Hindawi
Mobile Information Systems
Volume 2021, Article ID 1257046, 24 pages
https://doi.org/10.1155/2021/1257046

Review Article

A Review of P4 Programmable Data Planes for Network Security

Ya Gao and Zhenling Wang

Wuxi Institute of Technology, Wuxi 214121, China

Correspondence should be addressed to Ya Gao; gaoya@wxit.edu.cn

Received 6 October 2021; Revised 19 October 2021; Accepted 27 October 2021; Published 23 November 2021

Academic Editor: Sang-Bing Tsai

Network attacks show a trend of increased attack intensity, enhanced diversity, and more concealed attack methods, which put forward higher requirements for the performance of network security equipment. Unlike the SDN (software defined network) switch with a fixed-function data plane, switches with programmable data planes can help users realize more network protocols. Programming Protocol-independent Packet Processors (P4) is proposed to define the operations of the data plane and to implement user's applications, e.g., data center networks, security, or 5G. This paper provides a review of research papers on solving network security problems with P4-based programmable data plane. The work can be organized into two parts. In the first part, the programming language P4, P4 program, architectures, P4 compilers, P4 Runtime, and P4 target are introduced according to the workflow model. The advantages of P4-based programmable switching in solving network security are analyzed. In the second part, the existing network security research papers are divided into four parts according to the perspectives of passive defense, active defense, and combination of multiple technologies. The schemes in each category are compared, and the core ideas and limitations are clarified. In addition, a detailed comparison is made for the research on the performance of P4 targets. Finally, trends and challenges related to the P4-based programmable data plane are discussed.

1. Introduction

The rapid development of industrial IoT (Internet of Things), cloud computing, AI (Artificial Intelligence), and machine learning technology has helped enterprises and economic entities to catch the digital express train and promote the continuous and rapid growth of network bandwidth. Affected by the epidemic, various companies move to their online business. The ensuing threats related to network security are also on the rise.

A recent report [1] pointed out that the number of DDoS (distributed denial of service) attacks has been increasing rapidly for 4 consecutive years, and the magnitude, methods, and sources of attacks have all been significantly upgraded compared to the past. Among them, the number of high-traffic attacks above 100G in the first half of 2021 increased by more than half compared with the number of the same period last year. In addition, the number of hijacked hosts participating in DDoS attacks was only half of the hosts of the last year, but these hosts carried out more than four times the number of attacks. DDoS attacks adopt more ingenious methods to implement attacks, and pave the way for other serious network attacks, increasing the difficulty of identification and defense. The CrowdStrike [2] stated that based on data from July 1, 2020, to June 30, 2021, it was deduced that the attacker's breakthrough time was significantly reduced, with an average of only 1 hour and 32 minutes. Moreover, in 36% of intrusions, attackers can complete the attack, discover important data, or deploy ransomware in less than 30 minutes. The phenomenon of obtaining packets through sniffing and brute-force cracking of passwords has gradually increased with the significant improvement of computing capacity, and the problem of data leakage has become more and more serious. Powerful professional sniffing tools and cracking tools, such as Cain & Abel, dSniff, Hashcat, and THC Hydra, have exacerbated this phenomenon.

Network attacks show a trend of increased attack intensity, enhanced diversity, and more concealed attack methods, which put forward higher requirements for the performance of network security equipment. Traditional network security equipment, such as firewalls and encryption cards, is designed and implemented based on a wealth of prior knowledge to defend against most network attacks. However, with the continuous development of network attack and defense technologies, it is more difficult to identify network attacks. Traditional network security equipment lacks flexibility and has difficulty continuously responding to emerging network attacks.

In the OpenFlow-based software defined network [3], monitoring, data collection, and detection are implemented by logically centralized components (such as controllers). This requires a large amount of information related to the network status to be transmitted from the network device to the controller for analysis and processing. A single point of failure of the controller and the communication overhead between the centralized control plane and the data plane are the bottlenecks that limit the performance of its security defense function. The deployment of programmable defense measures in the SDN controller can only mitigate network attacks to a certain extent, because the software controller cannot achieve the ability to directly handle the number and speed of data plane-level flows. It is difficult to achieve fine identification and comprehensive defense by sampling instead of packet-by-packet inspection, especially for measures such as data cleaning, encryption, and identification.

The separation of data plane and control plane, hardware-independent programmability, and line-speed processing capabilities of up to hundreds of gigabytes give the programmable data plane a great inherent advantage in handling network security issues. The programming language P4 [4] is proposed for users to define the data plane functions, to support user-defined protocols and custom packet processing. These new features enable a large number of powerful defense measures, such as access control, topological confuse, heavy hitter detection, DDoS attack mitigation, and Hop-Count Filtering.

The P4-based programmable data plane can mitigate large-scale attacks with low overhead, detect attacks per packet, and respond to ever-changing attacks with hardware speed. However, at the same time, it should be noted that the programmable data plane has limited memory and cannot perform complex calculations like division and exponential or logarithmic calculations. Solving network security problems and achieving functional innovation and performance breakthroughs under the constraints of limited memory and limited operations constitute the research content of network security with P4-based programmable data plane.

The rest of this paper is organized as follows. Based on P4 workflow, the P4 programming language, the P4 architecture, the compiler, and the P4 target are introduced briefly in Section 2. Section 3 describes the development and evolution of the programmability, introduces the application and research directions of P4-based programmable switch in academia and industry, and analyzes its advantages in dealing with network security. Section 4 summarizes the research content of network security based on a P4 programmable data plane. According to the difference between active defense and passive defense, as well as the defense technology adopted, the research papers are divided into four parts: access control, privacy and encryption, availability, and integrated defense. Section 5 compares the performance of different P4 targets on software and hardware platforms. Section 6 summarizes trends and challenges. Section 7 provides the conclusion. Figure 1 depicts the structure of this paper.

Reviews before 2020 usually focus on OpenFlow-based SDN [5], and reviews from 2020 to 2021 [6–8] cover richer P4-related research. However, most of them start from the application scenarios of algorithms and fail to make detailed comparisons and analyses of existing research according to the characteristics of programmable data planes and the way in which network security is handled. For a review of control plane and SDN, refer to [9].

2. Programming Protocol-Independent Packet Processors (P4)

The P4 language is a data plane programming language that can achieve reconfigurability, protocol independence, and target independence. Next, the P4 language and each module of P4 workflow will be introduced, respectively.

2.1. P4 Language.

2.1. P4 Language. The P4 Working Groups [10] have proposed two standards so far: $P4_{14}$ [11] and $P4_{16}$ [12]. $P4_{14}$ has limitations in its use, such as a lack of means of describing various goals and architectures, loose semantics, and insufficient support for program modularization. In response to the shortcomings of $P4_{14}$, the $P4_{16}$ standard introduces strict types, expressions, and nested data structures; deletes declarative structures; and supports multiple different targets and pipeline architectures. The current version of the P4 language compiler has basically implemented most of the features in the P4 language.

$P4_{16}$ is a statically typed language and provides a number of base types, such as void, error, string, match_kind, and bool, as well as type operators that construct derived types, such as header, header_union, enum, struct, extern, parser, control, and package.

For the above-mentioned rich data types, P4 provides different arithmetic operations, such as operations for unsigned integer bit types, fixed-width signed integer, and variable-size bit types, as well as operations on lists, structure types, and so on.

For functions that do not belong to the core of the P4 language, $P4_{16}$ introduces the extern construct. The calling of extern module means that this function needs to be realized based on a specific target or architecture. The P4 pipeline only deals with the packet header, but the payload part of the packet can be dealt with using an extern object.

For a detailed description, refer to the $P4_{16}$ language specification [12], which defines the syntax, semantic rules, and requirements for conformant implementations of the language.

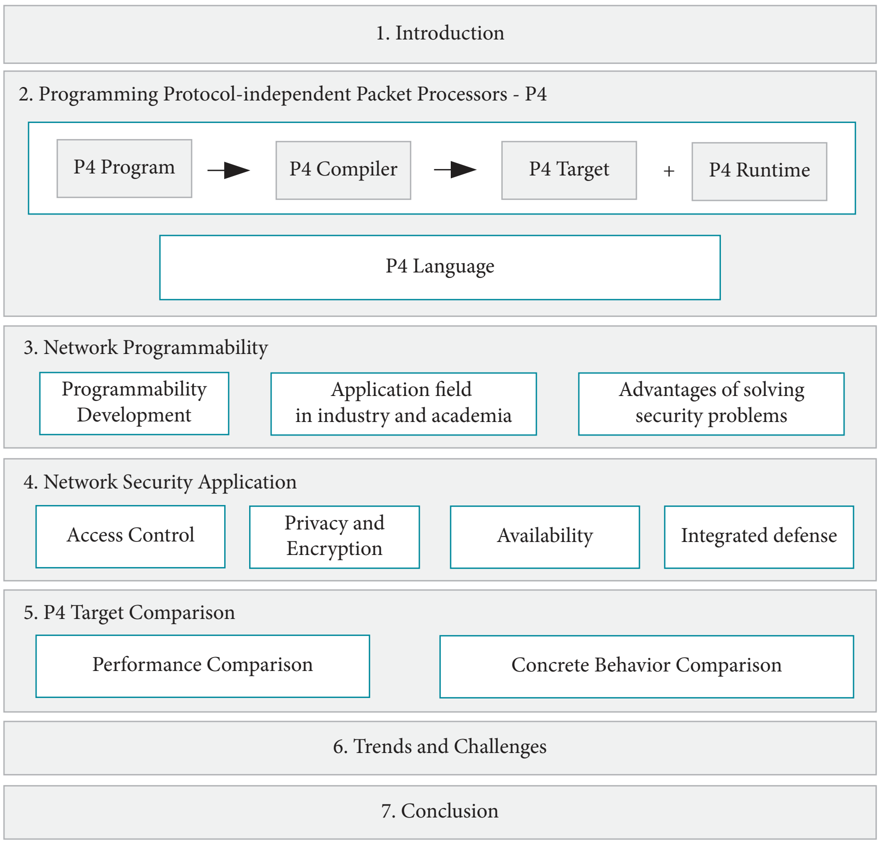
1. Introduction
2. Programming Protocol-independent Packet Processors - P4
P4 Program
P4 Compiler
P4 Target
+
P4 Runtime
P4 Language
3. Network Programmability
Programmability Development
Application field in industry and academia
Advantages of solving security problems
4. Network Security Application
Access Control
Privacy and Encryption
Availability
Integrated defense
5. P4 Target Comparison
Performance Comparison
Concrete Behavior Comparison
6. Trends and Challenges
7. Conclusion

Figure 1: Organization of the paper.

2.2. P4 Workflow. To implement a network application, users can directly program in the P4 language. After compilation, the underlying equipment is configured to fulfill the user's functional requirements. The concepts of P4 program, P4 compiler, P4 Runtime, and P4 target are introduced based on the P4 workflow model, as shown in Figure 2.

2.2.1. P4 Program. The P4 program defines the behavior of data plane and is provided by the user based on a specific architecture. P4 programs for a specific P4 architecture can be deployed to achieve all the goals of the same P4 architecture but cannot be transplanted across different architectures. A typical P4 program has 5 basic components:

Headers: defining the header format of packets

Parsers: defining the finite state machine of the parsing process

Table: defining the matching fields and the corresponding action

Action: defining an action instruction set, including constructing lookup keys, table lookup, and execution of actions

Flow controller: defining control program which determines the flow of data packet processing

2.2.2. P4 Compiler. P4 compiler, which is also provided by P4 target manufacturers, compiles P4 programs into configuration binaries. The work of the P4 compiler is divided into two steps, converting P4 program into target independent intermediate representation (IR) and mapping IR to a specific target.

The main functions of P4 compiler are as follows: (1) generating runtime mapping metadata to allow the control plane and data plane to communicate using P4 Runtime; (2) generating an executable file for the target data plane, specifying the header format and the corresponding operation.

Each P4 compiler is designed for a specific P4 target. For example, FPGA-based (Field-Programmable Gate Arrays) devices mainly use compilers such as Xilinx P4-SDNet and P4FPGA; bmv2 software switches use compilers such as p4c, PISCES [13], and P4-to-OVS. Barefoot Tofino's programmable ASIC uses the Barefoot P4 compiler.

2.2.3. P4 Target. P4 programmable nodes, so-called P4 targets, can be obtained through software (running on the CPU) or dedicated hardware (FPGA, ASIC). They have a packet processing pipeline whose structure is target-specific and is always described by an architecture model. Section 5 compares the performance and research results of different targets.

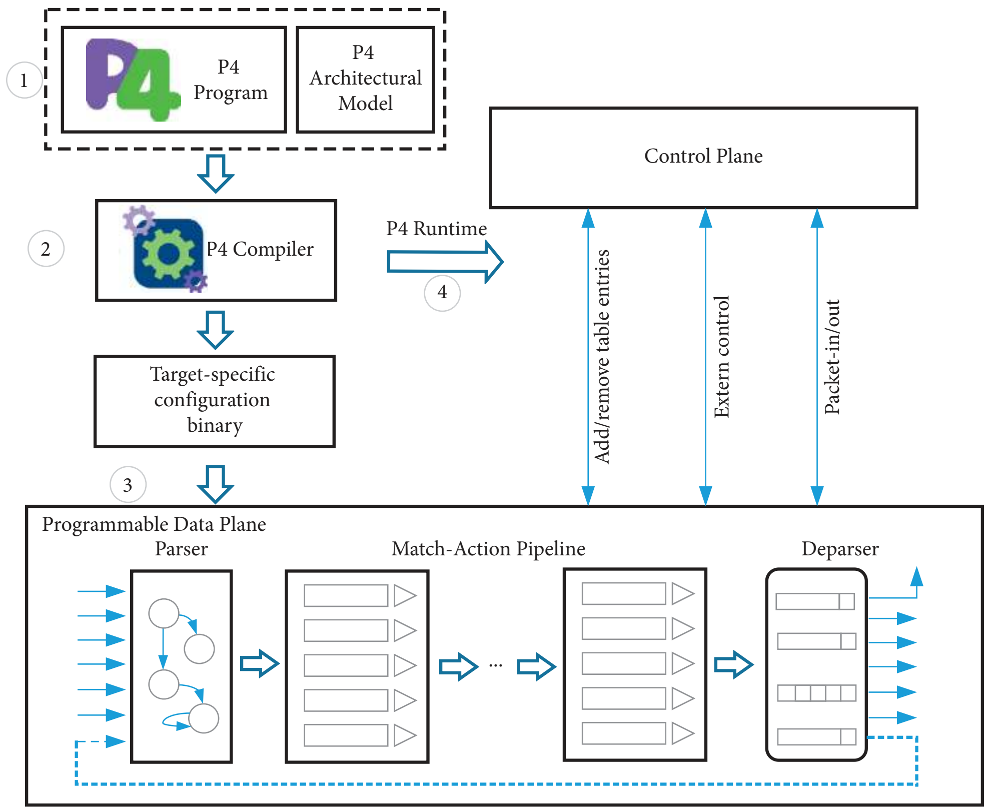

Figure 2: Workflow model based on P4 programmable data plane.

The P4 architecture is used as a programming model and provides the functional view of the P4 processing based on specific target. P4 target manufacturers provide different P4 architectures. For example, NetFPGA-based devices use SimpleSumeSwitch architecture [14], software switches such as bmv2 use V1Model architecture [15], and programmable switches like Barefoot Tofino ASIC (Application-Specific Integrated Circuits) [16] implement the PISA (Protocol-Independent Switching Architecture) architecture.

Take the PISA architecture as an example to briefly describe the data processing. The pipeline of the P4 program contains many control blocks, and the input packet is processed and transmitted to the next control block in the pipeline.

Parser. It is a control block which specifies how to parse the header of the packet. The parser is represented by a finite state machine, where the state is responsible for extracting a single header structure.

Match-Action Pipeline. It contains multiple match-action tables (MAT), each of which stores keywords and corresponding action data. These tables match packets according to different header fields and determine the operations to be performed if a match occurs. The ingress pipeline modifies packet and determines its output port. The egress pipe performs other necessary modifications. The control plane installs entries in the MAT to control the order of execution. Metadata is proposed to support stateful functions.

Deparser. Deparser composes the modified headers into packets in a regular order and sends them to the next step, such as forwarding to the output port, looping to the input port, or sending to the CPU.

2.2.4. P4 Runtime. P4 Runtime [17] is an Application Programming Interface (API) which is used to connect the control plane and data plane. For the control plane, P4 Runtime shields the hardware details of the data plane and is independent of the features and protocol the data plane supports. Devices based on different targets can be controlled by the same API.

3. Network Programmability

Programmability is defined as the capability of hardware and software to execute algorithms specified by users. Furthermore, network programmability refers to the ability to process algorithms executed in the network, especially the ability to execute in a single processing node. It means that the functions of the control plane and data plane can be defined and modified; both network equipment manufacturers and users can build networks that suit their needs and performance requirements.

3.1. Programmability Development. The functions of traditional network equipment (such as routers or switches) are defined by the manufacturers, and the protocols and algorithms are built-in, making users unable to modify them. Adding new functions requires equipment suppliers to design, manufacture, test, and deploy dedicated hardware components [18]. This process is cumbersome and expensive and usually takes 2-3 years or even longer. In addition, the driving force for manufacturers to develop and upgrade functions comes from the potential needs of a wide range of users for functions. Thus, innovation is usually lagging behind, which severely restricts the application and development of new technologies.

SDN has greatly improved the problem of rigid network protocol. With the separation of the data plane and the control plane, the centralized controller conducts management, status monitoring, and forwarding rules-making through the interface to data plane, such as OpenFlow [3]. Despite its great advantages, SDN architecture suffers from scalability and performance issues due to insufficient flexibility in stateful packet processing in SDN switches which support OpenFlow. The data plane relies on the rules installed by the controller to forward or drop network traffic, and it introduces the communication overhead of the control plane and data plane. In addition, the controller might become the target of network attacks. Although OpenFlow 1.5 supports up to 44 matching fields, it lacks the flexibility to support any new fields.

In an SDN switch with a programmable data plane, users can program for other functions and details of packet processing by using P4, in addition to the basic functions of a switch. Figure 3 shows the evolution of network programmability [6]. Hardware can be programmed without an understanding of the underlying hardware details, because P4 is applicable to various devices, such as ASIC, NPU (Network Processing Unit), and FPGA.

3.2. Application Fields of P4 in Industry and Academia.

Since 2016, P4-based programmable data planes have attracted great interest among academics [6–8], and the research mainly focuses on the following fields.

3.2.1. Traffic Measurement and Traffic Engineering.

With the emergence of programmable switches, fine-grained measurements can be performed at line rate, so functions such as congestion detection [19], active queue management [20], and load balancing [21] can be realized to improve the quality of service.

3.2.2. The Routing and Forwarding Function.

New routing strategies and forwarding behaviors, which are different from those of traditional routers and switches, can be realized based on P4-based programmable switches. A lot of research can be conducted further, such as L4 load balancing, source routing [22], and named data networking [23].

3.2.3. Advanced Network Support.

P4-based programmable switches are more sensitive to newly appear application scenarios than traditional equipment and can meet requirements of different types of networks, such as cellular networks [24], IoT [25], and time-sensitive networks [26].

3.2.4. Network Security.

As programmable switches can check and modify packet headers according to user customization, network security defense solutions, such as intrusion detection, encryption, DDoS attack mitigation, and topology scrambling, can be updated and upgraded.

3.2.5. Network Accelerated Computing.

Benefiting from the data processing capabilities of the programmable data plane, machine learning [27, 28], deep detection, and other functions which can accelerate the training/reasoning process can be realized.

Considering factors such as operability, resource occupation, multitenant sharing support, and compatibility with existing equipment, the scenarios in which P4-based programmable switches are used in the industry are much fewer than those in academia, and they are mainly concentrated in the Top-of-Rack switch of data center [29], INT (In-band Network Telemetry) network measurement [30], wireless backhaul equipment [31], equipment white-boxing [32, 33], and other fields.

3.3. Advantages of P4-Based Programmable Data Plane.

The processing of network attacks usually includes three steps: detection, classification, and mitigation.

(i) Identify the relevant fields of the packet header and make statistics on the features of the traffic. This step usually requires "seeing" as much information as possible.

(ii) Identify potential threats based on the feature's value. The step usually requires some parameter values as the basis for judgment, such as the resource usage of current equipment, the thresholds set in advance, and the parameters obtained through machine learning.

(iii) Mitigate attacks. Measures include dropping the packet, load balancing, limiting speed, or just warning other devices by sending signals to the controller.

Compared with OpenFlow, the programmable data plane has three key advantages when dealing with network attacks: visibility per packet, scalability, and high-speed processing capability.

(1) The visibility of each packet means that attack detection algorithms can be developed in the switch hardware and applied to each individual packet instead of sampling or aggregate tracking on the software controller.

As shown in Figure 4, for traditional security applications, defense measures that can be implemented at the hardware level include ACL and packet filtering, while higher-level firewalls and deep packet inspection strategies need to be implemented

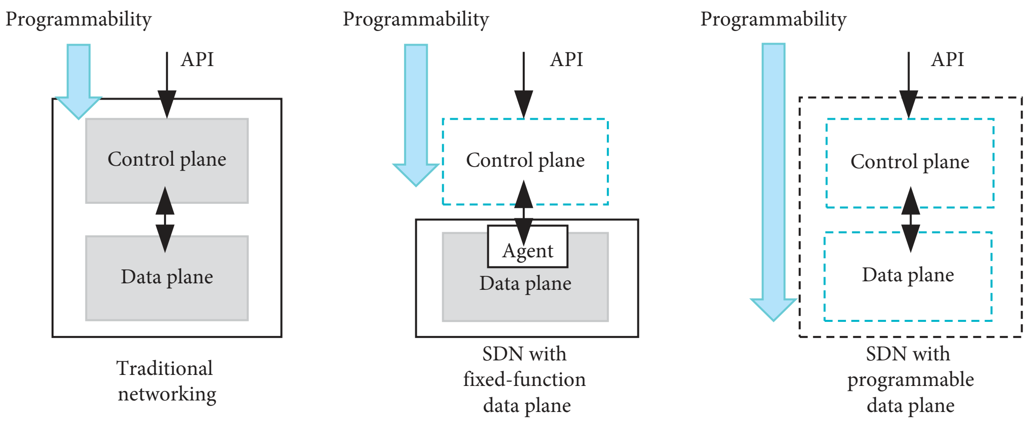

FIGURE 3: Traditional network, a fixed-function data plane SDN, and a programmable data plane SDN.

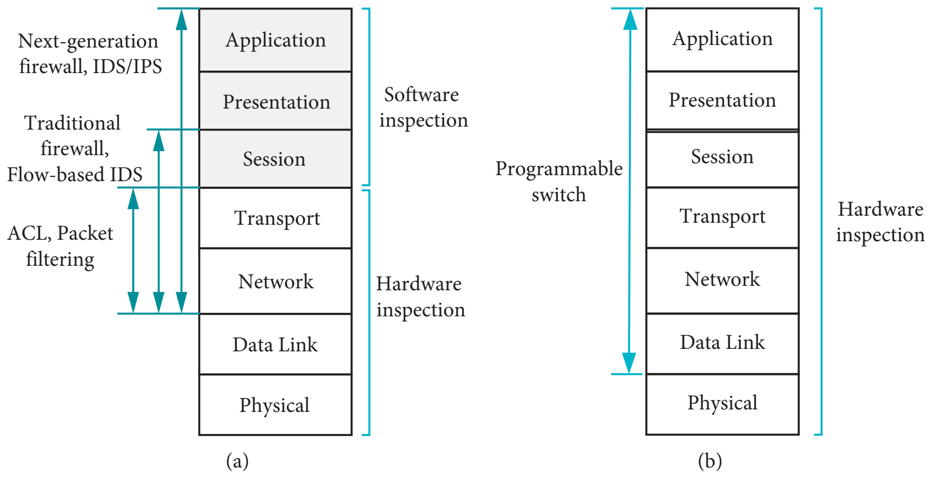

FIGURE 4: The OSI model for security applications: (a) conventional devices; (b) programmable devices.

by software [8]. This is also an important reason why traditional switches must upload sampled packets and hand them over to the controller. The programmable switch can process the protocol fields of each layer at the hardware level, meet the requirements of deep packet inspection and protocol field modification, and support custom protocols and complex defense strategies.

(2) Scalability means that since the defense is located directly in the switch, it will expand with the network scale and speed, making the centralized controller no longer a bottleneck.

In the context of security issues, the network-wide knowledge provided by the controller will not bring additional benefits for tasks which only need to complete the local state. Conversely, the controller's explicit participation in each stateful process will cause additional control signals and overloads and may cause the controller itself to malfunction.P4 provides state memory counters, meters, and registers, which are used to maintain state among multiple packets of the stream. Most of the defense measures are completed by the data plane, and the controller only receives and maintains a small number of statistical results, distributes flow tables, etc., which can reduce the complexity of the network and ensure the scalability of the defense scheme.

(3) The high-speed processing capability means that a lot of tasks can be done at line rate.

The programmable data plane can execute local policies, process packets at line rate, respond quickly to attack behaviors, and meet the real-time requirements of applications. Once an attack is detected, measures can be taken immediately to mitigate the attack of the switch without causing round-trip time delay to the remote controller.

Besides, applications which require complex processing can be executed at a low cost and in a distributed way on the data plane. Functions originally implemented by software can be offloaded to the data plane, so that the data plane transmits the results of execution to the controller instead of raw data. This can greatly reduce the processing burden of the controller and speed up the execution [34, 35]. This feature is especially applicable when solving network security issues. As the judgment of attack behavior usually requires collecting information of a large number of packets for analysis, this process consumes a large number of computing resources of the controller and increases communication overhead.

4. Network Security Application

By benefiting from the processing capabilities of the P4 programmable data plane, packet-by-packet inspection and filtering technology can be implemented. Therefore, aiming at network security, programmable switches can realize a variety of functions, including access control for incoming traffic, encryption performed on the data plane to protect privacy, as well as responding to attacks for the availability of the network, and upgrading the multilevel defense strategies to prepare for a rainy day. Figure 5 illustrates the research of network security based on a P4 programmable data plane.

4.1. Access Control.

4.1. Access Control. The restricting of access to objects or digital resources is called access control. Due to the advantages of programmable switches, authentication and authorization can be delegated to the data plane. Filtering untrusted traffic in advance can prevent the server from getting into the predicament of processing a large amount of intrusion information. Therefore, the server's computing resources can be used to process business applications. In Table 1, the careful contrast of the research shows some important differences.

4.1.1. Firewall. In [36], the authors implement a firewall in P4 for the first time. The parser can identify Ethernet, IPv4/IPv6, UDP, and TCP headers. There are two MATs; one MAT implements whitelist filtering, and the other MAT is a banned list, also called a blacklist. Packets matching the MAC/IP address of the network host in the blacklist will be directly discarded.

CoFilter [37] calculates the flow identifier by directly applying a hash function to the 5-tuple of each packet and uses exact matching table entries to calculate the flow identifier. The conflicting streams are mapped to the new index. Hash collision calculations are implemented on the server, requiring considerable memory and computing power. A copy of the packet at the beginning or the end of the flow is sent to the server for further processing, while other packets of the flow are processed just in the switch pipeline.

Because traditional firewalls lack support for functions such as topology creation of virtual network (VN), virtual machine migration, or personalized security requirements, Deng et al. [38] propose a software-based virtual firewall VNGuard. On the basis of SDN and NFV, VNGuard defines an advanced firewall policy language, which can manage and adjust virtual firewall settings according to virtual network requirements. P4Guard [39] replaces the software-based firewall in the VNGuard system with a P4-based configurable virtual firewall. The controller of P4Guard collects traffic statistics from switches and monitors network traffic. The data plane deals with packets according to firewall rules. The P4-based firewall includes two MATs. One MAT is used to control enabling/disabling the firewall at runtime, and the other MAT allows the Allow/Drop (metadata) of the 3/4 layer header field to be used as a matching key for outgoing port processing.

Ricart-Sanchez et al. [40] propose a full-featured 5G multitenant firewall based on FPGA to handle the GTP (GPRS tunneling protocol) data transmitted between the edge and the core network. Using the P4 programmable data plane to parse and match the GTP header fields, the TCAM table adds two parameters to the traditional quintuple, providing the firewall with the possibility of identifying different tenants and end users. The VNI (VXLAN Network Identifier) in the VXLAN header defines the tenant, and the GTP ID identifies the end user, as shown in Figure 6.

4.1.2. Port Knocking. Port knocking is a simple and ingenious authentication scheme used to open network ports. Since the implementation of the port knocking function does not require a controller to maintain the entire network information and has the possibility of local execution, Almaini et al. [41] propose a scheme to delegate the authentication to the data plane. Only when the host can send the correct port sequence will it be granted to start the connection. After a specific predefined period of time, the permissions will become invalid and the client needs to perform authentication again. Finite state machine (FSM), which contains a limited number of states, is adopted as the authentication unit in P4 switch. When a correct port number is received, FSM can jump from one state to the next state; if an incorrect sequence number is received, it rolls back to the initial state. The status data is recorded by metadata during the execution of P4. How the knocking sequence can be safely distributed to trusted nodes is not discussed in this article, and the unencrypted port knockout is easily affected by packet sniffing.

According to the different degrees of offloading the port knocking function, P4Knocking [42] is proposed with four implementation schemes. The first is full data plane offloading. The second and third are hybrid data and control plane offloading. The fourth implementation relies only on the controller. When it comes to performance, complete data plane offloading may be the best choice. Any operations related to port knocking perform at line rate. The control plane only installs table rules related to the knocking sequence and does not involve the master port knocking verification operation. Therefore, even if the connection to the control plane is interrupted, the data plane can still authenticate the knocking sequence.

Taking advantage of P4, two authentication schemes [43] are proposed to ensure that only legitimate devices can access the server. The two schemes are port knocking and port scanning mitigation. Experimental results show that the proposed two schemes improve the overall availability of the network, by offloading tasks of the controller and reducing the communication overhead between the data plane and the control plane, and will not have a significant negative impact on the performance of the switch.

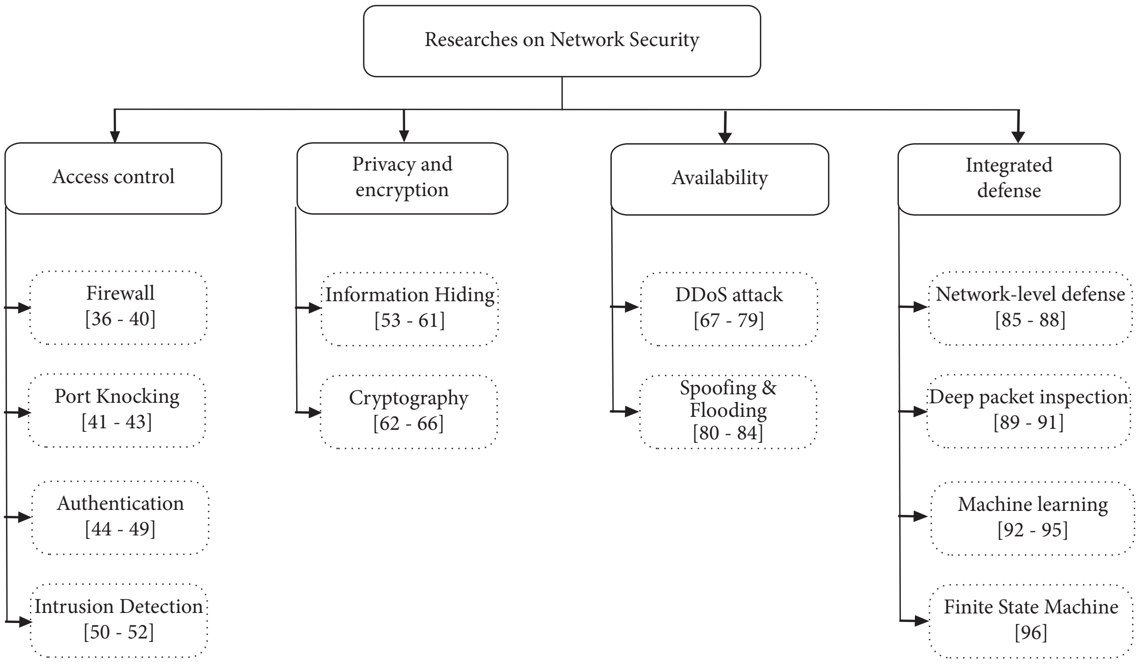
Researches on Network Security
Access control
Privacy and encryption
Availability
Integrated defense
Firewall [36 - 40]
Port Knocking [41 - 43]
Authentication [44 - 49]
Intrusion Detection [50 - 52]
Information Hiding [53 - 61]
Cryptography [62 - 66]
DDoS attack [67 - 79]
Spoofing & Flooding [80 - 84]
Network-level defense [85 - 88]
Deep packet inspection [89 - 91]
Machine learning [92 - 95]
Finite State Machine [96]

Figure 5: Research map of network security based on P4 programmable data plane.

Table 1: Comparison of network security research related to access control.

Ref.	Year	Platform	Scenario	Core idea	Limitation
[36]	2016	p4c-behavioral	Firewall	One MAT is a whitelist; the other MAT is a forbidden list. Packets that match the forbidden list will be directly discarded	No support for variable-length header field
CoFilter [37]	2018	Tofino	Flow table problem	A 5-tuple is used for hash calculation, and an exact match is used to map the conflicting stream to a new index	Solving hash conflicts and hash table optimization issues by using external servers
P4Guard [39]	2018	bmv2	Virtual network	A MAT is used to enable/disable the firewall. Another MAT uses Allow/ Drop (metadata) in the header field of the 3/4 layer as the matching key for outgoing port processing	P4Guard can only run on the transport layer
[40]	2018	NetFPGA SUME	5G	The TCAM table adds two parameters, VNI and GTP ID, on the basis of the traditional five-tuple to identify different tenants and end users	The default policy is to allowance
[41]	2019	bmv2	Task offloading	FSM is used as the authentication unit in P4 switch	How the knocking sequence can be safely distributed to trusted nodes is not discussed. Unencrypted port knockouts are susceptible to packet sniffing
[42]	2020	bmv2	Task offloading	According to the different degrees of offloading the port knocking function to the data plane, the discussion is divided into four situations	How the knocking sequence can be safely distributed to trusted nodes is not discussed. Unencrypted port knockouts are susceptible to packet sniffing
[43]	2021	bmv2	Task offloading	Offload port knocking and port scanning mitigation to the data plane	How the knocking sequence can be safely distributed to trusted nodes is not discussed. Scrolling back is not possible when the duration between knockings exceeds a certain threshold

Table 1: Continued.

Ref.	Year	Platform	Scenario	Core idea	Limitation
Poise [44]	2020	Tofino	BYOD	The client adds the context information header to the packet. The switch implements the ACL based on the information	Need to change the client. Lack of authentication, integrity, and confidentiality for context
P40f [45]	2019	bmv2	Passive fingerprinting	The information identifying the signature field is stored in metadata; the match-action table is searched using metadata as a key to obtain the outgoing port processing	The lack of complex strategies involving rate limiting traffic
E-Replacement [46]	2021	bmv2	IoT port scan	Using the concept of Hashpipe, E-Replacement mitigates the performance degradation caused by hash conflicts	No support for protocols other than TCP (such as UDP)
[47]	2020	bmv2, Netronome, NetFPGA SUME	SYN flood attack	SYN proxy scheme is implemented based on P4 through SYN cookie and SYN authentication	Performance depends on the availability of suitable cryptographic hash functions in the programmable data plane
[48]	2020	NetFPGA SUME	IP address spoofing	Six known antispoofing mechanisms are implemented based on P4, and their performance is compared	The control plane related functions are unclear
P4DAD [49]	2020	bmv2	IPv6 Neighbor Discovery Protocol	The register maintains binding entries between IPv6 addresses, port numbers, and address status. Packets that do not comply with the binding relationship are usually spoofing attacks	The address information needs to be maintained locally; it is not verified on a resource-constrained hardware platform
[50]	2018	bmv2	ICS	Two-level IDS: the first level of P4 switch analyzes and filters packets based on the whitelist; the second level further processes suspicious data	Lack of support for other ICS protocols and ability to detect more complex attacks
P4ID [51]	2019	bmv2	Intrusion detection	The first *n* packets of a flow are sent to IDS for analysis, and the rules are formed and installed on the data plane. The subsequent packets are processed according to MAT	No rate limiting on IDS traffic
[52]	2021	bmv2	Anomalies detection	Tracking packet execution path, maintaining path statistics, and measuring deviation using the chi-square test method	Implementation only in the data plane of the software

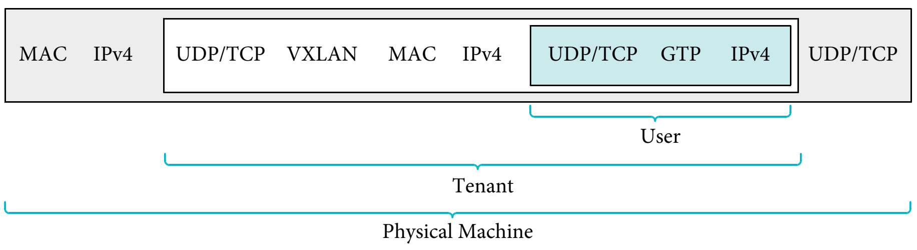

Figure 6: Packet structure used in edge-core 5G infrastructure.

4.1.3. Authentication. In enterprise and campus networks, there are multiple bring-your-own-devices (BYOD), such as mobile phones and laptops. Based on P4 devices, Poise [44] introduces context-aware strategies to provide protection mechanism in the BYOD scenarios. The BYOD client adds the context information to the packet header. Then, the devices that run the Poise program parse the information from the packet header and implement ACL based on the runtime context of the device. Compared with OpenFlow-based defense solutions, Poise has appropriate overhead and can flexibly respond to saturation attacks on the control plane. However, the system may be subject to attacks of interception and modification packet due to the lack of checks for the context packet.

Existing software-based passive fingerprinting tools have low availability and cannot keep up with the traffic in high-speed networks. Bai et al. [45] propose a fingerprint recognition system P40f, which is implemented based on a programmable data plane and can execute security policies of permission, drop, and redirection at line rate. P40f collects information about the p0f signature field in the parser and the match-action pipelines and stores the information in the metadata. The metadata is then used as a key in the lookup table in the last stage of the pipeline. If it hits one rule, the packet will be dealt with actions associated with the matching rule. Finally, P40f decomposes the packet header and forwards the packet to the next destination. P40f lacks a complex strategy involving rate limiting traffic.

Existing detection schemes for SDN port scan do not consider hash conflicts, resulting in reduced detection performance. Cai et al. [46] propose a new scanner data collection method, E-Replacement, which can reduce the performance degradation caused by hash conflict. In the E-Replacement scheme, the switch records the characteristics of scanning behavior and retains only suspicious data to reduce the storage pressure. The controller periodically extracts statistical information from switches and adopts an unsupervised machine learning algorithm to classify the data to eliminate the data labeling process.

In order to prevent SYN flooding DDoS attacks, Scholz et al. [47] propose a SYN proxy scheme based on P4 through SYN cookie or SYN authentication. The SYN cookies scheme modifies the relevant fields of each packet, which is completely transparent to the TCP client, while SYN authentication only modifies the corresponding fields during the handshake process, which is relatively easy to implement. The SYN proxy modifies the received packet according to the strategy used, the TCP flag in the packet, and the state reserved by the agent. The status whitelist and serial number changes are reserved in the form of a match-action table.

Based on the NetFPGA SUME P4 platform, six known antispoofing mechanisms against IP address spoofing are implemented in [48]. The author compared the resource usage of these antispoofing mechanisms on the FPGA. These mechanisms achieved a processing delay of approximately 2 μs per packet and a throughput of approximately 8.5 Gbit/s.

P4-based Duplicate Address Detection (P4DAD) [49] introduces a mechanism to filter fraudulent NDP (Neighbor Discovery Protocol) messages as a simple alternative to authentication or encryption. The registers of P4 switch maintain the binding entries between IPv6 addresses, port numbers, and address status. Packets belonging to message spoofing attacks usually do not comply with the above binding relationship. Therefore, P4DAD can detect and discard malicious NDP messages without modifying the NDP or host protocol.

4.1.4. Intrusion Detection.

Ndonda et al. [50] propose a two-level IDS (Intrusion Detection System) for Industrial Control System (ICS) networks. In the first level, P4 switches parse and process packets according to whitelist. Suspicious packets will be forwarded to the devices of the second level for deeper inspection instead of being directly rejected. In the second level, the dedicated host determines whether the packet is malicious and notifies the controller to update the filter on the switch to block or accept the connection.

Traditional Intrusion Detection System (IDS) device, which is at the entrance of the network, processes copies of network traffic. The amount of traffic that needs to be processed increases in proportion to the expansion of the network. P4ID (P4 Enhanced Intrusion Detection) [51] reduces the load of the IDS by applying prefiltering on the P4 switch. In the stateful phase, the first n packets of each new flow will be sent to IDS for analysis, and corresponding forwarding rules will be formed and installed on the data plane. In the stateless phase, packets are filtered according MATs.

Sanghi et al. [52] propose a scheme which performs statistical analysis on packet execution paths in the data plane to detect anomalies. As a packet might take any path—the tables applied and the actions executed—in a P4 program, the Ball–Larus encoding technique is used to track paths and construct the expected and observed distributions. Further, the switch uses a hash table to maintain per-path statistics and collects packet execution path distribution of real traffic over a period of time. The traffic distribution is compared with an expected one to find abnormal mode.

4.2. Privacy and Encryption.

Information about the user's identity and online operations can be mined from the sent packets. Extensive research has been conducted on user privacy and anonymity in the past. Methods, including scrambling, replacement, and encryption, are adopted to hide information or increase the complexity of cracking. However, the existing solutions have several limitations. Firstly, some solutions need to modify the entire Internet architecture, which is extremely impossible. Secondly, most of the solutions are performed based on software and cannot cope with the high-speed traffic. Recently, research gradually turns to using programmable switches to develop lightweight anonymity and encryption systems. Table 2 summarizes the research about privacy and encryption.

4.2.1. Information Hiding.

NetHide [53] encrypts the network topology to mitigate topology-centric attacks. The solution describes the network confusion problem as a multiobjective optimization problem; at the same time, the accuracy and utility are used as metrics. ILP (Integer Linear Programming) solver and heuristic methods are adopted to obtain relevant parameters.

SPINE [54] (Surveillance Protection in the Network Elements) obfuscates IP address to handle eavesdropping from intermediate networks. Different from the software-based methods, SPINE runs on the data plane without requiring the cooperation between hosts and switches. Figure 7 shows the SPINE architecture. Before the packet enters the untrusted entity, the trusted programmable switch encrypts the IP address, TCP sequence number, and ACK number. The encrypted data are inserted into the second half

Table 2: Privacy and encryption applications.

Ref	Year	Core idea	Security goal	Platform	Limitation
NetHide [52]	2017	Encrypting network topology	Confidentiality	N/A	The static measurement value of the flow is required before encrypting the topology
SPINE [54]	2019	Encrypting IP source address, TCP sequence, and ACK number	Confidentiality	bmv2	Key rotation is maintained by the central controller
[55]	2019	Encrypting IP source address	Confidentiality	bmv2	High cost of dynamic key update mechanism
[56]	2020	Differentiated encryption	Confidentiality	bmv2	No performance comparison.
MIMIQ [57]	2020	Implementing fine-grained and flexible IP address hiding encryption	Confidentiality	bmv2	An address migration process has a great impact on the throughput performance
LANIM [58]	2020	Three lines of Defense: different link selection, encryption protocol parameters, encryption load	Confidentiality	bmv2	Few experimental data; serious disorder problem; a waste of bandwidth resources
P4NIS [59]	2021	Three lines of Defense: different link selection, encryption protocol parameters, encryption load	Confidentiality	bmv2	Serious disorder problem; a waste of bandwidth resources
AMS [60]	2021	Reducing disorder in multipath scenarios	Confidentiality	bmv2	Easing rather than eliminating disorder
PANEL [61]	2019	1) Source address rewriting 2) Randomized IP ID and TCP sequence 3) Randomized TTL	Confidentiality	Tofino	P4 switch needs to store the original session information
[62]	2020	Implementing AES encryption by looping in the data plane	Confidentiality	Tofino	Encryption rounds are inversely proportional to throughput
P4-IPsec [63]	2020	Encapsulated safety payload (ESP) in tunnel mode	Confidentiality, integrity	bmv2, NetFPGA SUME, Tofino	The control plane is in charge of the key exchange protocol; encryption and decryption rely on user-defined P4 extern
P4-MACsec [64]	2020	Implementation of MACsec on P4 data plane	Confidentiality, integrity	bmv2, NetFPGA SUME	Encryption and decryption rely on user-defined P4 extern
ONTAS [65]	2019	Online network traffic anonymization system	Confidentiality	Tofino	Using the built-in outdated hash algorithm CRC-32 which is designed to calculate checksum rather than an encrypted hash algorithm
[66]	2020	Adding password label and sample to check the integrity	Integrity	bmv2	The terminal needs to fill in the password identification; the controller verifies the integrity of the sampling packet

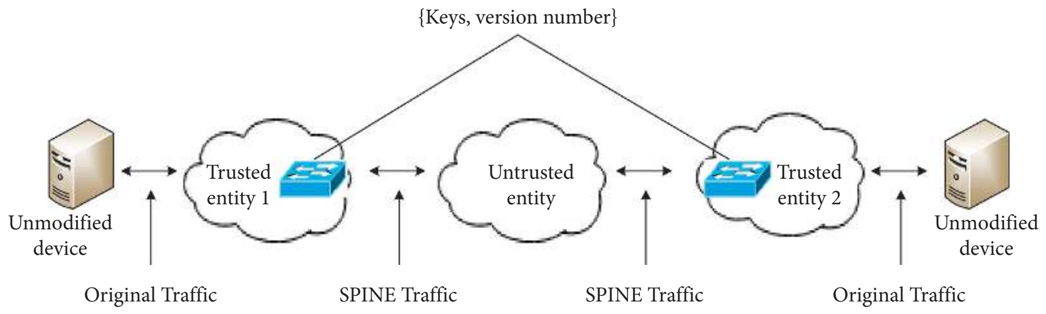

Figure 7: The SPINE architecture.

of the IPv6 destination address. Untrusted entity can see only encrypted address in the header. After the data enters the trusted entity, the receiving switch can use the version number to select an appropriate key to decrypt and restore the IPv4 header.

As an active defense mechanism, Chang et al. [55] encrypt the IP source address to realize the unlinkability of the IP address. Except for the edge P4 switch connected to the host at the sending and receiving end, other devices in the network are considered to be composed of untrusted switches/routers. The sender encrypts source IP address through an XOR operation. The P4 switch decrypts the sender's real IP address. A dynamic key update scheme is performed to ensure the randomness of the address conversion, which increases the difficulty of cracking for the attacker.

Traditional encryption strategies cannot dynamically encrypt packets based on their importance. Qin et al. [56] propose a scheme of updating security policy according to the different traffic distribution and types in the network. When the first packet of each new flow reaches the P4 switch, the P4 switch sends a request to the controller. Once the ONOS (open network operating system)-based controller receives the packet, the controller will decide whether the protection policy should be enabled according to a security policy. The controller stores the necessary data in register and updates MATs of the data plane to enable TCP sequence number replacement.

In order to further strengthen the security of IP address encryption, Govil et al. [57] propose a fine-grained and flexible IP address mixing scheme MIMIQ (Masking IPs with Migration in QUIC), which can randomize IP addresses in the flow, even every few packets. An address allocation server distributes IP addresses to clients and installs forwarding table entries to its switches. Except for the distribution of servers and a few key switches in the trusted network, MIMIQ will not modify the existing network and server infrastructure.

Liu et al. [58] propose a learning-based adaptive network immune mechanism (LANIM) based on machine learning, with three lines of defense schemes configured to prevent eavesdropping attacks. The first line of defense uses intelligent controllers to perceive the network status and make decisions based on the minimum risk machine learning algorithm for abnormal network conditions. The second line of defense uses stateful encryption strategies. The third line of defense adopts the existing encryption algorithm based on computational complexity.

P4NIS (P4-based Network Immune Scheme) [59] is also proposed with a three-line defense solution to resist eavesdropping. Firstly, packets belonging to the same flow will be transmitted through different links. The source/destination port and serial/confirmation number are replaced in the next line of defence. In the third line, the existing encryption mechanism is adopted to encrypt the payload of packets. The P4NIS architecture is shown in Figure 8.

Forwarding packets through different paths will cause extremely serious out-of-sequence problems. Besides, the unreasonable multipath defense actually brings waste of bandwidth resources. Zhou et al. [60] propose an Adaptive Multipath Scheduling (AMS) mechanism, which increases the difficulty of eavesdropping and improves the usage of the network bandwidth. AMS selects three network paths with similar characteristics and distributes packets among them. Experimental results show that compared with competitors, AMS increases the transmission throughput by 74% and reduces the out-of-sequence rate by 48%.

Moghaddam et al. [61] propose a lightweight and low-overhead network anonymity solution, PANEL (Practical Anonymity at the NEtwork Level), to overcome the problems the popular anonymity system has without changing the Internet routing and forwarding protocol. To hide information carried by packets, PANEL provides solutions like rewriting the source address, randomization of IP identification and TCP sequence, and randomization of TTL value. As the replacement of address information may cause the breakdown of the routing response, P4-based programmable devices need to store the original session information, maintain the forwarding table fwd_panel and the reverse table rev_panel. Check whether the destination IP address of the packet is stored and maintained locally. If it is, enter the reverse table to search for an address replacement; if not, search for the sending port through the forwarding table. Figure 9 shows the P4 program of the data plane ingress processing flow.

4.2.2. Cryptography. Since the current commercial programmable switch does not equip with a specific encryption processor and only supports simple arithmetic operations and limited lookup actions, it is not easy to implement AES (Advanced Encryption Standard). Chen et al. [62] propose a scrambled lookup table technology to reduce the operations required for AES encryption, and implement AES on a programmable switch based on Barefoot Tofino. The authors find that a single pipeline can execute two AES rounds. Therefore, the system utilizes a packet recycling technique by reinjecting packets into the pipeline. In this way, the AES-128 algorithm can be completed by using five pipe passes.

P4-IPsec [63] implements IPsec based on P4 switches. Compared with standard IPsec, the P4-IPsec tunnel is established and managed by a controller based on a predefined tunnel configuration file. The P4 switch replaces the Security Policy Database (SPD) and the Security Association Database (SAD) with a matching operation table. P4 extern implements the cipher suite, which means that the CPU of the switch performs encryption and decryption calculations with increased latency and reduced throughput. In addition, because P4-IPsec only implements in tunnel mode, the host needs to introduce additional Linux client agent tools.

P4-MACsec [64] implements MACsec based on P4 switch. MACsec uses cryptographic integrity checking or symmetric encryption to perform layer 2 protection. Its deployment requires knowledge of network topology information and undergoes a time-consuming and complicated initial setup. However, the Link Layer Discovery Protocol (LLDP) supported by traditional network switches has difficulty detecting topology changes in time. For link discovery and monitoring, P4-MACsec improves LLDP through encrypting payloads and sequence numbers. MACsec is implemented directly on the P4 data plane; the encryption/decryption operations are stored in the P4 processing pipeline in the form of P4 externs.

Kim et al. [65] propose an online network traffic anonymization system (ONTAS), which converts the anonymization strategy of network operators in a specific program based on P4. The program synthesizes the matching operation table in the data plane and anonymizes the incoming packet stream at line rate. ONTAS is implemented on a PISA structured switch, which is flexible and scalable, and supports a policy language used to express anonymous tasks.

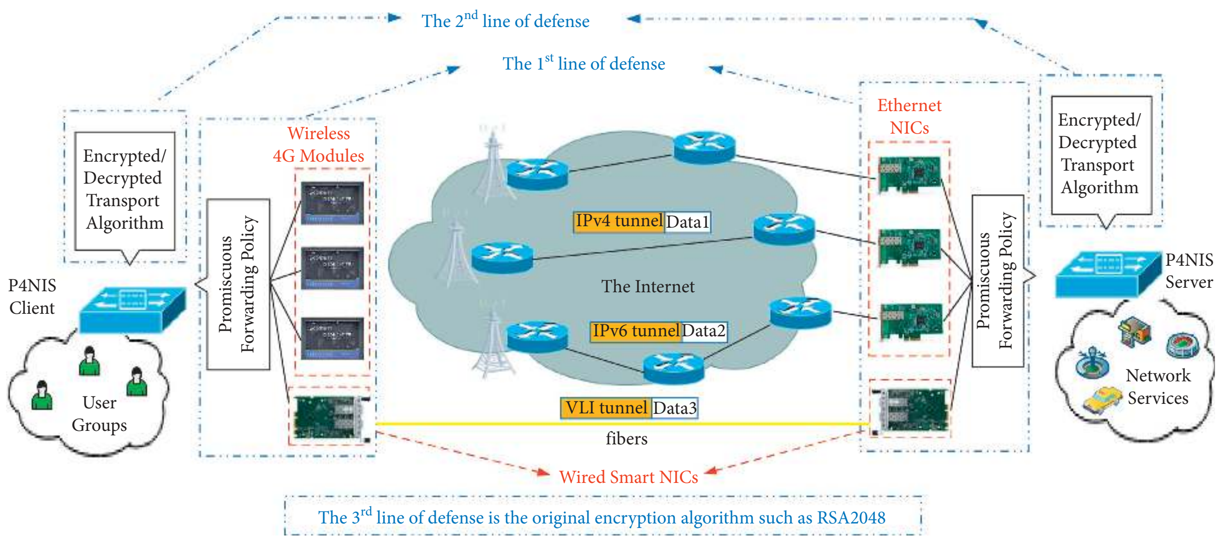
The 2nd line of defense
The 1st line of defense
Encrypted/ Decrypted Transport Algorithm
Wireless 4G Modules
Ethernet NICs
Encrypted/ Decrypted Transport Algorithm
P4NIS Client
Promiscuous Forwarding Policy
IPv4 tunnel Data1
The Internet
IPv6 tunnel Data2
VLI tunnel Data3
fibers
Promiscuous Forwarding Policy
P4NIS Server
User Groups
Network Services
Wired Smart NICs
The 3rd line of defense is the original encryption algorithm such as RSA2048

FIGURE 8: The P4NIS architecture.

```
control ingress {
    //Check if packet is a valid IPv4 packet
    if (valid(ipv4)) {
        //Check if TTL is larger than 0
        if (ipv4.ttl > 0) {
            //Check if dest IPv4 address belongs to us
            apply(match_panel_ip)
            {
                hit {
                //Apply reverse tables
                    if(valid(tcp)) {
                        apply(rev_panel_tcp);
                    }
                    if(valid(udp)) {
                        apply(rev_panel_udp);
                    }
                }
            }
            //If packet's IPv4 address does not belong to
            //us, apply forward tables
            if(panel_metadata.match_panel == 0) {
                //Assume all sessions are matched
                if(valid(tcp)) {
                    apply(fwd_panel_tcp);
                }
                if(valid(udp)) {
                    apply(fwd_panel_udp);
                }
            }
        ...
}
```

FIGURE 9: P4 program of the data plane ingress processing flow in the PANEL scheme.

Zuo et al. [66] propose a packet verification mechanism based on a programmable data plane. As shown in Figure 10, the sender adds a custom password identifier to the packet, which is stored in the Options field of the IP header. A P4 switch is added to the OpenFlow-based SDN. The P4 switch parses the packet password identification to achieve precise control of packet forwarding based on proportionality sampling. The controller verifies the integrity of the sampled packet and installs flow rules for abnormal packet to the OpenFlow switches. The proposed scheme can customize the matching field, and the forwarding delay is 2.5% of the comparison scheme. However, the algorithm can only detect forged and tampered messages and cannot identify the

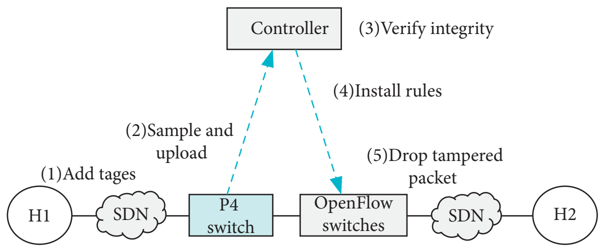

FIGURE 10: A P4 switch in the traditional SDN.

device location where the tampering or forgery occurs. In addition to adding a P4 switch for sampling, the host needs to be modified to add security labels.

4.3. *Availability.* The attacker sends large numbers of packets to exhaust the victim's bandwidth or computing resources. It usually appears in the form of DDoS attacks, flooding or spoofing, and the common types are as follows:

(1) Slow DDoS not only utilizes legitimate TCP behavior, but also sends packets with a strength similar to that of normal traffic, making it impossible to identify the victim through traditional methods
(2) Volumetric attacks are mainly related to the purpose of flooding bandwidth and flooding targets by attacking traffic using UDP or Internet Control Message Protocol (ICMP)
(3) Malicious hosts may add an amplifying method to their attacks and send requests which result in the response traffic reaching a level far exceeding the sending traffic, even casuing the DDoS rate to be higher than Tbps, that is, a Reflection Amplification (RA) attack
(4) In addition, since no real session is required to perform such attacks, based on reflection using this forgery scheme, adversaries usually evade detection by spoofing the source IP address

Although there are various forms of attacks, the defense scheme against attacks is usually divided into three steps, namely, detection, classification, and mitigation. Table 3 summarizes the DDoS defense solutions, focusing on how to detect an attack, the scenarios in which the attack has occurred, and how to mitigate the attack.

4.3.1. DDoS Attack. Grigoryan et al. [67] propose a cooperative mitigation mechanism for DDoS attacks, LAMP (Layer 7 Attack Mitigation with Programmable data planes), which relies on information from the application layer. If a host detects a DDoS attack on the application layer, an attack flag will be padded to the IP option header field of a packet, and the packet will be sent to the ingress switch. The ingress switch that received the packet with attack flag will drop all other packets of the flow.

A Telephone Denial of Service (TDoS) event means that there are a large number of malicious SIP INVITE packets in the network, which prevents users from using telephone services. Febro et al. [68] propose a mitigation mechanism, TDoSD@DP, against SIP proxy DDoS attacks. The stateful P4 register records the number of INVITE and BYE messages. If there is no BYE message after a large amount of INVITE information, the SIP INVITE packet will be discarded. TDoSD@DP mitigates the attack in the place closest to the source of the attack, avoiding more resource consumption caused by attack packets entering the network, and the detection and mitigation algorithm is simple.

In the follow-up work [69], the authors propose an improved method to implement a DDoS defense solution based on distributed sources to detect attacks near the source. In this solution, each port of the P4 switch records the number of packets. The controller compares the values counted at each port with a threshold. If the threshold is exceeded, the P4 switch will install a rule of discarding packets from the same port. By detecting malicious traffic in the first hop, this scheme saves computing and bandwidth resources.

In response to reflective amplification attacks, Kuka et al. [70] present an FPGA acceleration device, which filters malicious traffic in the backbone infrastructure at a high speed and efficiency before the attack traffic reaches the attacked target. The authors transplant a VHDL implementation to P4 program running on FPGA. The device extracts the packet that belongs to the affected part of the input traffic, and sends a digest to the controller. The impact of each source IP address on the attack is used to identify the offensive IP addresses by the heuristic algorithm. Once an attack is detected, the packet drop rule will be installed.

As a defense mechanism, Pushback [71] can identify the pattern of attack traffic and limit the traffic near the attack source. However, Pushback cannot identify more complex and smarter DDoS attack patterns due to the limited computing capacity of traditional switches. Mi et al. [72] propose an improved scheme, ML-Pushback (machine learning-based Pushback), for DDoS attack mitigation mechanism through machine learning technology. In ML-Pushback, the P4 switch collects discarded packets and sends a digest to the control plane. The deep learning module on the control plane extracts signatures on the digest and uses a decision tree model to classify the collected abstracts. After determining that it is attack traffic, it reduces the attack by limiting its rate.

As DDoS attacks usually cause disturbances in the distribution of IP addresses, BUNGEE [73] detects a DDoS attack through the entropy analysis of the addresses of incoming packets. Once a switch detects any entropy disturbances, it sends alert information to its upstream switches. These upstream devices filter packets from the device that raised the alarm. In addition, these devices also perform detection mechanism and alert their upstream devices, finally "pushing back" the effects of the attack. To confine the attack flows as close to the source as possible, the process above is repeated.

Lapolli et al. [74] implement an entropy-based DDoS detection scheme in a programmable switch. In order to meet the processing time requirement with limited storage space, the count-min sketches data structure is used to estimate the frequency of the source IP address and the destination IP address of the packet in the observation window. A memory-optimized longest prefix matching (LPM) lookup table is used to solve the computationally intensive arithmetic function and estimate the entropy value. When the entropy value exceeds a certain threshold, an attacked alarm is issued. The authors implemented the above method in a software switch and tested it with an anonymous network tracking data set.

Khooi et al. [75] propose a distributed in-network defense architecture (DIDA). Utilizing the programmable stateful data plane and effective data structure, they prove that the connection of each user can be tracked automatically and distributedly without taking up the energy of the network controller. DIDA relies on the count-min sketches data structure and the set monitoring interval to correlate the numbers of requests and responses to determine whether a device has sent unsolicited attack packets to victims in the ISP network. If a DDoS attack is detected, the edge router uses ACL to block traffic approaching the attacker.

Aiming at the multiple features of TCP/UDP traffic, Dimolianis et al. [76] implement a multifunctional DDoS defense scheme. This program detects three traffic features of DDoS attacks: (i) the number of packets received within a time interval; (ii) the percentage of the total traffic received by a specific subnet; (iii) the percentage of inflow to outflow. Two cases are considered for evaluation: two feature cases containing only the first two features (F2) and three feature cases containing all three features (F3). The results show that in small-scale attacks, the first two features are enough to identify the DDoS, while in the other two scenes, more features is better.

To defense both slow and volumetric DDoS attacks, Friday et al. [77] propose a DDoS detection and mitigation strategy. The strategy consists of two parts. Firstly, Bloom filters and time-related statistics are used for one-way traffic analysis. Then, the bandwidth used by various applications is recorded. The dynamic threshold is determined according to the statistical information. When the behavior of a specific

Table 3: DDoS attack defense schemes.

Ref	Field	Defense steps			Platform	Limitation
		Detecting	Classifying	Relieving		
LAMP [67]	Application layer attack	The host adds a label in a packet when an application layer attack is detected, and the ingress switch extracts the label	N/A	The ingress switch drops packets carrying the label	bmv2	The host needs to be modified to support tag addition and removal
TDoSD@DP [68]	SIP DDoS	The stateful P4 register records the number of SIP INVITE and SIP BYE packets	No BYE packet after a large amount of INVITE packets	Dropping SIP INVITE packets	bmv2	No support for multiple variable-length fields of the SIP header; based on $P4_{14}$; only the first 8 bytes of SIP packets can be detected
[69]	SIP DDoS	The controller compares the number of each port with the threshold	Exceeding threshold	According to the command of the controller, P4 switch will drop packets belonging to the same entry	bmv2	No support for encrypted packets (such as SIP/TLS)
[70]	AR-DDoS attack	Extracting the packet and forwarding a digest to the SDN controller	Identifying offensive IP addresses by the influence of each source IP address in the attack	SDN controller will install packet drop rules	FPGA	The controller detects and identifies attacks
ML-Pushback [72]	Pushback effect of DDoS attack	P4 switch sends a digest to the control plane	The deep learning module extracts the signature and uses the decision tree model to classify the collected abstracts	Limiting its rate	N/A	Relying on external calculations
BUNGEE [73]	Pushback effect of DDoS attack	Detecting a DDoS attack through the entropy analysis of the addresses of incoming packets	Exceeding Threshold	Sending alert information to its upstream switches. The upstream device runs filtering strategy	bmv2	Fixed threshold
[74]	DDoS attack	Estimating the frequency of the IP addresses of packets and then estimating the entropy value	Entropy exceeds a certain threshold	Raising an attack alert	bmv2	No clear mitigation scheme
DIDA [75]	AR-DDoS attack	Comparing the numbers of requests and responses based on the count-min sketches data structure and the sample interval	A device sends an unsolicited attack packet to the victim in the ISP network	Edge routers use ACLs to block traffic approaching the attacker	bmv2	Information transfer at the network level is not described in detail
[76]	DDoS attack	Extracting traffic characteristics, such as total traffic, proportion of inflow and outflow, and proportion of total subflow	Multiple feature values exceed the threshold	Raising an attack alert	Netronome SmartNIC	TCAM is required to store traffic information

Table 3: Continued.

Ref	Field	Defense steps			Platform	Limitation
		Detecting	Classifying	Relieving		
[77]	Volumetric and slow DDoS	Using Bloom filters and time-related statistics for traffic analysis; recording the bandwidth used by various applications and their respective transmission protocols	Determining the dynamic threshold based on the statistical information, specifically which traffic behavior exceeds the threshold	Dropping or rating limit	bmv2	Only comprehensive evaluation, lack of experimental data
DroPPPP [78]	DoS attack	Comparing the hash value of the source IP and MAC address with the stored hash value	The two values are not the same, and the time of the last attack is within 5s	Dropping	bmv2	TCAM is required to store precalculated or estimated values
INDDoS [79]	Volumetric DDoS attack	Adopting a sketch-based BACON data structure to estimate the traffic and perform theoretical analysis on it	Detecting destination IP addresses whose number of incoming connections exceeds threshold	Marking the victim	Tofino	Only the detected attack victims are reported without effective attack mitigation
[80]	SYN and DNS spoofing	N/A	N/A	Utilizing advanced dynamic resource sharing methods to allocate required mitigation resources	p4c-behavioral	Only the mitigation after the attack is detected; there is no detection scheme
[81]	SYN flooding and ARP spoofing	Comparing the number of SYN/ACK packets and ACK/FIN packets	The ratio of SYN/ACK and ACK/FIN packets exceeds a threshold	The controller installs packet discarding rule	bmv2	The data plane cache efficiency is low
[82, 83]	TCP SYN flooding	Counting the number of attempts of TCP SYN flooding	The number of attempts exceeds threshold	Dropping	bmv2 and NetFPGA SUME	The register needs to retain the statistical information of each session
[84]	TCP flooding	Extracting the features used in the maximum likelihood classifier	Whether it has been attacked is judged based on the result of learning	N/A	bmv2	No clear mitigation scheme

traffic exceeds the threshold, it is determined as malicious traffic, and the traffic is discarded or the rate is limited. Administrators can provide network parameters for dynamic threshold calculation and then install them on the data plane through the API.

The packets which perform DoS attacks usually come from different source IP addresses or MAC addresses, thus having different combinations of source MAC address and IP address. Based on the above observation, DroPPPP [78] is proposed to mitigate DoS attacks by P4 switch. This method compares the hash value of the source IP and MAC address with the hash value stored. If there is no match and the time of the last attack is within 5s, an attack is detected. TCAM is used to store the precalculated value and estimated value in the match-action table of the data plane switch.

Ding et al. [79] present a simple large-scale in-network DDoS victim identification solution, INDDoS. It records the number of target IP addresses based on the BACON sketch. The target whose IP address is touched by multiple source IPs exceeding a threshold on the data plane within a given time interval is marked as a victim. Ding et al. [79] describe the necessary modifications to implement INDDoS in a Tofino switch due to the limited physical resources and set the parameters based on the optimal values of theoretical analysis.

4.3.2. Spoofing and Flooding. Traditional DDoS antispoofing cleaners require specialized intermediate equipment, which increases the cost, delay, and complexity of the network. Afek et al. [80] implement OpenFlow 1.5- and P4-based network antispoofing systems to resist SYN and DNS spoofing. Adopting a high-speed programmable data plane, the scheme can execute low-level primitive operations at line rate. The numbers of rules and the controller messages exchanged are in proportion to the legitimate traffic. To protect several large servers, switches share the flow tables in a distributed manner.

Lin et al. [81] propose a mechanism to mitigate ARP spoofing and SYN flooding attacks. The proposed scheme counts the number of SYN/ACK and ACK/FIN packets and informs the controller if it is abnormal. The controller adds the IP address involved in the attack to the packet discarding rule. Lin et al. [81] implement a software switch based on OpenFlow and P4 and compare the performance in the two cases. In the OpenFlow-based solution, the throughput rate is relatively large, while the P4-based solution can work on the data plane, which greatly reduces the related data traffic.

In response to TCP SYN flooding attacks, Paolucci et al. [82, 83] provide dynamic traffic engineering schemes for optical bypass and traffic offloading and propose a stateful attack mitigation mechanism based on P4. The P4 register retains statistics for each session to record SYN flooding attacks. The port number of the last SYN packet and the number of SYN flooding attempts are recorded by different registers. If the number of attempts detected is larger than a predetermined threshold, the packet is dropped. The authors have implemented the above functions on the software switch bmv2 and NetFPGA SUME boards, respectively.

In response to TCP flood attacks, Musumeci et al. [84] combine maximum likelihood and P4 to effectively perform data mining and propose a P4-based DDoS attack mitigation measure using ML classifier. The authors compare the accuracy and complexity of different maximum likelihood estimation algorithms, that is, algorithm training time and prediction time. P4 code is provided to extract the features used in the maximum likelihood classifier. Features are embodied in the form of metadata, which is used to associate additional information with the packet itself. Whether it has been attacked is determined according to the results of the learning.

4.4. Integrated Defense. Compared with applications implemented in software, data plane computing can increase throughput by an order of magnitude or more. The performance improvement makes it possible to implement functions that were once performed on the control plan, including distributed computing, deep packet inspection, and machine learning acceleration. By offloading some algorithms to the data plane, more accurate and responsive solutions can be deployed, which helps reduce the overhead of the control plane.

4.4.1. Network-Level Defense. Xing et al. [85] propose a programmable and distributed link flood defense system, Ripple, which can resist dynamic changes of LFA. Ripple has developed a strategy language, a compiler, and a distributed runtime for link flood protection. The Ripple language derives an abstract defense panorama to accurately extract threat signals from network-wide traffic. Based on this panoramic view, panoramic defense can be achieved in a fully distributed manner. The attack signal is first extracted by the local switch and then propagated to achieve view synchronization.

Poseidon [86] is proposed to mitigate volumetric DDoS attacks. A set of defense primitives which is used to express a series of security policies is proposed. The defense primitives are partitioned according to their attributes for execution on switches and servers. Firstly, users adopt a set of defense primitives to represent their defense strategy. Then, the defense primitives are mapped to the programmable switch and, if necessary, to the server software to achieve effective defense. When the attack changes, the underlying defense primitives can be reconfigured according to the attack mode. Evaluation based on the prototype shows that the proposed scheme can resist volumetric attacks and adapt to dynamic attacks with low overhead. The shortcoming is the manual intervention of writing a defense strategy.

Liu et al. [87] propose a switch solution, Jaqen, for DDoS defense, which can run a wide range of detection and mitigation functions locally and respond to large-scale dynamic adaptive attacks. Jaqen designs switch-optimized, resource-efficient components for detection and mitigation. A flexible API is designed to build various best practice (and future) defense strategies that can effectively utilize the switching function. A network-wide resource manager is constructed, which can quickly adapt to changes in attack status. Experiments show that Jaqen can mitigate large-scale attacks at line rate (380 Gbps).

Xing et al. [88] propose an abstract solution, FastFlex, that designs defense measures into the network path based on constantly changing attacks and transfers the defense to the data plane. FastFlex proposes a key abstraction, the multimodal data plane, and solves the challenges of programming defense in the data plane. Usually, the network runs under the best configuration of centralized control computing. However, once it is attacked, it performs in a distributed manner to mitigate the attack. Hybrid vector attacks will trigger coexistence patterns in different areas, and changing attacks will respond with the same fast pattern adaptation. The disadvantage is the high complexity and security issue of cross-domain joints.

4.4.2. Deep Packet Inspection. DeepMatch [89] uses network processors (NPs) to perform line-speed regular expression matching on payloads, supports packet counting and reordering, and performs flow-based DPI. The authors implement DeepMatch on the Netronome SmartNIC. Through the regular expression matching function, the stateless packets are internally matched, and the execution speed can reach the line rate of the target; for the internal matching of stateful packets, the execution speed is about 20 Gbit/s. DeepMatch [89] introduces deep packet inspection (DPI) for packet payload.

Li et al. [90] propose a Hop-Count Filter (HCF) defense mechanism, NetHCF, which can reduce deceptive IP traffic. In NetHCF, the data plane provides services for legitimate packets at line rate. The control plane deals with the mirror of the lost packets and dynamically adjusts the status according to the network. The binding relationship between IP address and hop count is maintained in a table, and the hop count is incorrect when NAT is present.

Xing et al. [91] propose a defense system, called NetWarden, against network covert channels with unaffected performance. The programmable data plane is adopted to check and modify each packet header without communication interruption or performance limitation. Performance enhancement technology is used to offset the performance loss caused by the covert channel defense. NetWarden influences the feedback of the TCP congestion control mechanism of the sender and receiver in the form of a proxy, improves the sending rate, and reduces the retransmission. In order to compensate for the strict model of the programmable data plane and the inability to support all operations required for covert channel defense, a "fastpath/slowpath architecture" is proposed, which combines the versatility of software and the efficiency of hardware. Hardware fastpath supports some key operations that need to run at line rate, and software slowpath supports more expressive general operations with only a few calls. The hardware prototype of NetWarden is implemented in P4, which can detect a series of network hidden channels at full speed, alleviate these channels with negligible performance interference, and work smoothly in complex applications. The disadvantage is slowpath/fastpath communication delay.

4.4.3. Machine Learning. Intelligent decision support systems usually rely on a collection of traffic features, which are then input into a complex system based mainly on ML algorithms. Since ML can detect and discover patterns and behaviors that were previously invisible in network traffic, it helps to improve accuracy and responsiveness by understanding traffic in more detail, so it has been successfully applied to network security. Most of the ML-based security methods developed in the SDN environment are implemented on the control plane, and these schemes may bring problems related to accuracy and high overhead [92]. The programmability of the data plane makes ML-based security solutions more feasible.

The traditional table-based switch function has a limited ability to identify complex network attacks. Machine learning-based methods that provide high-precision have been widely used for packet classification, but they usually require packets to be forwarded to additional hosts, thus increasing network latency. The traditional machine learning-based methods always need to forward packets to additional hosts, thus increasing network latency. In order to overcome the limitation, the authors in [93] propose an IDS scheme based on Binary Neural Network (BNN) and federated learning. To implement the scheme on the data plane, BNN compresses the neural network into a simplified form, the weight is compressed into a single bit, and related calculations are converted into bitwise operations. Then, BNN is applied to identify the incoming packets at the edge of the network. In addition, the authors propose a federated learning scheme to continuously train the BNN function. The controller connects a lot of P4 targets and trains BNN using the samples received from the P4 targets. The cloud service collects these updates from the controller and updates the processed weight to the controller.

Barradas et al. [94] propose the system FlowLens which uses programmable switch to effectively support multipurpose security applications based on ML. FlowLens collects the characteristics of packet distribution at line rate and directly classifies the flow on the switch, so that network operators can reuse this measurement primitive to serve different flow classification tasks at runtime. To overcome the shortcomings of the data plane, FlowLens calculates an efficient memory representation of related characteristics for each flow, called "flow tag." Although the size of the traffic marker is small, it contains enough information for the switch to classify packets.

Gutiérrez et al. [95] focus on using the functions provided by the programmable data plane to implement AI, especially ML algorithms, and try to figure out how many operations in the algorithm can be executed on the data plane. How the data plane can complement the different stages of the machine learning-based Intrusion Detection System is discussed. Two cases demonstrate the feasibility of this method and emphasize the factors that must be considered when solving the indirect tasks of deploying solutions.

4.4.4. Finite State Machine. Laraba et al. [96] adopt an extended finite state machine (EFSM) to model the stateful security monitoring function. P4 switch is used to detect ECN (Explicit Congestion Notification) behavioral abnormalities. A seven-tuple is used to represent EFSM, which can describe more complex conditions and avoid creating a large number of states for variables. The status and variables during operation are stored in registers. If the final host does not match the state machine, the packet is discarded or, if possible, the wrong behavior is corrected.

5. P4 Target Comparison

P4 supports software-based targets and hardware-based targets, such as NPU, FPGA, and ASIC. Table 4 summarizes the performance of different P4 targets [6, 8, 97], representative products, and their proportions in research. Most of the research work is implemented on software switches, reaching nearly half.

(1) The use of software to realize network protocols is flexible in terms of functionality and complexity, but the processing capacity of the CPU limits the throughput, and interrupt and cache effects may influence delay and jitter.

(2) NPU is a kind of software-programmable ASICs which are optimized for network applications. They are part of a standalone network device or network card. Compared to the software switch, throughput and delay performance of NPU are relatively improved, but flexibility is reduced.

(3) FPGA can be programmed to realize almost any function. Apart from the limit on memory resources, FPGAs usually exceed most targets in terms of throughput, jitter, and latency. However, FPGA programming requires knowledge of specific hardware, and implementing network algorithms in HDL is very time-consuming.

(4) ASIC excels in terms of efficiency, that is, throughput, delay, and jitter. However, the limited expressive power of the ASIC instruction set also limits the flexibility of its function implementation.

There are several papers that compare the performance of multiple target data planes when implementing specific

Table 4: Performance comparison of different P4 targets.

	Throughput	Delay	Jitter	Resource constraints	Flexibility	Representative products	Proportion in the research (%)
Software/ CPU	+	>10 us	−−−−	++++	++++	p4c-behavioral, bmv2, T4P4S	48.5
NPU	++	5 us~10 us	−−−	+++	+++	Netronome NFP-6000 SmartNIC [98], Pensando Capri [99]	5
FPGA	+++	<2 us	−−	++	++	NetFPGA SUME [13], P4FPGA [100]	7.9
ASIC	++++	<2 us	−	+	+	Barefoot Tofino 1, Tofino 2 [15], Broadcom Tomahawk [101]	38.6

algorithms and strategies. The comparison is helpful to understand the limitations of related equipment. Table 5 compares the concrete behaviors of different targets.

Scholz et al. [102] discuss the necessity of implementing cryptographic hash functions in the data plane to mitigate potential attacks against hash collisions and propose the prototype implementation of the cryptographic hash function in three different P4 targets (CPU, SmartNIC, NetFPGA SUME). The main evaluation results can be seen in Table 5. Overall, the CPU-based implementation has the highest flexibility, but the SmartNIC-based implementation has the highest throughput. Users can adopt different platforms to implement the hash algorithm according to their performance requirements.

The extent to which the defects can be exploited in practice is discussed firstly in article [103]. Firstly, the authors write a series of programs which are composed of simple operations to test the three targets (bmv2 switch, P4NetFPGA and Tofino switch). In some scenarios, the behavior of these targets is inaccurate and needs to be paid attention to in programming to avoid these extreme phenomena. Then, the NAT (Network Address Translation) function implemented in P4 is checked to see if it contains any exploitable vulnerabilities. It is found that powerful attackers can use the simple NAT program and the classical switch program (switch.P4) to implement attacks against different targets. At last, the impact of the findings above on security is discussed.

Scholz et al. [104] discuss how to use their performance to protect the entire network from SYN flooding attacks. Thus, two defense schemes, SYN authentication and SYN cookie, are analyzed. The implementation difficulties when porting them to different targets (software, FPGAs, and network processors), as well as the performance of these three targets, are discussed in detail.

6. Trends and Challenges

Undoubtedly, the P4 programmable data plane will solve problems in new fields of the network, or newly appearing problems in traditional areas, inspire more data plane innovations, accelerate network equipment updates, and promote the network in the direction of being programmable and customizable. Overall, there are two main trends.

> One is function offloading: due to the high-speed data processing capability and programmability, part of the function of the control plane can be performed by the data plane in whole or in part. This will be a new attempt for the traditional SDN centralized controller solutions. In addition, some of the detection and defense tasks originally performed by the server can also be offloaded to the smart network card to reduce the processing burden of the server.
>
> The other is application innovation: the programmable data plane can add, delete, and modify protocol fields during data transmission and perform further processing, which provides great convenience for the realization of the new protocol. Based on the above characteristics, more and more new application research works are proposed, such as aggregation and disaggregation of IoT packets [105], network coding [106], and handover in 5G [107].

However, compared to solving application problems in new fields, it is urgent and inevitable to solve the problem of its own deficiencies in the programmable data plane.

6.1. Security Verification

6.1. Security Verification. Although the programmability of the data plane provides great convenience for rapid innovation and protocol development, it also involves hidden dangers because of the lack of correctness verification of the data plane.

Compared with the equipment manufacturer, the general user, who defines the forwarding behavior of a data plane, usually lacks experience. Attackers may launch an attack on the device based on the bugs in the program. It is particularly important to establish automated testing and verification tools to improve the security of the programmable data plane.

Due to different programming objects, existing software testing tools cannot be used directly for P4 programs. The design and implementation of program analysis and verification technology for P4 constitute an important research direction in the future.

6.2. Complex Calculation. Programmable switching only supports simple arithmetic operations for integer values, and line rate processing means that each packet only supports fewer operation steps. Although it is possible to increase the processing flow through loops, it is implemented at the expense of throughput, and it is just a superposition of simple operations rather than complex calculations in the

Table 5: Concrete behaviors on different targets.

	Purpose		CPU (bmv2)	FPGA (NetFPGA)	NPU (SmartNIC)	ASIC (Tofino)
[102]	Implementing cryptographic hash function		Easy to implement, but with high latency (a few milliseconds)	It has the lowest latency but cannot be integrated using native P4 functions	It has the highest throughput, but cannot handle packets up to 900 bytes	——
[103]	Impact of programs' bugs on network security	Reading invalid headers	Reading the flag value from the previous IPv4 packet	Reading 0	——	Reading the value of the Explore Flags field
		Writing invalid headers	Writing and reading successfully	Writing and reading successfully	——	It can Write and read, but writing an invalid IPv4 header will cause the Explore header to change
		Loops	Resubmitting: blocks processing, discarding all subsequent packets	Unsupported	——	Resubmitting: no more than one resubmission; more resubmissions will be deleted; egress to egress clone: causing flooding
		Resurrecting dead packets	The field marked as "deprecated" can be used to store metadata which marks dropped packets	The packet whose egress metadata is set as the value of the egress port will be restored rather than discarded	——	Packets that do not specify a valid egress port are discarded by default
[104]	Implementing SYN defense strategy		It cannot handle SYN flood traffic up to 14 Mpps	It can handle P4 programs with higher complexity; the delay is less than 10 μs, and there is no long tail; it only consumes one-third of the total resources	It processes data at almost linear speed; delay is between 1 and 4 μs;	——
[63]	Implementing IP Sec		The delay between the P4 switch and the SDN controller is very low in experiment. Actually, the transmission delay is relatively large, covering the impact of key generation	(1) No support for parsing variable-length header fields. (2) Lack of data exchange between P4 pipeline and P4 extern	——	No support for user-defined P4 extern that contains computationally intensive functions. In this solution, P4 extern is relocated to CPU or directly forwards IPsec-related streams to the encryption host

true sense. The lack of complex calculations affects the application scenarios of the data plane.

Existing schemes for overcoming computational limitations include precalculation and approximation. In the precalculation scheme, the value is calculated by the control plane or the extern CPU and installed in the match-action table or stored in the register, and the data plane queries related entries according to certain rules to obtain the calculated result. This method is only suitable for scenes with a small range and few target values. Another attempt is that approximate values can be used by applications instead of accurate values, which require extremely high computational overhead to obtain. The approximation scheme can achieve certain functions, but at the cost of accuracy.

In the future, the device manufacturer should produce devices whose data plane can support richer computing operations and has a higher computing capacity. On the other hand, the function which requires complex calculations can be decomposed into simpler operations. Efforts can be made to explore suitable decomposition methods for complex calculations.

6.3. Stateful Network Function. As a key feature of the programmable data plane, state processing can realize various novel applications that are impossible in non-programmable networks, such as fine-grained load balancing, distributed network telemetry, and stateful traffic engineering.

Since a certain amount of state data needs to be retained on the programmable data plane, the size of the on-chip memory directly affects the amount of state data that can be stored, further limiting the scale of flows that can be processed and the functions that can be implemented by the data plane.

For the problem of limited storage space for state information, one possible solution is that the memory of programmable switch should be expanded by adding more SRAM or DRAM, which may lead to higher costs and affect flexibility and scalability. The algorithm needs to be reasonably designed to allow the data plane to save as little state data as possible. Meanwhile, the scheduling between the data plane and the extended memory needs to be carefully designed to minimize the impact on performance.

7. Conclusion

The research of solving network security problems with P4-based programmable data plane is reviewed in this paper. The paper introduces the programming language P4 and components in the P4 workflow and discusses the development of network programmability. The advantages of P4-based programmable data plane in solving security problems are analyzed. On this basis, according to the perspectives of passive defense, active defense, and combination of multiple technologies, the existing network security applications are divided into four parts: access control, privacy and encryption, availability, and integrated defense. The schemes in each category are compared, and the core ideas and limitations are clarified. In addition, a detailed comparison is made for the research on performance of P4 target, and the research on using multiple target platforms to implement functions or algorithms is summarized. Finally, challenges related to the limitations of P4-based programmable data plane are discussed.

Data Availability

The labeled data sets used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare no conflicts of interest.

Acknowledgments

This work was supported by the Key University Science Research Project of Jiangsu Province under grant 19KJD510008 and the Future Network Scientific Research Fund Project under grant FNSRFP-2021-YB-55.

References

[1] Global DDoS threat Report for the first half of 2021, https://mp.weixin.qq.com/s/5P1TAx-U5bnfA5ePdNPmkw.

[2] 2021 Threat Hunting Report by CrowdStrike, https://www.crowdstrike.com/resources/reports/threat-hunting-report-2021/.

[3] N. McKeown, T. Anderson, H. Balakrishnan et al., "OpenFlow," *ACM SIGCOMM-Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[4] P. Bosshart, G. Varghese, D. Walker, and D. Daly, "P4: programming protocol-independent packet processors," *ACM SIGCOMM-Computer Communication Review*, vol. 44, no. 3, pp. 87–95, 2014.

[5] T. Dargahi, A. Caponi, M. Ambrosin, G. Bianchi, and M. Conti, "A survey on the security of stateful SDN data planes," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1701–1725, 2017.

[6] F. Hauser, M. Häberle, D. Merling et al., "A survey on data plane programming with P4: fundamentals, advances, and applied research," 2021, https://arxiv.org/abs/2101.10632.

[7] S. Kaur, K. Kumar, and N. Aggarwal, "A review on P4-Programmable data planes: architecture, research efforts, and future directions," *Computer Communications*, vol. 170, pp. 109–129, 2021.

[8] E. F. Kfoury, J. Crichigno, and E. Bou-Harb, "An exhaustive survey on P4 programmable data plane switches: taxonomy, applications, challenges, and future trends," *IEEE Access*, 2021.

[9] M. Karakus and A. Durresi, "A survey: control plane scalability issues and approaches in Software-Defined Networking (SDN)," *Computer Networks*, vol. 112, pp. 279–293, 2017.

[10] Website of the P4 Language Consortium, https://p4.org/.

[11] The P4 Language Specification, https://p4.org/p4-spec/p4-14/v1.0.5/tex/p4.pdf.

[12] P4 Language Consortium, P4_16 Language Specification, P4.

[13] M. Shahbaz, S. Choi, P. Ben et al., "PISCES: A programmable, protocol-independent software switch," in *Proceedings of the 2016 ACM SIGCOMM Conference*, pp. 525–538, Florianopolis, Brazil, August 2016.

[14] N. Zilberman, Y. Audzevich, G. A. Covington, and A. W. Moore, "NetFPGA SUME: toward 100 Gbps as research commodity," *IEEE Micro*, vol. 34, no. 5, pp. 32–41, 2014.

[15] Behavioral model (bmv2), 2015, https://github.com/p4lang/behavioral-model.

[16] Barefoot Tofino2, https://www.barefootnetworks.com/products/brief-tofino-2/.

[17] N. McKeown, T. Sloane, and J. Wanderer, *P4 Runtime-Putting the Control Plane in Charge of the Forwarding Plane*, 2017, https://p4.org/api/p4-runtime-putting-the-control-plane-in-charge-of-theforwarding-plane.html.2017.

[18] D. Horpácsi, S. Laki, P. Vörös, M. Tejfel, P. Gergely, and L. Molnár, "Asynchronous extern functions in programmable software data planes," in *Proceedings of the P4 Workshop in Europe (EuroP4)*, Cambridge, UK, September 2019.

[19] J. Geng, J. Yan, and Y. Zhang, "P4QCN: congestion control using P4-capable device in data center networks," *Electronics Journal*, vol. 8, 2019.

[20] C. Papagianni and K. De Schepper, "PI2 for P4: an active queue management scheme for programmable data planes," in *Proceedings of the 15th International Conference on emerging Networking Experiments and Technologies*, pp. 84–86, Orlando, FL, USA, December 2019.

[21] Naga Katta, M. Hira, C. Kim, A. Sivaraman, and J. Rexford, "HULA: scalable load balancing using programmable data planes," in *Proceedings of the ACM Symposium on SDN Research (SOSR)*, Santa Clara, CA, USA, March 2016.

[22] B. Lewis, L. Fawcett, M. Broadbent, and N. Race, "Using P4 to enable scalable intents in software defined networks," in *Proceedings of the IEEE International Conference on Network Protocols (ICNP)*, Cambridge, UK, September 2018.

[23] O. Karrakchou, N. Samaan, and K. Ahmed, "ENDN: an enhanced NDN architecture with a P4-programmable data

plane," in *Proceedings of the International Conference on Networking (ICN)*, Lisbon, Portugal, February 2020.

[24] F. Paolucci, F. Cugini, P. Castoldi, and T. Osinski, "Enhancing 5G SDN/NFV edge with P4 data plane programmability," *IEEE Network*, vol. 35, no. 3, pp. 154–160, 2021.

[25] F. E. Rodriguez Cesen, L. Csikor, C. Recalde, C. E. Rothenberg, and P. Gergely, "Towards low latency industrial robot control in programmable data planes," in *Proceedings of the IEEE Conference on Network Softwarization (NetSoft)*, Ghent, Belgium, 2020.

[26] R. Kundel, F. Siegmund, and B. Koldehofe, "How to measure the speed of light with programmable data plane hardware?" in *Proceedings of the P4 Workshop in Europe (EuroP4)*, Cambridge, UK, September 2019.

[27] D. Sanvito, G. Siracusano, and R. Bifulco, "Can the network be the AI accelerator?" in *Proceedings of the 2018 Morning Workshop on In-Network Computing*, pp. 20–25, Budapest, Hungary, August 2018.

[28] Z. Xiong and N. Zilberman, "Do switches dream of machine learning? toward in-network classification," in *Proceedings of the 18th ACM Workshop on Hot Topics in Networks*, pp. 25–33, Princeton, NJ, USA, November 2019.

[29] Facebook Announcement: Wedge 100, https://www.edge-core.com/tw/news-inquiry.php?cls=7&id=205.

[30] Y. Li, R. Miao, H. H. Liu et al., "HPCC: high precision congestion control," in *Proceedings of the ACM Special Interest Group on Data Communication, ser. SIGCOMM'19*, New York, NY, USA, August 2019.

[31] Data center network solution based on SONiC, https://www.caict.ac.cn/pphd/zb/odcc/2019/3/201909/t20190902_210577.htm.

[32] Open Networking Foundation (ONF), "Stratum - ONF launches major new open source SDN switching platform with support from Google," https://tinyurl.com/yy3ykw7g.

[33] Asterfusion: P4 CX-T switch, https://asterfusion.com/zh/cloudnetwork-switches/cx-t.

[34] J. Steadman and S. Scott-Hayward, "DNSxP: enhancing data exfiltration protection through data plane programmability," *Computer Networks*, vol. 195, pp. 1–13, 2021.

[35] J. Alvarez-Horcajo, I. Martinez-Yelmo, D. Lopez-Pajares, J. A. Carral, and M. Savi, "A hybrid SDN switch based on standard P4 code," *IEEE Communications Letters*, vol. 25, no. 5, pp. 1482–1485, 2021.

[36] P. Vörös and A. Kiss, "Security middleware programming using P4," in *Proceedings of the International Conference on Human Aspects of Information Security, Privacy, and Trust (HAS)*, Vancouver, BC, Canada, July 2016.

[37] J. Cao, J. Bi, Y. Zhou, and C. Zhang, "CoFilter: a high-performance switch-assisted stateful packet filter," in *Proceedings of the ACM SIGCOMM 2018 Conference on Posters and Demos*, Budapest, Hungary, August 2018.

[38] J. Deng, H. Hu, H. Li et al., "VNGuard: an NFV/SDN combination framework for provisioning and managing virtual firewalls," in *Proceedings of the IEEE Conference on Network Function Virtualization and Software-Defined Networking (NFV-SDN)*, San Francisco, CA, USA, 2015.

[39] R. Datta, S. Choi, A. Chowdhary, and Y. Park, "P4Guard: designing P4 based firewall," in *Proceedings of the IEEE Military Communications Conference (MILCOM)*, Los Angeles, CA, USA, October 2018.

[40] R. Ricart-Sanchez, M. Pedro, J. M. Alcaraz-Calero, and Q. Wang, "NetFPGA-based firewall solution for 5G multi-tenant architectures," in *Proceedings of the IEEE International Conference on Edge Computing (EDGE)*, Milan, Italy, July 2019.

[41] A. Almaini, A. Al-Dubai, I. Romdhani, and M. Schramm, "Delegation of authentication to the data plane in software-defined networks," in *Proceedings of the 2019 IEEE International Conferences on Ubiquitous Computing & Communications (IUCC) and Data Science and Computational Intelligence (DSCI) and Smart Computing, Networking and Services (SmartCNS)*, Shenyang, China, 2019.

[42] E. O. Zaballa, D. Franco, Z. Zhou, and M. S. Berger, "P4Knocking: offloading hostbased firewall functionalities to the network," in *Proceedings of the 2020 23rd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, Paris, France, February 2020.

[43] A. Almaini, A. Al-Dubai, I. Romdhani, M. Schramm, and A. Alsarhan, "Lightweight edge authentication for software defined networks," *Computing*, vol. 103, no. 2, pp. 291–311, 2021.

[44] K. Qiao, L. Xue, M. Adam, Y. Tang, A. Chen, and X. Luo, "Programmable in-network security for context-aware BYOD policies," in *Proceedings of the USENIX Security Symposium*, Boston, MA, USA, August 2020.

[45] S. Bai, H. Kim, and J. Rexford, "Passive OS fingerprinting on commodity switches," in *Proceedings of the CoNEXT'19*, Orlando, FL, USA, December 2019.

[46] Y.-Z. Cai, T.-Y. Lin, Y.-T. Wang, Y.-P. Tuan, and M.-H. Tsai, "EReplacement: efficient scanner data collection method in P4-based software-defined networks," *International Journal of Network Management*, vol. 31, no. 6, p. e2162, 2021.

[47] D. Scholz, S. Gallenmüller, Henning Stubbe, and G. Carle, "SYN flood defense in programmable data planes," in *Proceedings of the 3rd P4 Workshop in Europe*, Barcelona, Spain, December 2020.

[48] H. Gondaliya, G. C. Sankaran, and K. M. Sivalingam, "Comparative evaluation of IP address anti-spoofing mechanisms using a P4/NetFPGA-based switch," in *Proceedings of the P4 Workshop in Europe (EuroP4)*, Barcelona, Spain, December 2020.

[49] K. Peng, Y. Liu, and H. Lin, "P4DAD: securing duplicate address detection using P4," in *Proceedings of the IEEE International Conference on Communications (ICC)*, Montreal, QC, Canada, June 2020.

[50] G. K. Ndonda and R. Sadre, "A two-level intrusion detection system for industrial control system networks using P4," in *Proceedings of the International Symposium for ICS & SCADA Cyber Security Research (ICS-CSR)*, Hamburg, Germany, August 2018.

[51] B. Lewis, M. Broadbent, and N. Race, "P4ID: P4 enhanced intrusion detection," in *Proceedings of the IEEE Conference on Network Function Virtualization and Software-Defined Networking (NFV-SDN)*, Dallas, TX, USA, November 2019.

[52] A. Sanghi, K. P. Kadiyala, P. Tammana, and S. Joshi, "Anomaly detection in data plane systems using packet execution paths, SPIN'21," in *Proceedings of the ACM SIGCOMM 2021 Workshop on Secure Programmable network INfrastructure*, pp. 9–15, Vienna, Austria, December 2021.

[53] R. Meier, P. Tsankov, V. Lenders, L. Vanbever, and M. Vechev, "NetHide: secure and practical network topology obfuscation," in *Proceedings of the 27th{USENIX} Security Symposium*, pp. 693–709, Baltimore, MD, USA, August 2018.

[54] T. Datta, N. Feamster, J. Rexford, and L. Wang, "SPINE: surveillance protection in the network Elements," in *Proceedings of the USENIX Workshop on Free and Open*

Communications on the Internet (FOCI), Santa Clara, CA, USA, August 2019.

[55] D. Chang, W. Sun, and Y. Yang, "A SDN proactive defense mechanism based on IP transformation," in *Proceedings of the International Conference on Safety Produce Informatization (IICSPI)*, Chongqing, China, November 2019.

[56] Y. Qin, W. Quan, F. Song et al., "Flexible encryption for reliable transmission based on the P4 programmable platform," in *Proceedings of the 2020 Information Communication Technologies Conference (ICTC)*, Nanjing, China, May 2020.

[57] Y. Govil, L. Wang, and J. Rexford, "MIMIQ - masking IPs with migration in QUIC," in *Proceedings of the USENIX Workshop on Free and Open Communications on the Internet (FOCI)*, Austin, TX, USA, August 2020.

[58] M. Liu, D. Gao, G. Liu et al., "Learning based adaptive network immune mechanism to defense eavesdropping attacks," *IEEE Access*, vol. 7, pp. 182814–182826, 2019.

[59] G. Liu, W. Quan, N. Cheng, N. Lu, H. Zhang, and X. Shen, "P4NIS: improving network immunity against eavesdropping with programmable data planes," in *Proceedings of the IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Toronto, ON, Canada, July 2020.

[60] C. Zhou, W. Quan, D. Gao et al., "AMS: adaptive multipath scheduling mechanism against eavesdropping attacks with programmable data planes," in *Proceedings of the IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, Chongqing, China, 2021.

[61] H. Mohajeri Moghaddam and A. Mosenia, "Anonymizing masses: practical light-weight anonymity at the network level," 2019, https://arxiv.org/abs/1911.09642.

[62] X. Chen, "Implementing AES encryption on programmable switches via scrambled lookup tables," in *Proceedings of the Proceedings of the Workshop on Secure Programmable Network Infrastructure*, 2020.

[63] F. Hauser, M. Häberle, M. Schmidt, and M. Menth, "P4-IPsec: SitetoSite and host-to-site VPN with IPsec in P4-based SDN," *IEEE Access*, vol. 8, 2020.

[64] F. Hauser, M. Schmidt, M. Häberle, and M. Menth, "P4-MACsec: dynamic topology monitoring and data layer protection with MACsec in P4-based SDN," *IEEE Access*, vol. 8, 2020.

[65] H. Kim and A. Gupta, "ONTAS: flexible and scalable online network traffic anonymization system," in *Proceedings of the 2019 Workshop on Network Meets AI & ML - NetAI'19*, Beijing China, August 2019.

[66] Z. Zhibin, C. Chang, and X. Zhu, "A software-defined networking packet forwarding verification mechanism based on programmable data plane," *Journal of Electronics and Information Technology*, vol. 42, no. 5, pp. 1110–1117, 2020.

[67] G. Grigoryan and Y. Liu, "LAMP: prompt layer 7 attack mitigation with programmable data planes," in *Proceedings of the 2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*, Cambridge, MA, USA, November 2018.

[68] A. Febro, H. Xiao, and J. Spring, "Telephony denial of service defense at data plane (TDoSD@DP)," in *Proceedings of the IEEE/IFIP Network Operations and Management Symposium (NOMS)*, Taipei, Taiwan, April 2018.

[69] A. Febro, H. Xiao, and J. Spring, "Distributed SIP DDoS defense with P4," in *Proceedings of the 2019 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–8, IEEE, Marrakech, Morocco, April 2019.

[70] M. Kuka, K. Vojanec, K. Jan, and P. Benáček, "Accelerated DDoS attacks mitigation using programmable data plane," in *Proceedings of the ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, Cambridge, UK, September 2019.

[71] J. Ioannidis and S. M. Bellovin, "Implementing pushback: router-based defense against DDoS attacks," in *Proceedings of the NDSS*, San Diego, CA, USA, February 2016.

[72] M. Yu and A. Wang, "ML-Pushback," in *Proceedings of the 15th International Conference on emerging Networking Experiments and Technologies*, Orlando, FL, USA, December 2019.

[73] L. A. Quintero Gonz′alez, L. Castanheira, J. A. Marques, A. Schaeffer-Filho, and L. P. Gaspary, "BUNGEE: an adaptive pushback mechanism for DDoS detection and mitigation in P4 data planes," in *Proceedings of the 2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pp. 393–401, Bordeaux, France, May 2021.

[74] Â. Cardoso Lapolli, J. A. Marques, and L. P. Gaspary, "Offloading real-time DDoS attack detection to programmable data planes," in *Proceedings of the IFIP/IEEE Symposium on Integrated Management (IM)*, Washington, DC, USA, April 2019.

[75] X. Z. Khooi, L. Csikor, D. M. Divakaran, and M. S. Kang, "DIDA: distributed in-network defense architecture against amplified reflection DDoS attacks," in *Proceedings of the 2020 6th IEEE Conference on Network Softwarization (NetSoft)*, Ghent, Belgium, July 2020.

[76] M. Dimolianis, P. Adam, and V. Maglaris, "A multi-feature DDoS detection schema on P4 network hardware," in *Proceedings of the 2020 23rd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, Paris, France, February 2020.

[77] F. Kurt, E. Kfoury, E. Bou-Harb, and J. Crichigno, "Towards a unified in-network DDoS detection and mitigation strategy," in *Proceedings of the 2020 6th IEEE Conference on Network Softwarization (NetSoft)*, Ghent, Belgium, July 2020.

[78] Goksel Simsek, H. Bostan, A. K. Sarica et al., "Dropppp: a P4 approach to mitigating dos attacks in SDN," in *Proceedings of the International Workshop on Information Security Applications*, pp. 55–66, Springer, Thanjavur, India, November 2019.

[79] D. Ding, M. Savi, F. Pederzolli, M. Campanella, and D. Siracusa, "In-network volumetric DDoS victim identification using programmable commodity switches," 2021, https://arxiv.org/abs/2104.06277v3.

[80] Y. Afek, A. Bremler-Barr, and L. Shafir, "Network anti-spoofing with SDN data plane," in *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, Atlanta, GA, USA, May 2017.

[81] T.-Y. Lin, J.-P. Wu, P.-H. Hung et al., "Mitigating SYN flooding attack and ARP spoofing in SDN data plane," in *Proceedings of the 2020 21st Asia-Pacific Network Operations and Management Symposium (APNOMS)*, Daegu, Korea, September 2020.

[82] F. Paolucci, F. Cugini, and P. Castoldi, "P4-based multi-layer traffic engineering encompassing cyber security," in *Proceedings of the Optical Fiber Communication Conference (OFC)*, San Diego, CA, USA, March 2018.

[83] S. Yang, B. Lu, L. Cui et al., "An efficient pipeline processing scheme for programming Protocol-independent Packet Processors," *Journal of Network and Computer Applications*, vol. 171, 2020.

[84] F. Musumeci, V. Ionata, F. Paolucci, F. Cugini, and M. Tornatore, “Machine-learning-assisted DDoS attack detection with P4 language,” in *Proceedings of the IEEE International Conference on Communications (ICC)*, Montreal, QC, Canada, July 2020.

[85] J. Xing, W. Wu, and A. Chen, “Ripple: a programmable, decentralized link-flooding defense against adaptive adversaries,” in *Proceedings of the 30th USENIX Security Symposium*, Vancouver, Canada, 2021.

[86] M. Zhang, G. Li, S. Wang et al., “Poseidon: mitigating volumetric DDoS attacks with programmable switches,” in *Proceedings of the NDSS*, San Diego, CA, USA, June 2020.

[87] Z. Liu, N. Hun, G. Nikolaidis et al., “Jaqen: a high-performance switch-native approach for detecting and mitigating volumetric DDoS attacks with programmable switches,” in *Proceedings of the 30th {USENIX} Security Symposium*, pp. 3829–3846, Vancouver, BC, Canada, August 2021.

[88] J. Xing, W. Wu, and A. Chen, “Architecting programmable data plane defenses into the network with FastFlex,” in *Proceedings of the 18th ACM Workshop on Hot Topics in Networks*, Princeton, NJ, USA, November 2019.

[89] J. Hypolite, J. Sonchack, S. Hershkop, D. Nathan, A. DeHon, and M. S. Jonathan, “DeepMatch: practical deep packet inspection in the data plane using network processors,” in *Proceedings of the ACM Conference on emerging Networking Experiments and Technologies (CoNEXT)*, Barcelona, Spain, 2020.

[90] G. Li, M. Zhang, L. Chang et al., “NetHCF: enabling line-rate and adaptive spoofed IP traffic filtering,” in *Proceedings of the 2019 IEEE 27th International Conference on Network Protocols (ICNP)*, pp. 1–12, IEEE, Chicago, IL, USA, October 2019.

[91] J. Xing, K. Qiao, and A. Chen, “NetWarden: mitigating network covert channels while preserving performance,” in *Proceedings of the 29th {USENIX} Security Symposium*, San Diego, CA, USA, August 2020.

[92] J. Xie, F. Richard Yu, T. Huang, R. Xie, L. Jiang, and C. Wang, “A survey of machine learning techniques applied to software defined networking (SDN): research issues and challenges,” *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 393–430, 2018.

[93] Q. Qin, K. Poularakis, K. L. Kin, and L. Tassiulas, “Line-speed and scalable intrusion detection at the network edge via federated learning,” in *Proceedings of the IFIP-TC6 Networking Conference (Networking)*, Valencia, Spain, May 2020.

[94] D. Barradas, N. Santos, L. Rodrigues et al., “FlowLens enabling efficient flow classification for ML-based network security applications,” in *Proceedings of the Network and Distributed Systems Security (NDSS) Symposium*, San Diego, CA, USA, August 2021.

[95] S. A. Gutiérrez, J. W. Branch, L. P. Gaspary, and J. F. Botero, “Watching smartly from the bottom intrusion detection revamped through programmable networks and artificial intelligence,” 2021, https://arxiv.org/abs/2106.00239.

[96] A. Laraba, J. François, I. Chrisment, S. R. Chowdhury, and R. Boutaba, “Defeating protocol abuse with P4: application to explicit congestion notification,” in *Proceedings of the IFIP-TC6 Networking Conference (Networking)*, Valencia, Spain, May 2020.

[97] D. Scholz, H. Stubbe, S. Gallenmüller, and G. Carle, “Key properties of programmable data plane targets,” in *Proceedings of the 2020 32nd International Teletraffic Congress (ITC 32)*, pp. 114–122, 2020.

[98] Netronome Agilio CX SmartNICs, 2019, https://www.netronome.com/products/agilio-cx/.

[99] Pensando Distributed Services Architecture SmartNIC, https://www.servethehome.com/pensando-distributed-services-architecture-smartnic/.

[100] H. Wang, R. Soulé, H. Tu Dang et al., “P4FPGA: a rapid prototyping framework for P4,” in *Proceedings of the ACM Symposium on SDN Research (SOSR)*, Santa Clara, CA, USA, 2017.

[101] Tomahawk/BCM56960 Series, https://www.broadcom.com/products/ethernet-connectivity/switching/strataxgs/bcm56960-series.

[102] D. Scholz, A. Oeldemann, F. Geyer et al., “Cryptographic hashing in P4 data planes,” in *Proceedings of the 2019 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, pp. 1–6, IEEE, Cambridge, UK, September 2019.

[103] M. Valentin Dumitru, D. Dumitrescu, and C. Raiciu, “Can we exploit buggy P4 programs?” in *Proceedings of the Symposium on SDN Research*, San Jose, CA, USA, 2020.

[104] D. Scholz, S. Gallenmüller, H. Stubbe, B. Jaber, M. Rouhi, and G. Carle, “Me love (SYN-) cookies: SYN flood mitigation in programmable data planes,” 2020, https://arxiv.org/abs/2003.03221.

[105] S.-Y. Wang, C.-M. Wu, Y.-B. Lin, and C.-C. Huang, “High-speed data-plane packet aggregation and disaggregation by P4 switches,” *Journal of Network and Computer Applications*, vol. 142, 2019.

[106] R. Kumar, V. Babu, and D. M. Nicol, “Network coding for critical infrastructure networks,” in *Proceedings of the IEEE International Conference on Network Protocols (ICNP)*, Cambridge, UK, September 2018.

[107] P. Palagummi and K. M. Sivalingam, “SMARTHO: a network initiated handover in NG-RAN using P4-based switches,” in *Proceedings of the 2018 14th International Conference on Network and Service Management (CNSM)*, pp. 338–342, IEEE, Rome, Italy, November 2018.