with $IP_r \rightarrow IP_v$, can accurately reach the corresponding public service requester.

2) Malicious Packets for Public Services During DRDoS Attacks: To perform the amplification attack, the attacker spoofs its real IP address (i.e., IP_a) to the victim's address (i.e., static address IP_v or some leaked dynamic address $IP_{v_i^{t'}}$). Once the public server serving as the reflector receives the spoofed request, i.e., $IP_a(IP_v||IP_{v_i^{t'}}) \rightarrow IP_r$, it immediately sends an enlarged reply directly to the victim with IP_v. When the reply arrives at the upstream ISP of the victim, *filter* will directly refuse the forwarding due to the lack of correct authentication code in the spoofed destination IP address (i.e., IP_v or $IP_{v_i^{t'}}$). This allows the victim's bottleneck link to be able to maintain normal service with historical clients.

3) Legitimate Packets for Individual Clients During Flooding Attacks: Similar to the legitimate reply sent to the victim during DRDoS attacks, when an individual legitimate client[6] sends a packet to the victim, it also embeds authentication code into the destination IP address. Note that the client's protocol stack still does not require any changes, but *Pisces* requires the client to maintain a key at the application layer. More implementation details about key distribution and synchronization will be introduced in Section III-D. Based on the key and encryption mechanism, the application layer transforms the victim's static IP address (i.e., IP_v) into an address embedded with authentication code (i.e., $IP_{v_i^t}$). Therefore, the packet from IP_c to $IP_{v_i^t}$ at the network layer can be decrypted and authenticated by *filter*, and then updated to the static IP address (i.e., IP_v) by *translator*. In the end, the legitimate packet reaches the desired service host.

4) Malicious Packets for Individual Clients During Flooding Attacks: Since flooding attacker cannot obtain key and encryption mechanism, a malicious client can only send requests to the static IP address (i.e., IP_v) of the victim.

Due to the lack of correct authentication code in the static address, our proposed *Pisces* will filter out the packets (i.e., $IP_a \rightarrow IP_v$[7]) as soon as they reach the victim's upstream ISP. In other words, malicious traffic generated by the flooding attacker cannot reach the victim's bottleneck link to launch an effective attack.

Overall, the basic framework of our *Pisces* to defend against DDoS attacks can be illustrated in Algorithm 1. More internal details and principles about *translator* and *filter* will be introduced in Sections III-B and III-C, respectively.

Algorithm 1: The Basic Framework to Defend Against Various DDoS Attacks.

Input: Network-layer packet header, list of selected ports $\mathcal{L}$, and protected address prefix $Protected_Prefix$
if $IP_{destination} \in Protected_Prefix$ **then**
  Execute the *FILTERING* function of *filter* module, i.e., Algorithm 2
  Execute the *RESTORATION* function of *translator*, i.e., (2)
else
  if $(IP_{source} \in Protected_Prefix)\&\&PORT \in \mathcal{L}$ **then**
    Execute the *TRANSFORM* function of *translator*, i.e., (1)
  else
    Routinely forward received packets
  end if
end if

B. Translator Module

To maximize compatibility with the existing network architecture and various protocol stacks, the *translator* of *Pisces* only processes the source IP address or the destination IP address. Specifically, the *translator* has two functions, i.e., $TRANSFORM$ function and $RESTORATION$ function.

Fig. 2 takes an IPv6 address as an example to illustrate the configuration, transform, and restoration in detail. As the Internet routable identifier of the host, the IP address has the ability to locate a host. For an ordinary address (e.g., the first row in Fig. 2), it consists of *Routing Prefix*, *Subnet ID*, and *Interface Identifier*. In terms of network configuration, the victim can configure *Subnet ID* and *Interface Identifier* independently.

[5]For the sake of concise description, unless otherwise expressly specified, the upstream ISP in this paper refers to the victim's nearest neighbor upstream ISP.

[6]A legitimate client refers to a client who has registered for the service provided by the victim. For an attacker who initiates a flooding attack, it is difficult to control a large number of legitimate clients. Therefore, in this article, we assume that there are no legitimate clients in the hosts controlled by the attacker.

[7]In addition to eavesdropping on static destination IP address (i.e., IP_v), attackers may also eavesdrop on dynamic destination IP address (e.g., $IP_{v_i^t}$) with authentication information utilized by legitimate clients. The dynamic destination IP address is associated with time and changes dynamically. With purely eavesdropping on some instances on partial moments, the attacker still cannot achieve a sustained and effective attack.

TABLE II
MAPPING

Flow Index	Protocol Number	Port Number	Service Description
1	0x11	11211	Memcached
2	0x06	11211	Memcached
3	0x11	53	DNS
4	0x11	123	NTP
5	0x11	19	CHARGEN
6	0x11	161	SNMP-1
7	0x11	199	SNMP-2
8	0x11	17	QOTO
9	0x11	137	NetBIOS
10	0x11	1434	MSSQL

Moreover, the number of hosts that these two parts can express far exceeds the number of hosts owned by the victim. To deploy our defense mechanism, we require the victim to compress *Subnet ID* and *Interface Identifier* into *Subnet Interface Identifier*, and the remaining address bits are configured as static parameters according to a *Static Policy*. Note that these static parameters are pre-configured value without actual semantics. With this configuration, *Routing Prefix* and *Subnet Interface Identifier* in the regularized address (e.g., the second row in Fig. 2) can be utilized to identify a specific host in the global network.

In our work, we regard *Routing Prefix* and *Subnet Interface Identifier* as a whole, denoted by *Locator*. In addition to the address bits of the locator, the remaining address bits can be utilized to store authentication code. Therefore, the ISP reserves an entire unique /48 (i.e., the AID in Fig. 2) for every service that could potentially request protection. Correspondingly, the first 80 bits of the IPv6 address can distinguish different services. Note that the $TRANSFORM$ function will only be activated for outgoing traffic to the reflectors. To identify reflector-related traffic, the *translator* is required to maintain a special protocol-port mapping table. For example, one item in the table corresponding to DNS traffic specifies that $protocol\_number = 0x11$ (i.e., UDP protocol) and port number $port\_number = 53$. Table II provides a configuration example of the special mapping table, which the administrator can configure according to specific scenarios. Note that the referred mapping table is small enough. It does not take up too much storage, nor does it bring obvious table lookup overhead.

After determining that the packet belongs to reflector-related outgoing traffic, $TRANSFORM$ utilizes (3) to cryptographically generate verification information.

$$CODE = \mathcal{E}_T\left(IP_r, Locator, DF; key\right) \tag{3}$$

where $\mathcal{E}_T$ represents an encrypted fingerprint generation algorithm (e.g., HMAC-SHA1, HMAC-SHA256, HMAC-SHA512, CBC-MAC, SM3, etc.) and key refers to the relevant secret cryptographic key. Moreover, key can be configured using a simple master key (e.g., MK) and a key derivation function (e.g., $HKDF$ [57]), i.e., $key = HKDF(MK)$. IP_r is the IP address of the targeted reflector. $Locator$ can be extracted from the source IP, and DF is a dynamic factor like a timestamp or other variables. Immediately afterwards, the static parameters in the source address are replaced by the $CODE$. In other words, $IP_{v_i^t}$ in (1) is the combination of $Locator$ and $CODE$.

For incoming traffic to the protected victim, the $RESTORATION$ function will restore the non-*Locator* part of the destination IP address to static parameters in accordance with the pre-configured *Static Policy* (e.g., all zero elements). Since *Locator* is sufficient for routing to a specific host, *Static Policy* is the most efficient approach to establish a unique connection between transformed IP address and the real static IP address.

C. Filter Module

Similar to the *translator*, our *filter* is also highly compatible with the existing network architecture and protocol stacks. In Section III-B, we have introduced that the *translator* will equip the IP address with the correct authentication code for legitimate traffic. The referred authentication code will be extracted by the *filter*. In fact, the *filter* is the front-end tool to defend against attack traffic.

Regardless of whether it is the victim's incoming traffic related to reflectors, once the victim requests the upstream ISP to activate *Pisces*, all packets will be checked by the *filter*. Only packet that carries correct authentication code can be forwarded by the *filter*. Otherwise, the packet will be discarded directly. First, for the legitimate reflector-related packet, the *translator* embeds authentication code into the IP address (i.e., (1) and (3)). On the contrary, there is a lack of correct authentication code in malicious reflector-related packet. For incoming traffic to the victim, we utilize (4) to decrypt the encrypted data in the destination IP address.

$$isRight = \mathcal{D}_F\left(IP_{v_i^t}, DF; key, rule\right) \tag{4}$$

where $\mathcal{D}_F$ represents the decryption algorithm corresponding to (3).

For traffic sent directly from the client to the victim (i.e., non-reflector related traffic), we propose the insensible authentication code distribution mechanism. It can ensure that legitimate packets directly from the client can pass the filter inspection. As it involves the cooperation of different modules, the detailed distribution mechanism, as well as the corresponding encryption and decryption algorithms, will be introduced in Section III-D.

As described in Algorithm 1, the *filter* only performs *FILTERING* function on traffic sent to the victim. Therefore, the $FILTERING$ function of the *filter* can be illustrated by Algorithm 2.

Compared with flooding attacks, the protocol interaction logic under DRDoS attacks is easier to understand. Therefore, we have Fig. 3 to illustrated the protocol diagram under the flooding attack scenario. Note that our $FILTERING$ function is executed packet by packet. Although in the implementation of Algorithm 2, different internal verification mechanisms are executed for reflected and non-reflected traffic, our $FILTERING$ function can defend against a specific type of attack (e.g., DRDoS attack or flooding attack), and it can also handle scenarios where two types of attacks coexist.

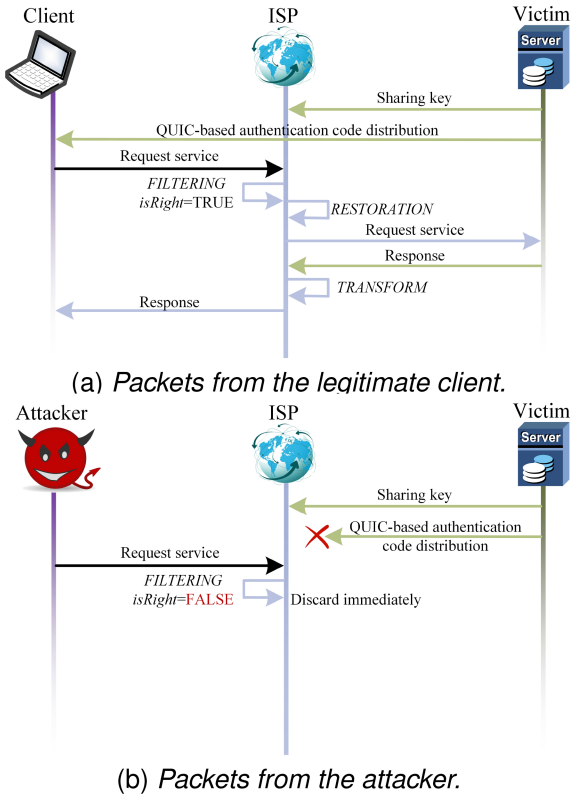


Fig. 3. Protocol diagram for packets during flooding attacks.

Algorithm 2: *FILTERING* Function.

if $PORT \in \mathcal{L}$ **then**
  Set the value of $toForward$ by (4)
else
  Set the value of $toForward$ by Algorithm 3
end if
if $toForward == 1$ **then**
  Routinely forward the received packet
else
  Discard immediately
end if

D. QUIC-Based Authentication Code Distribution

Regardless of the type of DDoS attack scenario, legitimate traffic often coexists with malicious traffic. To forward legitimate traffic normally while discarding malicious traffic in parallel, the *filter* utilizes the authentication code embedded in the header of each packet to distinguish whether it is legitimate. For reflector-related traffic, the *translator* embeds authentication code in the source IP address of each packet sent by the victim. Correspondingly, the destination IP address of each packet replies to the victim also contains authentication code. However, for non-reflective traffic (i.e., packets sent directly from the client to the victim), we cannot require *translator* to embed authentication code for legitimate traffic.

To enable the *filter* to distinguish whether the packet directly from the client is legitimate, we propose and implement the QUIC-based authentication code distribution mechanism. Note that this mechanism is non-inductive. Through the existing secure transport layer protocol (i.e., QUIC), it can distribute authentication code to legitimate clients, without requiring user-perceived operations and any modification of client implementation details (e.g., program module and protocol stack). In addition, this distribution mechanism can ensure that the header of each packet generated by legitimate clients contains authentication code, which has the characteristics of efficient extraction, stability, and durability. Since the authentication code is embedded in the header of each packet, our *Pisces* still works normally even if the application-layer data is encrypted. Therefore, our proposed *Pisces* has the advantage of inherent privacy preservation.

Regarding the implementation of our proposed authentication code distribution mechanism, we have the following two assumptions. First, all the victims' servers are compatible with the QUIC protocol. Second, all legitimate users employ QUIC-enabled client (e.g., Chrome browser and Chromium) to enjoy the protected services provided by the victim (e.g., Chrome server). According to different scenarios and demands, the authentication code distribution can be divided into the following two aspects.

1) Initial Authentication via Pre-Shared Key: To speed up the establishment of a secure connection, both QUIC protocol [58], [59] and TLS 1.3 protocol [60] support pre-shared key (PSK) mode, which can be used for subsequent handshakes to avoid public key operations. In other words, pre-shared key mode is the basis for 0-RTT, because the pre-shared key issued by a previous connection can be reused when a new connection is to be established. This can be implemented through the cooperation of *NewSessionTicket message* and *Client Hello* message. First, after authentication of the previous connection is finished, the server sends a *NewSessionTicket* message containing a ticket to the client. When a next connection is to be established, the client extracts the *key identifier* from the *ticket*, and fills it to the PSK extension of the first *Client Hello* message. Then, the client can utilize the key indicated by PSK to encrypt the early application-layer data, and carry the encrypted application-layer data in the first *Client Hello* message.

To enable legitimate traffic origin from a registered or ordinary client to pass the *filter* inspection, our proposed *Pisces* distributes authenticatable identification (abbreviated as AID) through the cooperation of *NewSessionTicket* message and *Client Hello* message. The referred AID is an authenticatable identification with new semantics defined by *Pisces*. It can contain client identifier (CID), cryptographic key identifier (KID), trust level (TRL), expire time (EXT), authentication code ($AuCode_{PSK}$), etc.. Specifically, KID, TRL and EXT jointly fill the $Auxiliary$ domains in Fig. 2, while $AuCode_{PSK}$ fill the $Code$ domains alone. KID points to a key K_{PSK}, which is generated by the victim according to its master key MK and KID. Taking the fields other than KID and $AuCode_{PSK}$ in the AID as input, we can get the value of $AuCode_{PSK}$ via a cryptographic hash function and a secret cryptographic key, denoted by (5).

$$AuCode_{PSK} = \mathcal{E}_{PSK}\left(CID, TRL, EXT; K_{PSK}\right) \quad (5)$$

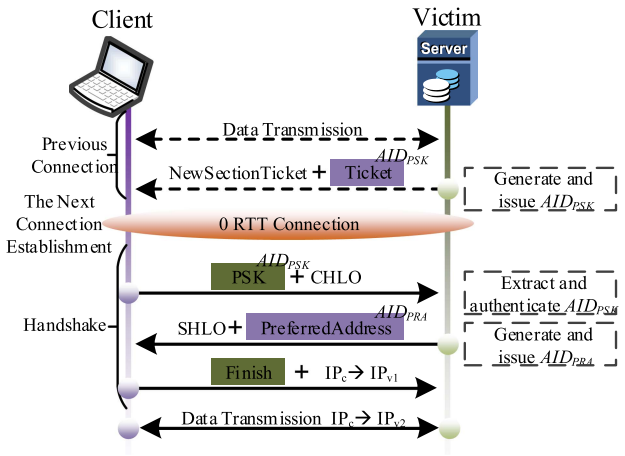


Fig. 4. QUIC-based distribution.

where $\mathcal{E}_{PSK}$ refers to the cryptographic hash function and the specific implementation details will be discussed in Section IV-B.

As illustrated in Fig. 4, the victim applies (5) to generate an AID associated with a specific customer, and then issues the AID to the client through the *ticket* field in the *NewSessionTicket* message. When a new connection is to be established, *Client Hello* message sent by the client contains the previously obtained AID. When the packet reaches the upstream ISP of the victim, the *filter* will use CID, TRK, and EXT in the AID as the input of (5), and obtain the corresponding key K according to KID. Only if the calculated result is consistent with the $AuCode$ carried in the AID, the packet is recognized as legitimate traffic and then forwarded. After authentication in this connection is finished, the victim can update the value of TRL based on the number of connection, and then issue an updated AID to the client. As a result, a legitimate client who always accesses the victim may be issued a more trustable AID. To avoid modifying the server, a middle network device in front of the victim can take over the work of issuing AID and verifying AID.

In summary, the new semantic AID embedded in the PSK extension of the first *Client Hello* message ensures that the legitimate client can complete the initial authentication when our proposed *Pisces* is activated.

2) Dynamic Migration via Preferred Address: Through the cooperation of *NewSessionTicket* message and *Client Hello* message, we can ensure that the legitimate client can establish a new connection with the victim. However, the *Client Hello* message only exists in the handshake phase. Once the handshake is completed, we cannot continue to identify legitimate packets through the AID embedded the PSK extension of the *Client Hello* message. Therefore, we implement the dynamic migration via *preferred address*, ensuring that the established connection can continuously transmit data with the victim.

Regarding the QUIC protocol, its late binding mechanism can support connection migration. As illustrated in Fig. 4, through the victim's public IP address (e.g., the address obtained through DNS), the client sends the *Client Hello* message to request the victim's service. Then, the *Preferred Address* field in the replied

Algorithm 3: Stateless Verification.

```
if InitinalConnection == 1 then
  if EXT is valid then
    return Code == AuCode_PSK
  else
    return 0
  end if
else
  return Code == AuCode_PrA
end if
```

Server Hello (SHLO) message will indicate a new IP address, requiring the client to migrate packets to the new destination IP address. After the handshake is completed, the client applies the value in *Preferred Address* to configure the subsequent destination address. Therefore, when sending application-layer data, the destination IP address of the packet is not the victim's public IP address, but the dynamic IP address extracted from the the *Preferred Address* field in the replied Server Hello (SHLO) message.

Similar to reflection-related packets, the IP address specified by *Preferred Address* is embedded with authenticatable identification (i.e., AID_{PrA}). In other words, *Preferred Address* is also composed of $Locator$ and AID_{PrA} together, illustrated in Fig. 2. This ensures that the packets from legitimate clients can pass the *filter* inspection. The referred AID_{PrA} consists of CID and $AuCode_{PrA}$. Specifically in Fig. 2, CID and $AuCode_{PrA}$ serve as $Auxiliary$ domains and $Code$ domains, respectively. We can obtain $AuCode_{PrA}$ through (6).

$$AuCode_{PrA} = \mathcal{E}_{PrA}\left(Locator, CID, DF; K_{PrA}\right) \tag{6}$$

where $\mathcal{E}_{PrA}$ can be the same algorithm as $\mathcal{E}_{PSK}$ or other cryptographic hash algorithms. K_{PrA} is the symmetric key associated with CID, which can be obtained through a secure key derivation algorithm with the cooperation of MK.

Different from (5), (6) applies DF instead of EXT to maintain timeliness. After one client ends the service provided by the victim, it may take a relatively long time to enjoy the service again. Therefore, the EXT indicates a relatively long effective time period. On the contrary, in the service process, using frequently changed DF can increase the flexibility and confidentiality of $AID_preferred$. It may take a relatively long time for the client to enjoy the service again after a service.

Note that whether it is through PSK to complete the initial authentication or through $PreferredAddress$ to achieve dynamic migration, the corresponding AID generation and distribution can be implemented by a middle network device in the protected network. This can avoid modifying the victim's server program.

Since our AID_{PrA} is embedded in the IP address, we can ensure that the network-layer header of each packet sent by the client to the victim contains the AID_{PrA}. Therefore, regardless of whether the connection is established for the first time, we can implement application-layer stateless verification through the QUIC protocol, illustrated in Algorithm 3.

In addition to embedding AID_{PrA} into the IP address, $PreferredAddress$ can also serve load balancing. In fact, when embedding the AID_{PrA} into the $PreferredAddress$, it is not required to modify the $Locator$ field of the IP address. For example, $IP_1 = 2001:0db8:85ac:0000:0000:0000$. When the server responds to the client, $IP_{PrA} = 2001:0db8:85ac:0000:283d:8346$. IP_1 and IP_{PrA} have the same $Locator = 2001:0db8:85ac$, so the client is still accessing the same server. To facilitate load balancing in parallel, we can also modify the $Locator$ before embedding the AID_{PrA}. For example, we set $PreferredAddress$ to $IP_{PrA} = 2001:0db8:85a1:0000:56dc:9872$. Correspondingly, the client will subsequently access the server with $Locator = 2001:0db8:85a1$ instead of the server with $Locator = 2001:0db8:85ac$. Note that the $Locator$ must be modified before embedding AID_{PrA}. Because the calculation of AID_{PrA} depends on the relevant $Locator$.

IV. Evaluation

In this section, we present the implementation of *Pisces* and the evaluation results.

A. Hardware Implementation

We implement a prototype of *Pisces* on a commercial high-end type of universal service router. This type of router is applicable in large-scale and high-volume network scenarios including IP backbone, IP bearer network, WAN, MAN, DCI, etc., where DDoS defense is most needed. This router can support up to eight boards or units, including Line Processing Unit (LPU) and Service Processing Unit (SPU), as illustrated in Fig. 5(a). LPU is used for basic IP forwarding at line rate. SPU supports multiple ARM CPUs and is mainly used for more complex services, such as Network Address Translation (NAT) and IPSec.

We implement *translator* or *filter* on the flexible and programmable SPU. In other words, routers in the real world capable of performing NAT or IPSc is eligible for deployment of our approach. The total processing procedure for a packet is illustrated in Fig. 5(b). Upon the arrival of an incoming packet, the inbound LPU decides whether the packet needs to be processed by *Pisces*, i.e., whether source or destination address belongs with the victim network and whether the protocol and ports are under protection. If so, the packet is redirected to the SPU for *translator* or *filter* processing, and then directed to the outbound LPU and forwarded out. Otherwise the packet directly sent to the outbound LPU.

1) Inbound LPU: Identifying target traffic. In practice, *Pisces* is deployed at ISP router at the entry of the victim network. So outbound traffic is subjected to *translator* while inbound traffic is subject to *filter*. The victim network submits the IP prefixes it wants to protect to the ISP, and the ISP configures the prefixes into the router. The victim network may also submit a list of protocols and/or ports, otherwise the default protocols and ports can be used, e.g., Table II. Each victim corresponds with a *Pisces* instance residing on the SPU. Access Control List (ACL) is used for binding victim (prefixes, protocols and/or ports) to the instance and the SPU where the instance resides.

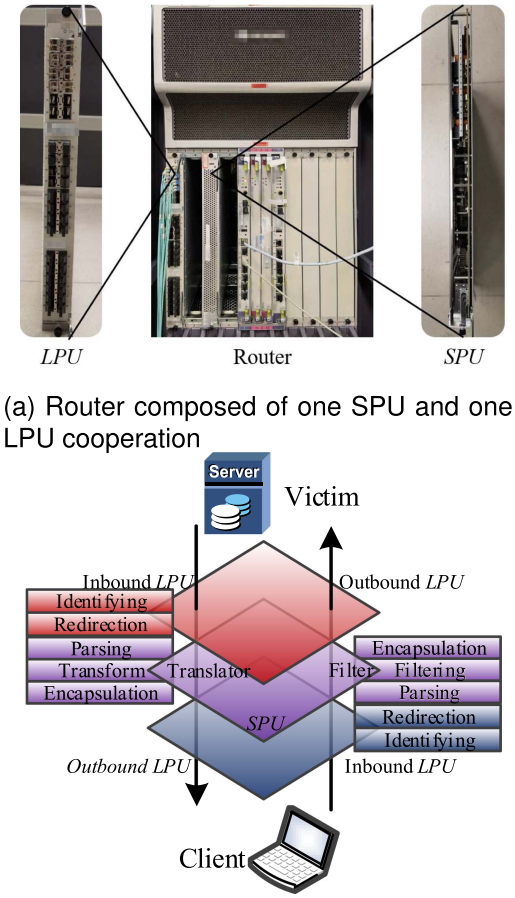


Fig. 5. An implemented routing product and processing logic of LPU and SPU on each packet.

Redirection. Only the packets that hit the ACL will be redirected to the SPU. Each packet is associated with some meta data, which is redirected together with the packet. The meta data includes the target instance, SPU, module (*translator* or *filter*), and other parameters used for layer 3 or layer 2 forwarding. By this means, only the protected traffic consumes SPU resources and experiences additional latency, while other traffic experiences modest affects.

2) SPU: The SPU supports up to 4 ARM CPUs, which are used for service processing. Some cores are configured to perform basic functions, such as header parsing, and load balancing packets to other cores. The rest can be used for *Pisces* services.

Pisces instance context includes the instance keys, code position, code length, etc. The context is stored in the memory and accessible to all CPUs. For maximal balance, per-packet load balancing to all CPUs can be used. However, for better cache hit rate, we implements a per-instance load-balancing. Since the number of instances may not be too big, we can perform fixed bindings for some instances if the load is highly imbalanced.

Once the SPU receives the packet, it first parses the meta data encapsulated in the frame header, and retrieves the instance ID and target module. Then the instance performs the specified *translator* or *filter* function on the packet, according to

Algorithm 1. Finally, the forwarding information base (FIB) is looked up, and the packet is encapsulated and redirected to the outbound LPU.

We use HMAC-SHA1 as the code generation algorithm and implement it using a proprietary cryptography library for ARM. Note that the CPUs we use are capable of hardware hash function acceleration for MD5 and SM3, which is nearly as 10 times efficient as directly using the CPU. For two reasons, we didn't enable the hardware acceleration. First, we choose HMAC-SHA1 as the code generation algorithm, because MD5 is obsolete and SM3 is not as wildly used as SHA1 as a secure hash function. Second, using the CPU computing directly has higher reference value for the community because most commodity CPUs may not have hardware cryptography acceleration.

We also implement a session table to optimize performance and latency, which is especially useful for heavy-hitter flows, such as an elephant flow or a high-rate malicious flow. The packet header and corresponding code are cached in the table, so a packet can just be matched against the table instead of performing cryptographic computing. Note that IPv4 packets cannot support authentication code, and can only be authenticated with the session table. In fact, IPv6 is accelerating its deployment and is being adopted by more and more people. Regarding IPv6 deployment on the global Internet [61], for example, *T-Mobile USA* has achieved 92.31% of IPv6 deployment on June 8, 2022. In addition, from September 23, 2021 to September 23, 2022, the adoption of IPv6 has achieved a relative increase of 18.78% [62]. Therefore, DDoS defense with IPv6 is very significant.

B. Experimental Configuration

In our experiment, we use one LPU and one SPU. There are 2 ARM CPUs, each with 48 cores. Under intense workload, up to 36 cores on each CPU are used for *Pisces* services.

We choose the first 32 bits of the HMAC-SHA1 output as the authentication code. As illustrated in Fig. 2, we put the authentication code in the last 32 bits of the IPv6 address. We use 512-byte IPv6 packets for performance evaluation, unless otherwise specified.

The test traffic can be divided into two categories. The first category is the traffic whose IP addresses, protocols (TCP and UDP) and ports (Table II) are under victim's protection, named as victim traffic. The second category is all other traffic, named as non-victim traffic for simplicity.

The experiment topology is simple. A tester connects to the router's LPU with multiple $10G$ links. When evaluating throughput, we use 5 inbound interfaces and 5 outbound interfaces. When evaluating the impact of defense mechanisms on specific transmission tasks, we use 3 inbound network interfaces (i.e., *InA*, *InB*, and *InC*) and 1 outbound network interface (i.e., *OutA*). Among them, malicious traffic enters the router from *InA* and *InB*, and benign traffic (i.e., files transmitted by *wget*) enters the router from *InC*.

C. Performance Analysis

In terms of performance evaluation and analysis, we first focus on throughput. Moreover, the latency inflation caused by our proposed *Pisces* are discussed. Since our mechanism is processing in the packet granularity, the latency referred to in this paper is the latency of a single packet. Each flow contains multiple data packets. Along with latency, we also utilize jitter to evaluate the robustness of our proposed *Pisces* in a single flow. In addition to evaluating latency inflation, we also utilize FCT to evaluate the impact of our defense mechanism on specific tasks.

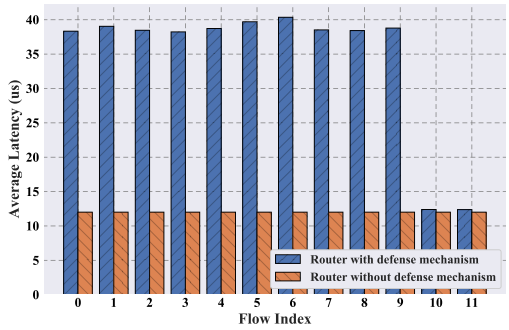
1) Throughput: Since the throughput bottleneck is at the SPU, we solely send victim traffic at the inbound interfaces, and all the traffic is redirected to SPU. We gradually increase inbound rate of the victim traffic rate to test the maximal lossless throughput. In 10 rounds of experiments, packets start to lose at 41 Gbps in every round. No packet is lost at 40 Gbps incoming rate. The loads on the two CPUs are balanced, each with 20 Gbps throughput.

It is expected that if the load is balanced, total throughput grows linearly with the number of CPUs. If there are 2 SPUs, each of which is equipped with 4 CPUs, the maximal total throughput can reach 160 Gbps if the load is balanced. To defend against $Tbps$-scale attacks, one router is not sufficient, and load balancing to multiple routers is required according to our current prototype implementation. However, the current implementation does not utilize all the hardware acceleration power. According to our evaluation of the SM3 hashing algorithm, the throughput of CPU implementation and hardware acceleration on 2 cores are 536,521.39 KBps and 5,310,081.71 KBps, respectively, resulting in a 9.9x performance difference.[8] Besides, our current implementation requires CPUs to perform basic forwarding and load balancing tasks, which can be offloaded to network processors. Hardware acceleration optimization and $Tbps$-scale DDoS defense on a single router are our future work.

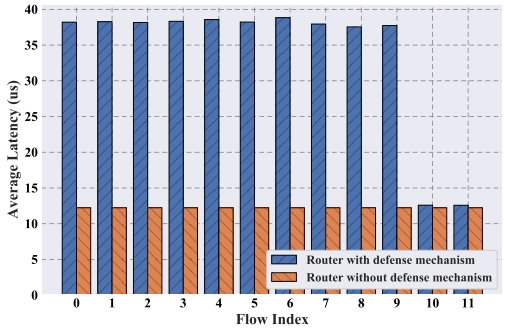
2) Latency Inflation: As introduced in Section III, our proposed *Pisces* consists of two modules, i.e., *translator* and *filter*. These two modules are executed in different stages of data transmission, and we separately evaluate the latency inflation caused by these two modules.

To evaluate the latency inflation caused by *translator*, we first illustrate the average latency through Fig. 6 . Due to the uncertainty of encountering malicious traffic, the router can choose whether to activate the defense mechanism. It can be found from Fig. 6(a) that when the defense mechanism is not activated, no matter which protocol the flow is based on, the packet-granularity latency is always about 12 us. The reason why the latency is so low and stable is that the router simply performs packet forwarding. On the contrary, once our defense mechanism is activated, the latency intensifies. Especially for the flows related to the victim, the latency inflation phenomenon is extremely obvious. This is because the *translator* is required to embed authentication information into the IP address of each packet from the victim. LPU and SPU perform a series of defense-related operations introduced in Section IV-A. As we all know, in order to defend against malicious traffic, latency inflation is inevitable. Note that although the latency inflation phenomenon is obvious, the packet-granularity latency is also controlled within 42 us. Section IV-C-4 will

[8] SM3 in asynchronous mode and the input data size is 16KB.



(a) *Translator*



(b) *Filter*

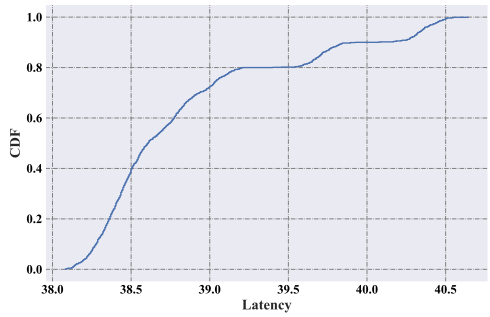
Fig. 6. Comparison of performance when the defense mechanism is activated and not activated.

further confirm that the latency inflation is negligible for specific tasks.

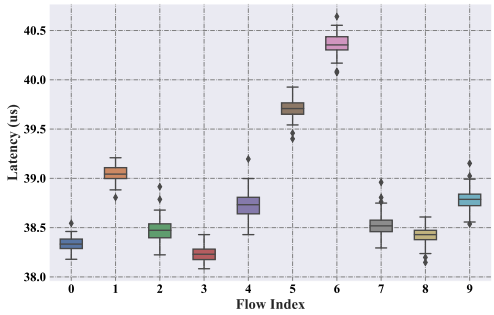
In Fig. 6(a), the two flows on the far right pass through our router, but are not related to the victim. Regarding these two flows, the packet-granularity latency slightly expanded, but the scale of expansion is less than 0.5 us. Although the router activates our *Pisces*, the packets of these two flows are not required to be embedded with authentication information. Compared with the scenario where the defense mechanism is not activated, the router simply judges whether the packet comes from the victim, which incurs a small additional latency. This also further demonstrates that our proposed *Pisces* only works on the victim's packets, and it is very friendly to the non-victim's packets.

To prevent malicious traffic from attacking the victim, the *filter* is required to distinguish whether the packet is benign or malicious. Fig. 6(b) illustrates the latency inflation caused by the *filter*. It can be found that, the changes caused by the activation of the defense mechanism are very similar to those in Fig. 6(a). Overall, for packets sent to the victim, the latency inflation caused by the *filter* is within a controllable range and very stable. For packets sent to non-victims, the *filter* caused almost no performance change. This illustrates that the *filter* is friendly to non-victim data packets.

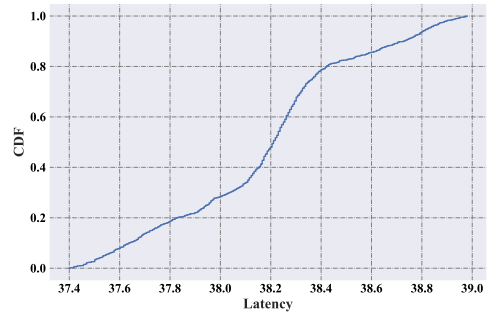
The indicators in Fig. 6 are the average performance of multiple evaluations. Regarding the evaluation of stability, we further provide the distribution of various indicators during multiple evaluations through Fig. 7. Note that the relevant results of Fig. 7 are collected under the situation that the router activates the defense mechanism.



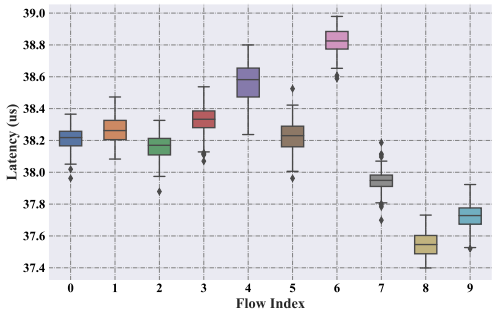
(a) Overall distribution of latency with the *translator*



(b) Detailed distribution of latency with the *translator*



(c) Overall distribution of latency with the *filter*



(d) Detailed distribution of latency with the *filter*

Fig. 7. Distribution of victim-related benign flows' packet-granularity latency during multiple evaluations.

First of all, we put together the packet-granularity latency of all flows related to the victim, and plot the cumulative distribution function (CDF) in Fig. 7(a). It can be found that

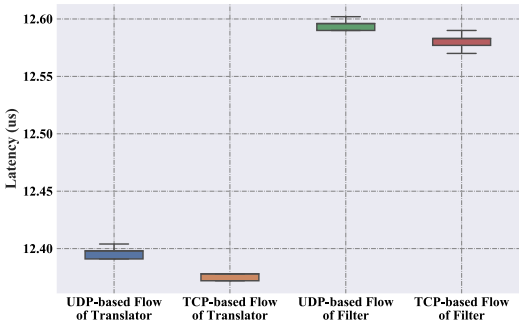


Fig. 8. Distribution of non-victim-related benign flows' latency.

more than 80% of the latency indicators are less than 39.5 us. Then, Fig. 7(b) separately illustrates the distribution of latency indicators for flows based on different protocols in multiple evaluations. For each individual flow, the upper and lower spacing of each rectangular box in Fig. 7(b) is less than 0.2 us, which means that the packet-granularity latency changes slightly during multiple evaluations. By comparing the rectangular boxes of different flows, it can be found that the difference in the protocol will cause the difference in latency. However, this difference does not exceed 2.5 us, which is very small. Regarding the phenomenon related to the *filter* in Fig. 7(c) and (d), it is consistent with the latency distribution of the *translator*. This once again demonstrates that the scale of latency inflation has stable characteristics.

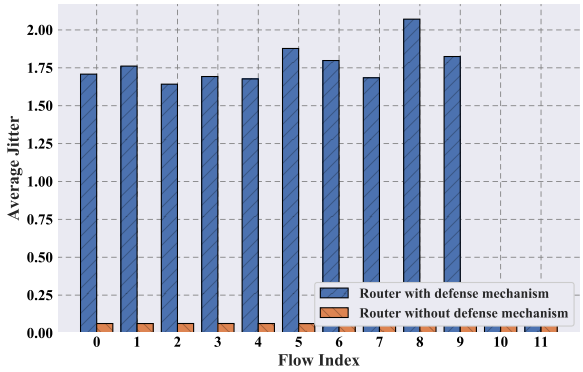
While the router assists the victim in defending against attacks, the router also serves other non-victims. We have separately evaluated the UDP and TCP protocols, as illustrated in the Fig. 8. With multiple evaluations, whether it is TCP or UDP, the latency inflation caused by the *filter* is more obvious. Although there is latency inflation, the scale is small. Moreover, the results of multiple evaluations are also relatively concentrated. This demonstrates that our defense mechanism is stable and friendly to non-victims.

3) Jitter Inflation: Similar to latency inflation, we also evaluate the newly proposed *Pisces* on jitter inflation. The referred jitter in this paper is defined as the absolute error in delay between the current packet and the previous packet. In other words, the jitter described in this paper is not the error of the maximum latency and the minimum latency of a flow, but is refined to the latency error between every two adjacent packets. Therefore, the jitter here can describe the fluctuation of a flow during transmission.

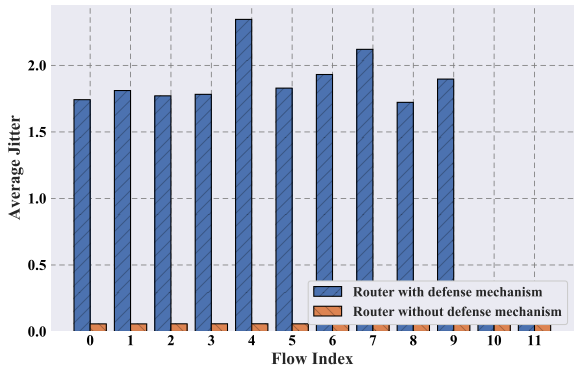
Regarding the average jitter caused by the *translator* and *filter*, Fig. 9 illustrates the performance evaluation. Similar to latency, the jitter increases after the defense mechanism is activated. When the defense mechanism is activated, the latency change (i.e., jitter) of two adjacent packets is also within 2.5 us, indicating that our *translator* is very stable.

Overall, no matter the average value (i.e., Fig. 9) or the distribution (i.e., Figs. 10 and 11), the conclusions about jitter are consistent with the conclusions about latency.

4) FCT Comparison: In addition to the latency inflation, we use a variety of different scale transmission tasks to evaluate



(a) *Translator*



(b) *Filter*

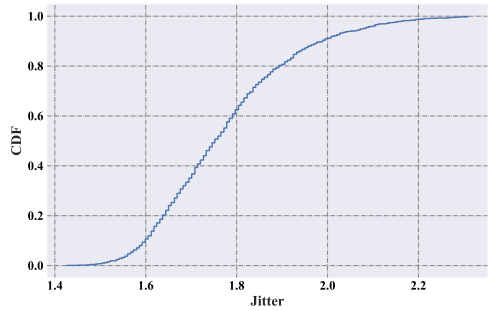
Fig. 9. Comparison of jitter when the defense mechanism is activated and not activated.

the impact of defense mechanisms on specific tasks. More specifically, we utilize *wget* to transfer files of different sizes. While transferring files, we created a variety of attack scenarios of different scales. Thus, we evaluate the positive and negative effects of our *Pisces* on FCT in different scenarios.

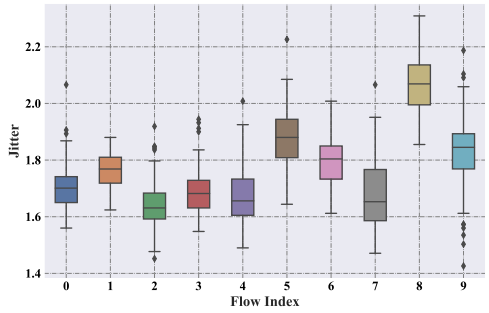
Fig. 12 illustrates the FCT in different scenarios after the defense mechanism is activated. Here, the X-axis value refers to the percentage of attack traffic per inbound network interface. First of all, for a small file of $10M$, the corresponding curve in Fig. 12 is basically a straight line, indicating that the FCT is very stable. This can also be found in Table III. Except for the last column which is $0.02s$, the other columns are all $0.03s$. When the file is enlarged to $100M$, the FCT displayed by *wget* is always $0.3s$,[9] which is a linear expansion of 10 times. When the file expands to 1G or even 10G, the corresponding FCTs in Fig. 12 also expand with slight fluctuations. These slight fluctuations are errors accumulated in relatively long transmissions, not caused by malicious traffic.

For a fixed-size file in Fig. 12 (i.e., each curve in Fig. 12), when the scale of malicious traffic increases (i.e., the X-axis value), the performance of FCT hardly changes. This illustrates that our *Pisces* can accurately prevent malicious blocking file transfers between the victim and other benign hosts. Moreover,

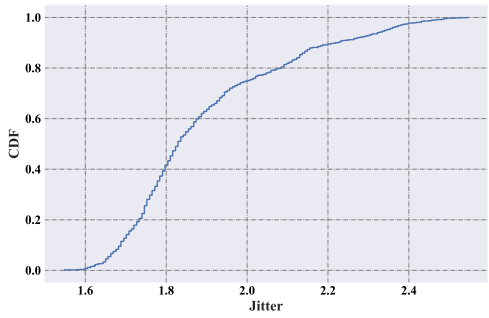
[9]Note that since we cannot specify the number of decimal places displayed by *wget*, the accuracy of FCT displayed for files of different sizes is different. For example, the FCT of a 10M file is accurate percentile, while the FCT of a $100M$ file is only accurate to the tenth.



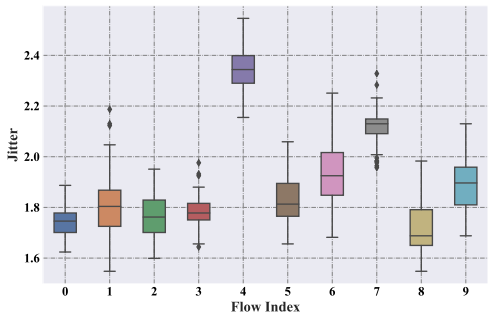
(a) Overall distribution of jitter with the *translator*



(b) Detailed distribution of jitter with the *translator*



(c) Overall distribution of jitter with the *filter*



(d) Detailed distribution of jitter with the *filter*

Fig. 10. Distribution of victim-related benign flows' packet-granularity jitter during multiple evaluations.

our proposed *Pisces* can completely defend against malicious traffic of any scale.

In fact, if our defense mechanism is not activated, attackers can easily achieve malicious purposes. Table III presents the FCT evaluation results of files with various sizes when the defense mechanism is activated and not activated. When there is no malicious traffic (i.e., the leftmost column in Table III),

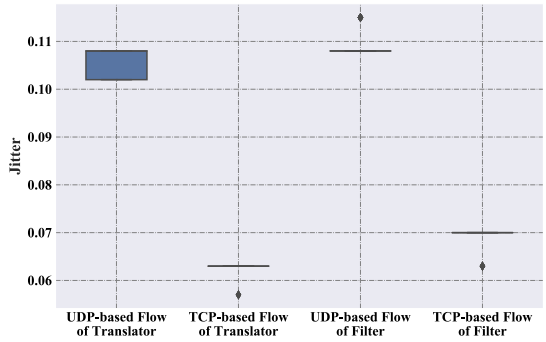


Fig. 11. Distribution of non-victim-related benign flows' jitter.

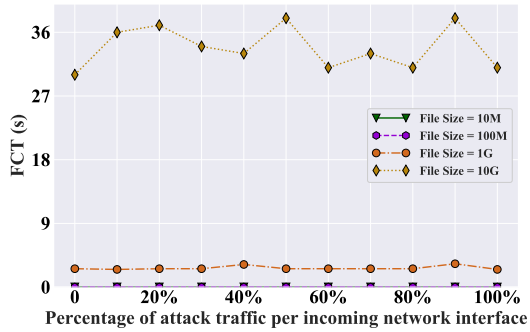


Fig. 12. FCT when coexisting with attack traffic.

whether the defense mechanism is activated or not does not make FCT change significantly. After activating the defense mechanism, some FCT results are even better, e.g., $2.6s < 3.4s$ and $30s < 37s$. That is to say, although our *Pisces* causes the packet-granularity latency inflation, it does not cause obvious additional costs for specific tasks. Note that low cost is a key factor that can facilitate real-world deployments [38], [43]. When the malicious traffic per inbound network interface increases to 40%, the attacker has the ability to effectively attack the victim. If our defense mechanism is not activated, the transmission time of $10M$ file will suddenly increase from $0.02s$ to $5m55s$. If the file is larger, or the malicious traffic is more, the similar negative impact will be more obvious.[10] This will result in the victim being unable to provide any services to the outside world. On the contrary, as long as our defense mechanism is activated, attacks of any scale cannot affect the victim's external services.

V. Discussion

A. Scalability

In addition to the refined design of the system (i.e., Section III) and the experiments with high-performance commercial routers (i.e., Section IV), we further analyze that the *Pisces* has good scalability in multiple aspects, e.g., deployment strategy, integration with existing cybersecurity system, and implementability of IPv4.

[10]Due to the low transmission rate in some experiments, we only recorded the remaining transmission time displayed by *wget* after a period of transmission. For example, $> 85m$ or $> 20h24m$. The dashes in Table III indicate that the transmission rate is too low and the transmission task fails.

TABLE III
PERFORMANCE COMPARISON OF OUR PROPOSED *PISCES* IN FCT

File Size	Defense or not	Percentage of attack traffic per incoming network interface (second)										
		0	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
$10M$	without defense	0.02	0.03	0.02	0.02	$5m55s$	$> 25m42s$	$> 33m27s$	–	–	–	–
	with defense	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.02
$100M$	without defense	0.3	0.3	0.3	0.3	$> 85m$	$6h10m$	$> 4h35m$	–	–	–	–
	with defense	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3
$1G$	without defense	3.4	2.6	2.6	2.9	$> 20h24m$	$3d15h$	$> 9d10h$	–	–	–	–
	with defense	2.6	2.5	2.6	2.6	3.2	2.6.	2.6	2.6	2.6	3.3	2.5
$10G$	without defense	37	34	30	33	$> 6d2h$	$33d12h$	$> 40d18h$	–	–	–	–
	with defense	30	36	37	34	33	38	31	33	31	38	31

Deployment Strategy. In addition to following the implementation details given in Sections IV-A and IV-B, network administrators and cybersecurity professionals can further optimize the implementation and management of *Pisces* by deploying strategies. Logically, *filter* and *translator* can be deployed separately. For example, the victim network can deploy *translator* by itself, and let the upstream ISPs deploy *filter*. This is especially helpful if the victim network employs multiple providers and uses provider-independent addresses. In this case, the victim can share its key with all its providers to use. Specifically, the key can be periodically updated by the victim, e.g., the last day of every month. Moreover, on the second day after the attack stops, the victim will update the key in time. So even if the round-trip paths are asymmetric, the code can still be correctly processed.

Integration with Existing Cybersecurity System. AI-based cybersecurity system are one of the emerging DDoS defenses. In fact, our *Pisces* can be bidirectionally integrated with AI-based DDoS defense, i.e., *Pisces* for AI-based DDoS defense and AI-based DDoS defense for *Pisces*. Regarding the former, the training of AI methods relies heavily on high-quality data. The *Pisces* does not require any data support and can detect zero-day DDoS attacks. These detected attack behaviors can be used as training data for AI-based DDoS defenses. Regarding the latter, when an attacker replays a legitimate packet, the *Pisces* cannot detect it immediately, resulting in a short-term attack. At this time, AI-based DDoS defenses can analyze the behavior of the packet from a more macro perspective, thereby making up for the shortcomings of the *Pisces*.

Implementability of IPv4. The implementation of the scheme proposed in this paper relies on a big available address space of IPv6 to carry verifiable information. However, it is impractical to carry verifiable information in an IPv4 address with the limited space. To defend against IPv6 traffic attacks, we can reserve 64-bit, 48-bit, or 32-bit space in the IP address of the protected server to set authentication code.

However, for 32-bit IPv4 addresses, there is not enough space for embedding verifiable information. In fact, the method to prevent reflective amplification attacks can also be applied in IPv4 traffic. We can adopt the 16-bit source port number of the outgoing traffic to place the authentication code. For example, even if a 12-bit authentication code is embedded in the port number, the attack traffic can also be reduced by 2^{12} times when a guess attack is considered. To prevent flooding attacks, if modifying the client's program is possible, the source port number can also be used to carry the authentication code. However, it faces huge challenges to require all the clients to cooperate to modify its method of using the source port number. Therefore, some traditional defense solutions may also need to be deployed if there is still large volume of IPv4 attack traffic.

B. Security, Privacy, and Ethical

In addition to improving the security of the *Pisces* through design details and hardware implementation, we further analyze threats such as replay attack, as well as privacy and ethics.

Replay Attack. Attackers may get a legitimate packet and then replay it, which can enable malicious traffic pass through the *filter*. While *Pisces* cannot prevent all replayed packets (it doesn't have to because the goal of *Pisces* is merely to mitigate DDoS traffic), it still has a lot of benefits. First, it greatly raises the difficulty and cost of the attacker. In the reflection attack, the attacker cannot get feedback whether a packet passes through a *filter* or not, unless it can monitor the link from the *filter* to the victim. So the brute-force success rate is $2^{-\tau}$ where τ is the code length. In the flooding attack, a malicious host will get one packet through in $2^{\tau-1}$ tries on average. If the attacker replays the packet from multiple hosts, the *filter* will see a huge amount of traffic from that packet. Then the router can just set a blacklist blocking that source-destination IP pair. If *Pisces* is not used, blacklist can be easily bypassed since the attacker can used forged source addresses.

Privacy and Ethical. Our *Pisces* only transforms and filters IP addresses when forwarding packets based on IP addresses, and does not check other fields of the packet (e.g., payload) at all. Moreover, the *Pisces* can operate normally even if application-layer is encrypted. Therefore, our *Pisces* has inherent privacy protection capabilities. Accordingly, the *Pisces* objectively defends against DDoS attacks based on verifiable information without causing any ethical issues.

VI. Conclusion and Future Directions

To defend the most common two types of DDoS attacks, we propose the legitimate traffic marking and authentication technology to prevent amplification attacks, and we further present insensible AID distributing and stateless verification technology to mitigate flooding attacks. Moreover, neither the cooperation with multiple parties nor the modification of the protocol stack is required to implement the defense protocol stack. Overall, our *Pisces* enable network devices to efficiently identify malicious and legitimate traffic based on the authentication code in the packet header.

Although the *Pisces* has achieved good performance, it can still be further optimized, including the following directions:

1) Expansion of IPv4. By utilizing the redundant space of IPv6 address, the newly proposed *Pisces* achieve excellent defense performance on IPv6 traffic. However, due to limited IPv4 address space, it is difficult to embed the authentication code in the header of IPv4 address. In the future, we will continue to explore easy-to-deploy anti-DDoS solutions for IPv4 traffic.

2) Friendly Rekey. Rekey is easy for reflection attacks because the router does not require cooperation with clients. During rekeying period, the router can simply keeping the old and new keys, and the code verified by either key can pass the *filter*. However, the rekey for flooding attack is challenging. One sort is to rekey at peace time without verifying code. But the problem is that under attack period, the clients who do not have an up-to-date code will be filtered out. We leave the design of more friendly rekey method for future work.

3) Integration with AI methods. AI technologies such as deep learning and large models have good performance in DDoS attacks. Moreover, the *Pisces* can be bidirectionally integrated with AI-based defenses. In the future, we can further analyze the deep integration and joint deployment of *Pisces* and AI technologies to achieve reliable, accurate and real-time DDoS defense.

4) Incentives. As a new mechanism, *Pisces* can effectively defend against DDoS attacks. However, improving cyberspace security still requires upstream ISPs and enterprises to deploy *Pisces*. In addition to optimizing the mechanism, an incentive model can be designed from an economic perspective to promote the widespread deployment of *Pisces*.

Acknowledgments

The authors would like to thank the editors and anonymous reviewers for their constructive comments and guidance.

References

[1] J. Mirkovic and P. Reiher, "A taxonomy of DDoS attack and DDoS defense mechanisms," in *Proc. ACM SIGCOMM Comput. Commun. Rev.*, 2004, pp. 39–53.

[2] T. Anderson, T. Roscoe, and D. Wetherall, "Preventing internet denial-of-service with capabilities," in *Proc. ACM SIGCOMM Comput. Commun. Rev.*, 2004, pp. 39–44.

[3] Z. Liu, H. Jin, Y.-C. Hu, and M. Bailey, "MiddlePolice: Toward enforcing destination-defined policies in the middle of the internet," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 1268–1279.

[4] N. Systems, "NETSCOUT threat intelligence report," 2020. [Online], Available: https://www.netscout.com/threatreport

[5] O. Yoachimik and J. Pacheco, "DDoS threat report for 2023 Q 4," 2024. [Online], Available: https://blog.cloudflare.com/ddos-threat-report-2023-q4

[6] K. Lab, "DDoS breach costs rise to over 2 M for enterprises finds Kaspersky lab report," 2018. [Online], Available: https://usa.kaspersky.com/about/press-releases/2018_ddos-breach-costs-rise-to-over-2m-for-enterprises-finds-kaspersky-lab-report

[7] EINNEWSDESK, "DDoS protection and mitigation market size, share and growth analysis for 2024–2033," 2024. [Online], Available: https://tech.einnews.com/amp/pr_news/694330813/ddos-protection-and-mitigation-market-size-share-and-growth-analysis-for-2024-2033

[8] R. Mohammadi, R. Javidan, and M. Conti, "SLICOTS: An SDN-Based lightweight countermeasure for TCP SYN flooding attacks," *IEEE Trans. Netw. Service Manage.*, vol. 14, no. 2, pp. 487–497, Jun. 2017.

[9] K. Lakshminarayanan, D. Adkins, A. Perrig, and I. Stoica, "Taming IP packet flooding attacks," in *Proc. ACM SIGCOMM Comput. Commun. Rev*, 2004, pp. 45–50.

[10] A. S. Mamolar, P. Salvá-García, E. Chirivella-Perez, Z. Pervez, J. M. A. Calero, and Q. Wang, "Autonomic protection of multi-tenant 5G mobile networks against UDP flooding DDoS attacks," *J. Netw. Comput. Appl.*, vol. 145, 2019, Art. no. 102416.

[11] M. Kührer, T. Hupperich, C. Rossow, and T. Holz, "Exit from hell? Reducing the impact of amplification DDoS attacks," in *Proc. USENIX Secur.*, 2014, pp. 111–125.

[12] A. Mishra, N. Gupta, and B. Gupta, "Defense mechanisms against DDoS attack based on entropy in SDN-Cloud using POX controller," *Telecommun. Syst.*, vol. 77, no. 1, pp. 47–62, 2021.

[13] A. Yaar, A. Perrig, and D. Song, "SIFF: A stateless internet flow filter to mitigate DDoS flooding attacks," in *Proc. IEEE Symp. Secur. Privacy*, 2004, pp. 130–143.

[14] X. Yang, D. Wetherall, and T. Anderson, "A DoS-limiting network architecture," in *Proc. ACM SIGCOMM Conf. Appl., Technol., Architectures, Protoc. Comput. Commun.*, 2005, pp. 241–252.

[15] S. M. M. Mirnajafizadeh, A. R. Sethuram, D. Nyang, and R. Jang, "Enhancing network attack detection with distributed and in-network data collection system," in *Proc. USENIX Secur. Symp.*, 2024, pp. 5161–5178.

[16] 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.

[17] Z. Liu, H. Jin, Y.-C. Hu, and M. Bailey, "Practical proactive DDoS-Attack mitigation via endpoint-driven in-network traffic control," *IEEE/ACM Trans. Netw.*, vol. 26, no. 4, pp. 1948–1961, Aug. 2018.

[18] M. Tran, M. S. Kang, H.-C. Hsiao, W.-H. Chiang, S.-P. Tung, and Y.-S. Wang, "On the feasibility of rerouting-based DDoS defenses," in *Proc. IEEE Symp. Secur. Privacy*, Oakland, 2019, pp. 1169–1184.

[19] S. Ramanathan, J. Mirkovic, M. Yu, and Y. Zhang, "SENSS against volumetric DDoS attacks," in *Proc. Annu. Comput. Secur. Appl. Conf.*, 2018, pp. 266–277.

[20] T. King, C. Dietzel, J. Snijders, G. Doering, and G. Hankins, "BLACKHOLE community," *IETF RCF 7999*, pp. 1–9, 2016. [Online] Available: https://datatracker.ietf.org/doc/html/rfc7999

[21] L. Jiao, R. Zhou, X. Lin, and X. Chen, "Online scheduling of traffic diversion and cloud scrubbing with uncertainty in current inputs," in *Proc. ACM Int. Symp. Ad Hoc Mobile Netw. Mobile Comput.*, 2019, pp. 271–280.

[22] S. K. Fayaz, Y. Tobioka, V. Sekar, and M. Bailey, "Bohatei: Flexible and elastic DDoS defense," in *Proc. USENIX Secur.*, 2015, pp. 817–832.

[23] W. You, L. Jiao, J. Li, and R. Zhou, "Scheduling DDoS cloud scrubbing in ISP networks via randomized online auctions," in *Proc. IEEE Conf. Comput. Commun.*, 2020, pp. 1658–1667.

[24] T. Alharbi, A. Aljuhani, and H. Liu, "Holistic DDoS mitigation using NFV," in *Proc. IEEE 7th Annu. Comput. Commun. Workshop Conf.*, 2017, pp. 1–4.

[25] A. Sadeghian and M. Zamani, "Detecting and preventing DDoS attacks in botnets by the help of self triggered black holes," in *Proc. Asia-Pacific Conf. Comput. Aided Syst. Eng.*, 2014, pp. 38–42.

[26] C. Dietzel, A. Feldmann, and T. King, "Blackholing at IXPs: On the effectiveness of DDoS mitigation in the wild," in *Proc. Int. Conf. Passive Act. Meas.*, 2016, pp. 319–332.

[27] S. Agarwal, T. Dawson, and C. Tryfonas, "Mitigation via regional cleaning centers," Sprint ATL Research, Tech. Rep. RR04-ATL-013177, 2003.

[28] Z. Liu, Y. Cao, M. Zhu, and W. Ge, "Umbrella: Enabling ISPs to offer readily deployable and privacy-preserving DDoS prevention services," *IEEE Trans. Inf. Forensics Secur.*, vol. 14, no. 4, pp. 1098–1108, Apr. 2019.
[29] A. Yaar, A. Perrig, and D. Song, "StackPi: New packet marking and filtering mechanisms for DDoS and IP spoofing defense," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 10, pp. 1853–1863, Oct. 2006.
[30] I. B. Mopari, S. G. Pukale, and M. L. Dhore, "Detection and defense against DDoS attack with IP spoofing," in *Proc. Int. Conf. Comput., Commun. Netw.*, 2008, pp. 1–5.
[31] L. Kavisankar and C. Chellappan, "A mitigation model for TCP SYN flooding with IP spoofing," in *Proc. Int. Conf. Recent Trends Inf. Technol.*, 2011, pp. 251–256.
[32] A. Belenky and N. Ansari, "IP traceback with deterministic packet marking," *IEEE Commun. Lett.*, vol. 7, no. 4, pp. 162–164, Apr. 2003.
[33] K. Park and H. Lee, "On the effectiveness of probabilistic packet marking for IP traceback under denial of service attack," in *Proc. IEEE Conf. Comput. Commun.*, 2001, pp. 338–347.
[34] C. Jin, H. Wang, and K. G. Shin, "Hop-count filtering: An effective defense against spoofed DDoS traffic," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2003, pp. 30–41.
[35] J. Ioannidis and S. M. Bellovin, "Implementing pushback: Router-based defense against DDoS attacks," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2002, pp. 1–12.
[36] J. Li et al., "Toward adaptive DDoS-Filtering rule generation," in *Proc. IEEE Conf. Commun. Netw. Secur.*, 2023, pp. 1–9.
[37] I. Cvitić, D. Perakovic, B. B. Gupta, and K.-K. R. Choo, "Boosting-based DDoS detection in Internet of Things systems," *IEEE Internet Things J.*, vol. 9, no. 3, pp. 2109–2123, Feb. 2022.
[38] A. Mishra, B. K. Joshi, V. Arya, A. K. Gupta, and K. T. Chui, "Detection of distributed denial of service (DDoS) attacks using computational intelligence and majority vote-based ensemble approach," *Int. J. Softw. Sci. Comput. Intell.*, vol. 14, no. 1, pp. 1–10, 2022.
[39] A. D. Keromytis, V. Misra, and D. Rubenstein, "SOS: Secure overlay services," in *Proc. ACM Conf. Appl., Technol., Architectures, Protoc. Comput. Commun.*, 2002, pp. 61–72.
[40] C. Dixon, T. E. Anderson, and A. Krishnamurthy, "Phalanx: Withstanding multimillion-node botnets," in *Proc. Symp. Netw. Syst. Des. Implementation*, 2008, pp. 45–58.
[41] D. G. Andersen, H. Balakrishnan, N. Feamster, T. Koponen, D. Moon, and S. Shenker, "Accountable internet protocol (AIP)," in *Proc. ACM Conf. Appl., Technol., Architectures, Protoc. Comput. Commun.*, 2008, pp. 339–350.
[42] B. Parno, D. Wendlandt, E. Shi, A. Perrig, B. Maggs, and Y.-C. Hu, "Portcullis: Protecting connection setup from denial-of-capability attacks," in *Proc. ACM Conf. Appl., Technol., Architectures, Protoc. Comput. Commun.*, 2007, pp. 289–300.
[43] M. Zhang et al., "Poseidon: Mitigating volumetric DDoS attacks with programmable switches," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2020, pp. 1–18.
[44] Z. Liu et al., "Jaqen: A. high-performance switch-native approach for detecting and mitigating volumetric DDoS attacks with programmable switches," in *Proc. USENIX Secur. Symp.*, 2021, pp. 3829–3846.
[45] S. Yoo, X. Chen, and J. Rexford, "SmartCookie: Blocking large-scale SYN floods with a split-proxy defense on programmable data planes," in *Proc. USENIX Secur. Symp.*, 2024, pp. 217–234.
[46] R. Doshi, N. Apthorpe, and N. Feamster, "Machine learning DDoS detection for consumer Internet of Things devices," in *Proc. IEEE Secur. Privacy Workshops*, 2018, pp. 29–35.
[47] C. Fu, Q. Li, M. Shen, and K. Xu, "Frequency domain feature based robust malicious traffic detection," *IEEE/ACM Trans. Netw.*, vol. 31, no. 1, pp. 452–467, Feb. 2023.
[48] S. U. Jafri, S. Rao, V. Shrivastav, and M. Tawarmalani, "Leo: Online ML-based traffic classification at multi-terabit line rate," in *Proc. USENIX Symp. Netw. Syst. Des. Implementation*, 2024, pp. 1573–1591.
[49] G. Zhou, Z. Liu, C. Fu, Q. Li, and K. Xu, "An efficient design of intelligent network data plane," in *Proc. 32nd USENIX Secur. Symp.*, 2023, pp. 6203–6220.
[50] J. Yan et al., "Brain-on-switch: Towards advanced intelligent network data plane via NN-Driven traffic analysis at line-speed," in *Proc. USENIX Symp. Netw. Syst. Des. Implementation*, 2024, pp. 419–440.
[51] Y. Zhao, K. Xu, J. Chen, and Q. Tan, "Collaboration-enabled intelligent internet architecture: Opportunities and challenges," *IEEE Netw.*, vol. 36, no. 5, pp. 98–105, Sep./Oct. 2022.
[52] C. Fu, Q. Li, M. Shen, and K. Xu, "Detecting tunneled flooding traffic via deep semantic analysis of packet length patterns," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2024, pp. 3659–3673.
[53] Y. Feng, J. Li, and T. Nguyen, "Application-layer DDoS defense with reinforcement learning," in *Proc. IEEE/ACM Int. Symp. Qual. Service*, 2020, pp. 1–10.
[54] W. Xu et al., "TsuKing: Coordinating DNS resolvers and queries into potent DoS amplifiers," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2023, pp. 311–325.
[55] R. McLaughlin, F. Pagani, N. Spahn, C. Kruegel, and G. Vigna, "Regulator: Dynamic analysis to detect ReDoS," in *Proc. USENIX Secur. Symp.*, 2022, pp. 4219–4235.
[56] X. Wang et al., "Effective ReDoS detection by principled vulnerability modeling and exploit generation," in *Proc. IEEE Symp. Secur. Privacy*, 2023, pp. 2427–2443.
[57] Wikipedia, "HKDF," 2024. [Online], Available: https://en.wikipedia.org/wiki/HKDF
[58] J. Iyengar and M. Thomson, "QUIC: A UDP-based multiplexed and secure transport," *IETF Internet-Draft*, pp. 1–129, 2018. [Online] Available: https://datatracker.ietf.org/doc/draft-ietf-quic-transport/14/
[59] Q. De Coninck et al., "Pluginizing QUIC," in *Proc. ACM Conf. Appl., Technol., Architectures, Protoc. Comput. Commun.*, 2019, pp. 59–74.
[60] H. Krawczyk and H. Wee, "The OPTLS protocol and TLS 1.3," in *Proc. IEEE Eur. Symp. Secur. Privacy*, 2016, pp. 81–96.
[61] W. I. Lanuch, "Network operator measurements," Jun. 08, 2022. [Online], Available: https://www.worldipv6launch.org/measurements/
[62] Google, "IPv6 statistics," 2022. [Online], Available: https://www.google.com/intl/en/ipv6/statistics.html

Yi Zhao (Member, IEEE) received the BEng degree from the School of Software and Microelectronics, Northwestern Polytechnical University, Xi'an, China, in 2016, and the PhD degree from the Department of Computer Science and Technology, Tsinghua University, Beijing, China, in 2021. From 2021 to 2023, he served as an assistant researcher and postdoctoral fellow with the Department of Computer Science and Technology, Tsinghua University, where he is a recipient of the Shuimu Tsinghua Scholar Program. Currently, he is an associate researcher with the School of Cyberspace Science and Technology, Beijing Institute of Technology. His research interests include privacy protection, data security, network security, machine learning, and adversarial attack defense. He is a member of ACM.

Bingyang Liu received the PhD degree from the Department of Computer Science and Technology, Tsinghua University, in 2014. He was a postdoctoral researcher with Institute for Network Sciences and Cyberspace, Tsinghua University from 2014 to 2016. Now he is the director of Network Technology Lab in Huawei, and the vice chairman with the Network Technology Committee of China Institute of Communications. In Huawei, he is also a chief expert in the deterministic network in data communication field, and a senior researcher in data center networks and wide-area networks.

Weiyu Jiang received the PhD degree from the University of Chinese Academy of Sciences, in 2015. Currently, she is a technical expert with the Network Technology Lab, and a chief expert in network intrinsic security field in Huawei. Her research interests include network security architecture and protocols, cloud security and AI cluster network security.

Ke Xu (Fellow, IEEE) received the PhD degree from the Department of Computer Science and Technology of Tsinghua University, Beijing, China, where he serves as a full professor. He has published more than 200 technical papers and holds 11 US patents in the research areas of next-generation Internet, Blockchain systems, Internet of Things, and network security. He is a member of ACM. He has guest-edited several special issues in IEEE and Springer Journals. He is an editor of *IEEE Internet of Things Journal*. He was Steering committee chair of IEEE/ACM IWQoS.

Qi Li (Senior Member, IEEE) received the PhD degree from Tsinghua University. He is currently an associate professor with the Institute for Network Sciences and Cyberspace, Tsinghua University. He has worked with ETH Zurich and the University of Texas at San Antonio. His research interests include network and system security, particularly in Internet and cloud security, mobile security and Big Data security. He is currently an editorial board member of the *IEEE Transactions on Dependable and Secure Computing* and *ACM Digital Threats: Research and Practice*.

Chuang Wang received the BE degree from the Harbin Institute of Technology. He was a technical expert with the Network Technology Lab of Huawei. His research areas include future network architecture and techniques.

Zongxin Dou joined Huawei in 2010. Now he is the senior engineer with Network Technology Lab. His research areas include data communication protocols and network security.

Yanjun Liu received the BE degree from Northeastern University in 1999. He joined Huawei in 2005. He is currently with the Network Technology Lab working on topics in systems, networking, and cybersecurity research.