Programmable Data Planes for Network Security

Gursimran Singh[1], H.B. Acharya[2], Minseok Kwon[1]

[1]Rochester Institute of Technology, [2]Oklahoma State University

Abstract. The emergence of programmable data planes, and particularly switches supporting the P4 language, has transformed network security by enabling customized, line-rate packet processing. These switches, originally intended for flexible forwarding, now play a broader role: detecting and mitigating attacks such as DDoS and spoofing, enforcing next-generation firewall policies, and even supporting in-network cryptography and machine learning. These capabilities are made possible by techniques such as recirculate-and-truncate and lookup-table precomputation, which work around architectural constraints like limited memory and restricted instruction sets.
In this paper, we systematize recent advances in security applications built on programmable switches, with an emphasis on the capabilities, challenges, and architectural workarounds. We highlight the non-obvious design techniques that make complex in-network security functions feasible despite the constraints of the hardware platform, and also comment on remaining issues and emerging research directions.

Keywords: Programmable data planes, P4, network security, DDoS mitigation, firewalls, machine learning, cryptography.

1 Introduction

Computer networks face increasingly sophisticated attacks, which require rapid and adaptive response. Conventional networking equipment, developed as a closed design, limits network operators to use facilities hard-coded by manufacturers. The first step in removing such limitations was the development of Software-Defined Networking (SDN), which centralizes control plane functions and allows for flexibility in controller-level applications (with a standard south-bound interface between controller and switch) [55]. The second step has been the rise of programmable network switches, which allow the operator to carry out more and more functions in the data plane, i.e. on the switch itself.

Programmable switches empower network administrators to implement customized packet processing functions, in-line in the switch. This has strong implications for speed and scale (as sending things to the controller almost inevitably leads to a bottleneck and prevents line-rate packet processing). Programming Protocol-independent Packet Processors (P4) has emerged as the *de facto* data plane programming language, enabling the deployment of sophisticated packet processing algorithms in network switches. This includes not only network management and telemetry, but also network security, making programmable switches a compelling platform for deploying in-fabric defenses. This paper surveys these emerging security applications.

1.1 P4 and Programmable Switches

P4 [13] is a domain-specific language that allows the network administrator (not the router manufacturer) to specify the structure of packets and control how these packets are processed by the data plane of network devices. Its aim was to allow the network admin to customize the processing pipeline for packets belonging to any free-form protocol (hence the name, programming protocol-independent packet processors). The Portable Switch Architecture (PSA) and Protocol Independent Switch Architecture (PISA) describe the capabilities of switches that can be programmed using P4, including a programmable parser, match-action pipeline, and deparser [71].

David C. Wyld et al. (Eds): CMLA, CSEIT, NLPD, GRAPH-HOC, SPM, NeTCoM, NCS, ArIT, DMSE – 2025
pp.163 -180, 2025.CS & IT - CSCP 2025 DOI: 10.5121/csit.2025.151314

Programmable switches, like all switches, operate on the match-action paradigm. A switch matches a field from a packet against values in a table, and the matched entry specifies corresponding actions to apply to the packet. These switches can operate at line rates (100 Gbps or higher), so their logic units execute simple operations only, primarily bit manipulation and basic arithmetic. The pipeline architecture consists of multiple stages, containing memory elements (SRAM, TCAM) and ALUs, and also a parser and deparser.

The main difference from other switches is that the parser, which extracts fields from the packet, is *programmable*. In other words, the end user can specify a schema for a new protocol, and the switch will slice up the packet according to this schema. This allows the flexibility to add support for new protocols at any time, and have it be supported by any switch that adheres to the P4 standard. Finally, the deparser stitches the packet back together after processing, so it can be forwarded to its destination, looped back, mirrored, etc.

A couple of intentional limitations in the architecture, for the sake of fast packet processing, cause the programmable switch to be limited as an "active network" node.

1. The parser is not truly a top-down parser, able to parse the packet as a general-purpose compiler parses code. It is much more limited and essentially fetches bit-slices of a given length starting from a given offset. So complex parsing (as required by say HTTP, with its optional, re-ordered, and variable-length fields) is not supported. In other words, the language offers sufficient flexibility for defining novel protocol headers and tags, but not enough for full in-network processing (for example, for query processing in the network fabric).
2. Complex operations, like floating-point operations or encryption, are not natively supported by standard P4-compliant architectures. This, again, limits the intelligence of the switch when the majority of traffic is encrypted, and the power of in-network computation.

We now come to the insight that motivated this paper. Programmable switches are quite limited as a platform for general computation, but are still a very attractive platform for network security applications. Most importantly, they can process packets extremely fast, on the order of 100x faster than a typical CPU [38]. They do have some instructions for programming, using state associated with packets (registers and monitors), though limited to operations that were thought to be necessary for protocol- and target-independent network operations. But in addition they have significant chip area for memory to support tables (early switches, like Reconfigurable Match Tables (RMT), allocated over 50% of its chip area for memory [14]) and such tables can be creatively used as look-up tables for computation, not just to match and forward packets.

This has led to two important research directions – the first, directly using such switches to provide security applications such as DDoS mitigation and firewalls; the other, "hacking" the architecture by using lookup tables for encryption [17] or the recirculate-and-truncate technique packets to achieve true parsing and pattern-matching [38], i.e. sidestepping the limitations mentioned above. Accordingly, we present a comprehensive study that covers both straightforward ("kosher") and workaround-based ("hacky") approaches for implementing network security applications in the data plane, as well as a discussion of the open challenges and research directions.

This paper has the following main contributions:

– We systematize programmable data plane security research across three major categories (**For in-network attack mitigation**, **As middleboxes and firewalls**, and **For in-network computations and ML-based security**), and analyzing fundamental trade-offs between programmability, performance, and security.

– We highlight the non-obvious design techniques (such as **recirculate-and-truncate** and **lookup-table pre-computation**) that make complex security functions feasible despite hardware constraints, and identify current challenges and future research directions in the field.

The rest of this paper is organized as follows. Section II explores programmable switches as "better switches", i.e. enhanced packet-forwarding devices and how they help with threat mitigation in networks. Section III expands on their role as security middleboxes, notably application-layer firewalls and cryptography. Section IV then discusses their emerging use as platforms for machine learning-based security. We then present some discussion, focusing on broad challenges and future directions, and conclude the paper.

2 Smart Switches: In-Network Attack Mitigation

In this section, we focus on the network security role of programmable switches that act simply as an updated, more powerful version of traditional switches. Even in this limited role, such switches can detect and mitigate many attacks, some of them at line rate (providing superior performance compared to traditional software-based solutions, and much lower costs than dedicated hardware). We first mention P4-based solutions to address two traditional network threats, spoofing and DDoS attacks, then continue on to discuss how these switches also help with hardening the network and making it quickly adaptable to threats.

2.1 Spoofing Protection

Spoofing attacks, where attackers impersonate legitimate devices, have always been a persistent threat in network environments, and traditional switches have defended against such attacks for decades. CloudFlare reports that even now, almost all large-scale layer-3 DDoS attacks rely on IP spoofing [53]. Several anti-spoofing mechanisms have been developed with programmable switches, essentially updating the tools found in traditional L2 switches (Port Security, DHCP snooping, Dynamic ARP Inspection, and IP Source Guard).

– **NetHCF**: Li et al. [47] propose NetHCF, which maintains a mapping table between IP addresses and hop counts (IP-to-Hop-Count or IP2HC), allowing it to infer spoofed traffic by comparing hop counts in arriving packets with expected values from legitimate sources. The IP2HC table is stored in-controller with the most active IP entries cached in the data plane using registers. aggregating nodes with common IP prefixes and hop-count values.
– **DroPPPP**: Simsek et al. [66] develop DroPPPP, which detects spoofed addresses by calculating a hash of the source IP and MAC addresses, then comparing it against previously stored hash values. DroPPPP uses three registers (hash values, timestamps, attack flag) and identifies and blocks spoofed packets at line rate.
– **P4DAD**: Kuang et al. [42] proposed P4DAD to secure Duplicate Address Detection. P4DAD stores bindings between IPv6 addresses and ports in the switch (much like a DHCP binding table), and verifies the authenticity of Neighbor Advertisement messages, preventing attackers from disrupting the address configuration process.
– Other implementations, such as those by Narayanan et al. [58] and Gondaliya et al. [30], have explored additional anti-spoofing techniques including Network Ingress Filtering (NIF), Reverse Path Forwarding (RPF), Spoofing Prevention Method (SPM), and Source Address Validation Improvement (SAVI) solution for DHCP.

P4-based anti-spoofing mechanisms offer much lower latency and full network-wide visibility of all traffic, compared to host-based approaches. They also do not incur the costs and delays of additional hardware or modifications to routers, as the required modifications to the switch behavior are made through dataplane programming (not hardware-level changes).

2.2 DDoS Attack Mitigation

A second traditional area of network security involves Distributed Denial-of-Service attacks. Programmable switches have been used to develop robust defenses that balance two goals: high throughput and accuracy.

P4-based algorithms were quickly developed to detect heavy hitters (traffic flows of large volume), which often constitute DDoS attacks. For example, **HashPipe** [68] identifies the k-heaviest flows with high accuracy entirely in the data plane using a pipeline of hash tables. **PRECISION** [11] uses probabilistic recirculation to monitor a small fraction of packets for a second pipeline traversal, balancing memory constraints with accuracy. **MV-Sketch** [69] implements an invertible sketch that applies majority vote algorithms to find candidate heavy flows. And **Elastic Trie** [43] targets hierarchical heavy hitters, changes in traffic patterns, and superspreaders, which can be used for DDoS, anomaly, worm, and spam detection.

Subsequently, P4-based systems have been built to detect and respond to such attacks.

- **POSEIDON** [79]: Provides network operators with an expressive policy language to specify DDoS attack mitigation strategies. POSEIDON partitions functions between P4 switches and general-purpose servers, adapting to dynamic attacks by generating new defense policies and reconfiguring switches.
- **Jaqen** [52]: Implements switch-native DDoS detection and mitigation in the data plane, covering a wide range of volumetric attacks within ISPs, and provides APIs for querying metrics and flexible mitigation.
- **DDoSD-P4** [44]: Implements a fully in-network inspection mechanism with three stages: entropy estimation, traffic characterization, and anomaly detection. **EUCLID** [21] extends DDoSD-P4 to enable scalable detection and mitigation of volumetric DDoS attacks in the data plane.
- **INDDoS** [23]: Employs a novel data structure called BACON (combining Bitmaps and Count-Min Sketch) to estimate per-destination flow cardinality, i.e. they identify hosts to which a large number of flows are targeted as likely DDoS victims.

There also exist implementations that focus on specific protocols and attack vectors commonly used in DDoS.

- **DNS Amplification Defense**: Khooi et al. [40] proposed DIDA, a distributed in-network defense architecture against DNS amplification attacks, monitoring both requests and responses across the network.
- **SIP DDoS Defense**: Febro et al. [27] detect DDoS attacks using the Session Initiation Protocol by monitoring INVITE and REGISTER packets at the first-hop switch.

These DDoS mitigation techniques are not only much faster than software-based solutions (or even other SDN/NFV defenses that rely on the control plane), but unlike traditional traffic scrubbing centers that use expensive proprietary hardware (say $ 1 M – $10 M), they combine high performance and simple reconfigurability at a very affordable price ($5 k – $10 k for an industrial-scale switch).

3 Smart Switches as Middleboxes and Firewalls

Programmable switches not only act as enhanced switches for in-line threat mitigation, they also increasingly serve the functions previously performed by full-featured middleboxes. In this section we examine systems that build such switches into firewalls (including application-layer firewalls) or cryptographic endpoints (say, for a VPN) directly in the network fabric.

3.1 Firewalls and Access Control

Firewalls control traffic flow between networks based on predetermined security rules, either filtering in the network layer (typically using the header fields: source IP, source port, destination IP, destination port, and protocol) or in the application layer (Deep Packet Inspection – for example, URL filtering). Programmable data planes offer new opportunities for implementing high-performance, flexible firewall functionality directly in network hardware, as seen in the following systems:

- **P4Guard**: Datta et al. [22] developed P4Guard, a configurable firewall using P4. The authors define a high-level, hardware-independent security policy language; the controller translates these policies written in the language, into match-action table rules that are deployed to the switch. P4Guard implements tables for firewall rules, based on IP, TCP, UDP, and ARP header fields, and maintains a counter table to record statistical flows, periodically sending this information to the controller for analysis.
- **CoFilter**: Cao et al. [16] proposed a stateful packet filter implemented on a switch. CoFilter maintains flow state by computing a hash of the 5-tuple and storing it as a flow ID (fid), enabling stateful filtering with limited memory. CoFilter uses a management server to update flow tables, to avoid hash collisions over time.
- **Stateful Firewalls**: Li et al. [48] implemented a stateful firewall for cloud environments using P4. Their approach leverages the expressiveness of P4 to parse packets and filter them based on firewall policies while tracking TCP connections to allow only packets belonging to established sessions.
- **Authentication and Port Knocking**: "port knocking" is a covert channel where a signal is sent by sending requests to a specific sequence of ports. Almaini et al. [7] implemented port knocking on programmable switches as an authentication mechanism. The system stores authenticated and unauthenticated connections using match-action table rules inserted by the controller. In a follow-up work [6], they add One-time Password (OTP) authentication to protect against replay attacks.
  Zaballa et al. [77] presented multiple implementations of port knocking, ranging from pure data-plane implementations (using registers to store knocking states), to hybrid approaches that delegate some functions to the controller.
- **Mobile Network Firewalls**: Ricart-Sanchez et al. [61] developed a firewall for 5G mobile network infrastructure, positioned between edge and core networks. Their design was later extended [62] to support multi-tenant 5G infrastructures.

P4-based firewalls offer several advantages over traditional header-based (network-layer) firewalls. They are just as fast as a hardware firewall, and rapidly adapt to new threats without hardware upgrades, unlike traditional fixed-function firewalls. More importantly, however, the flexibility of a programmable dataplane allows for the development of more general security systems, combining network security and management. We mention a few important systems below.

- **NetWarden**: Xing et al. [74] created a defense mechanism that preserves TCP performance while mitigating covert storage and timing channels. NetWarden performs header inspection and modification on fields commonly used in covert channels (e.g., TTL, TCP reserved fields). For advanced storage channel protection, it replaces headers with newly generated ones and stores mappings between original and new headers in stateful memory.
- **Poise**: Kang et al. [39] developed a system for context-aware security policies, particularly for Bring Your Own Device (BYOD) environments. Poise translates high-level policies into P4 programs that can enforce dynamic access control based on device runtime context (e.g., operating system version).
- **FastFlex**: Xing et al. [75] created an architectural framework supporting various defense mechanisms. Their system transforms defense applications ("boosters") into Packet Processing Modules that can be optimally mapped to network resources, enabling dynamic scaling during attacks.
- **Stateful Security Monitoring**: Laraba et al. [45] modeled security monitoring functions as Extended Finite State Machines (EFSM) in P4. This approach enables detection of protocol abuse, such as ECN protocol misuse and optimistic ACK attacks, by mapping protocol states and transitions to P4 primitives.
- **5G Network Slice Protection**: Bonfim et al. [12] developed FrameRTP4, a system for detecting and mitigating attacks in 5G network slices in real-time, using data structures like Bloom Filters, Count-Min Sketch, and Invertible Bloom Lookup Tables for efficient monitoring.

These security mechanisms demonstrate the versatility of programmable data planes; quite complicated logic can be implemented using the limited resources on a switch, in conjunction with proper algorithms and data structures. However the question remains whether pure dataplane programming is powerful enough to perform next-generation firewalling, i.e. Deep Packet Inspection. Surprisingly, it can.

The key breakthrough enabling string and regex matching in the data plane, despite the parser's limitations, is a technique known as recirculate-and-truncate. Since the P4 parser is single-pass and cannot loop to parse arbitrary data (as seen in packet payloads!) this approach instead recirculates a packet through the switch pipeline multiple times. On each pass, the first byte of the payload is removed for inspection, allowing the switch to effectively simulate a sliding-window traversal of the packet one byte at a time.

By coupling this mechanism with a deterministic finite automaton (DFA) implemented in the match-action tables, the switch can track pattern matches across passes. Each stage in the pipeline corresponds to a transition in the DFA, enabling deep packet inspection at line rate, despite the absence of loops or complex parsing logic. This technique powers systems such as PPS, DeeP4R, and BOLT.

- **PPS** Jepsen et al.'s PPS (P4 Programmable String search) system [38] was the first paper to match strings using a deterministic finite automaton (DFA) mapped to the match-action pipeline of a PISA switch. PPS stored the DFA transitions into multiple match stages in the pipeline, each performing a single-character match and state transition. The DFA is stored in a series of ternary match tables, and each table maps an input character and current state to a next state. This mapping is highly efficient (due to the fixed pipeline architecture of PISA), though the memory overhead grows with the number of DFA states and input symbols. PPS supports thousands of keyword patterns and achieves line-rate processing at 10–100 Gbps, significantly outperforming conventional CPU- and FPGA-based DPI implementations in latency and throughput.

- **DeeP4R** Gupta et al. extend this idea with DeeP4R [31], a full next-gen firewall in P4. DeeP4R implements Deep Packet Inspection by recirculating a cloned copy of the packet through the switch pipeline. On each iteration, the first byte of the clone packet is cut off for a DFA-based pattern match. The original remains untouched, and is forwarded if the clone matches no known patterns in the firewall (otherwise, it is discarded). Thus P4's `clone` and `recirculate` primitives are used to achieve byte-wise inspection, with as many passes as needed. DeeP4R handles thousands of patterns with limited register and pipeline resource usage, and can enforce layer-7 security policies (e.g., URL blacklists) with low overhead in real-time traffic.
- **BOLT** Large rule sets or complex pattern collections can be too costly to match (DFA state explosion). BOLT [78] achieves more scalable multi-pattern string matching, using two optimizations. BOLT reduces DFA transitions using k-stride (multiple input bytes are consumed at a step) to reduce pipeline depth, and also reduces the size of the DFA using bit slicing to compactly represent the transition logic. This achieves a balance between memory usage and matching speed, and allows for the best operation within the strict memory and stage constraints of real switches.

3.2 Cryptographic Primitives in the Data Plane

The first challenge when considering secure connections in the data plane, is what secure operations can be supported on a programmable switch. Despite the limited instruction set of P4, several research efforts have successfully implemented cryptographic functions:

- **AES Encryption**: Chen [17] proposed "scrambled lookup tables" to implement AES encryption entirely in the data plane without controller interaction. The approach supports all three variants of AES (AES-128, AES-192, and AES-256) and leverages packet recirculation to simulate the multiple rounds required by the algorithm. Evaluations showed throughput of 10.92, 8.76, and 7.37 Gbps for AES-128, AES-192, and AES-256, respectively.
- **Diffie-Helman with AES**: Oliveira et al [59] build on the above system to include Diffie-Hellman key exchange, so the smart switch is not only performing encryption but also key setup.
- **Content Permutation**: Lin et al. [50] developed a secret permutation mechanism for P4 switches to protect 5G packet payloads. The algorithm partitions the payload into codewords and shuffles them across multiple stages of the ingress pipeline, achieving permutation at line rate.

In addition to the above "pure" implementations, some noteworthy implementations have made use of non-standard hardware.

- **Cryptographic Hash Functions**: Scholz et al. [63] extended multiple P4 targets to support cryptographic hash functions, including SipHash-2-4, Poly1305-AES, BLAKE2b, HMAC-SHA256, and HMAC-SHA512. Different implementations were tested on various targets including CPU-based (t4p4s), NPU-based, and FPGA-based platforms. CPU targets were easily extensible but had higher latency; NPU offered the highest throughput; and FPGA targets provided the lowest latency.
- **FPGA-accelerated Cryptography**: Malina et al. [54] accelerated cryptographic operations for FPGA-based network cards and integrated them as P4 externs. They implemented symmetric ciphers (AES-GCM-256), digital signatures (EdDSA), and hash functions (SHA3) using VHDL, achieving 26.24 Gbps for cryptographic functions and 4.51 Gbps for hash functions.

3.3 Secure Protocols in the Data Plane

Besides cryptographic primitives, researchers have implemented complete security protocols on programmable switches. We note that so far none of these were implemented purely in the data plane.

- **P4-MACsec**: Hauser et al. [34] implemented Media Access Control security (MACsec) using P4. MACsec provides point-to-point security between enabled endpoints within a local area network. Their implementation features a two-tier control plane, with local controllers handling time-sensitive operations and a central controller for coordination. For encryption and authentication, P4-MACsec uses AES in Galois/Counter Mode (AES-GCM) implemented as P4 externs.
- **P4-IPsec**: The same group [33] further implement Internet Protocol Security (IPsec), specifically the Encapsulating Security Payload (ESP) protocol in tunnel mode, with AES Counter Mode (AES-CTR) and NULL ciphers. Instead of using the Internet Key Exchange (IKE) protocol, P4-IPsec relies on an SDN controller for Security Association (SA) management, reducing message exchanges. The crypto manager is implemented either on the switch's CPU via PCIe or on an external crypto host.
- **P4NIS**: Li et al. [51] proposed a Network Immune Scheme against eavesdropping attacks in IoT networks. P4NIS implements multiple defense layers: traffic distribution across different network paths to confuse eavesdroppers, transport layer header encryption, and packet payload encryption. While the data plane handles traffic distribution and some encryption, complex encryption operations are delegated to the controller or IoT devices.

In other words, the restricted instruction set of P4 has made it impractical to implement cryptographic systems purely in the data plane with standard switches. Researchers have addressed this by using external resources (CPU, controller), and this is a limitation creating potential bottlenecks and security vulnerabilities in the communication channel. Also the actual encryption is handled using dedicated hardware and not a standard switch. We therefore identify an important research gap: to build on the early work of Chen [17] and create a complete security solution (such as an IPSec endpoint).

We note that there is a tension between the amount of on-chip memory needed for the look-up tables and the security of the solution (more complex cryptographic algorithms will need more memory); similarly, stronger security typically requires more complex algorithms and will reduce throughput. Another concern is whether we allow controller dependence for operations such as key management. Despite these challenges and caveats, a successful implementation of cryptography directly on standard switches would offer significant advantages for network security, enabling high-speed encryption and authentication without dedicated security appliances.

4 Smart Switches, In-Network Computation and Machine Learning

In the previous sections, we considered the use of programmable switches in their role as switches and as middleboxes for network security. However, the increasing sophistication of cyber threats and the growing complexity of modern networks have made traditional rule-based security approaches insufficient. This section focuses on Machine Learning-based security applications within programmable networks, and on how it is possible to do at least some limited ML inference in the data plane.

ML models deployed in programmable user planes have been effective in several security domains, including DDoS and intrusion detection. More comprehensively,

- **Intrusion Detection Systems (IDS):** ML-based IDS implemented in the user plane, like Planter [81] and Soter [73], identify malicious traffic by analyzing packet-level features directly in the switch pipeline. Compared to traditional control-plane IDS, these user-plane approaches reduce detection latency from milliseconds to nanoseconds, significantly limiting the impact of attacks.
- **DDoS Attack Mitigation:** Tree-based models embedded in programmable switches have proven effective for rapid Distributed Denial of Service (DDoS) detection. BACKORDERS [19] implements Random Forest (RF) models in the programmable data plane to identify DDoS attacks with high accuracy. Similarly, pHeavy [80] enables detection of heavy flows characteristic of volumetric attacks through stateful flow-level monitoring in the switch.
- **Encrypted Traffic Analysis:** Recent advancements enable security analysis of encrypted traffic without breaking encryption. Flowrest-based approaches [3] for encrypted traffic classification use statistics of packet sizes and inter-arrival times to identify potentially malicious encrypted communications at line rate, achieving accuracy levels around 87-95% across various encrypted application classification tasks.
- **Smart Grid Protection:** Ultra-low latency cyberattack detection for Smart Grid (SG) systems has been demonstrated using Decision Tree (DT) models embedded in programmable switches [5]. This approach achieves 99% accuracy in detecting attacks against Distributed Network Protocol 3 (DNP3) for industrial control systems, with sub-microsecond latency – 17,000 times faster than control-plane solutions.
- **Botnet and Malware Detection:** In-Network Classification (INC) [29] enables line-rate detection of botnet propagation in programmable switches. Similarly, Friday et al. [28] proposed a methodology for ransomware detection in P4 switches. These approaches typically identify command-and-control communication patterns and anomalous data exfiltration characteristic of malware.

4.1 Architectural Approaches to ML-based Security

Machine learning models in programmable networks can operate at three levels: per-packet, per-flow, or with joint inference that combines both. Each inference strategy has distinct tradeoffs in terms of latency, resource usage, and detection power.

Packet-Level Inference Packet-level (PL) inference extracts features from individual packets in isolation, typically focusing on header fields. This design enables rapid threat detection without requiring flow state.

- **Comprehensive Coverage:** Every packet is inspected and classified at line rate, ensuring that malicious traffic is not missed—crucial for detecting zero-day exploits or malformed headers.
- **Low Latency:** Detection occurs immediately from the first packet, avoiding delays associated with flow aggregation.
- **Minimal State Overhead:** No per-flow state is required, conserving limited memory resources.

Several systems have demonstrated the feasibility and effectiveness of PL inference in P4 switches, including IIsy [76], Planter [81], Mousika [72], and Soter [73].

However, this approach cannot capture behavioral patterns that span multiple packets. Additionally, matching against a large number of patterns for every packet can create computational bottlenecks in constrained environments.

To address these challenges, Henna [1] introduced hierarchical packet-level inference. By decomposing complex classification tasks into simpler, cascaded subtasks, Henna achieved a 21% F1-score improvement on a 21-category IoT identification task while maintaining efficient resource usage.

Flow-Level Inference Flow-level (FL) inference aggregates features across multiple packets within a flow, allowing richer behavioral analysis and more accurate threat detection.

- **Behavioral Visibility:** Enables detection of attacks that unfold over time, such as data exfiltration, command-and-control, or covert channel activity.
- **Reduced Redundancy:** Classifying flows, rather than every packet, reduces repetitive processing.

Notable frameworks that support FL inference in programmable switches include Flowrest [4], FlowLens [10], pForest [15], and SwitchTree [46].

Despite their advantages, FL approaches introduce new challenges:

- **Delayed Classification:** Threats can go undetected during the early phase of a flow, before enough data is available.
- **State Exhaustion Risk:** Maintaining per-flow statistics can deplete limited memory resources and may be abused by attackers to launch state exhaustion attacks.

Combined Inference Joint inference approaches seek to combine the responsiveness of PL models with the context-awareness of FL models.

Sequential Multi-Model Approaches: NetBeacon [82] uses a series of decision tree classifiers applied at different points in the packet lifecycle. Early packets are classified using packet-level features, while later packets are matched using accumulated flow-level context. This architecture improves coverage and accuracy but increases memory and pipeline complexity.

Unified Model Approaches: Jewel [2] adopts a single random forest model trained on both PL and FL features. It uses dynamic feature weighting during training to adjust inference depending on packet position within a flow. Jewel outperforms both packet-only and flow-only models, achieving an average accuracy improvement of 3.2%, while maintaining lower resource usage than NetBeacon's multi-model pipeline.

4.2 Advantage: Performance and Accuracy

A standard issue in conventional security architectures is the delay in processing and communication:

1. **Control-Plane ML:** Traditional SDN security approaches where ML models run in the controller induce delays of 1-100+ milliseconds due to control-user plane communication overhead [35].
2. **External Security Appliances:** Traffic redirection to dedicated security appliances can add 1-10+ milliseconds of latency, creating a substantial window of vulnerability.
3. **Hybrid Approaches:** Even solutions that distribute tasks between planes, using the user plane for feature extraction and the control plane for inference [10, 9, 8, 64], still incur millisecond-level delays.

In contrast, user-plane ML can enable detection and mitigation at line rate, and experimental studies of in-switch security solutions demonstrate dramatic speedups compared to conventional approaches.

- **Smart Grid Security:** A DT model for attack detection in Smart Grid systems achieved a detection latency of just 356 nanoseconds – 6,000 times faster than solutions using user-plane feature extraction with control-plane inference, and 17,000 times faster than full control-plane approaches [5].
- **DDoS Detection:** In-switch models for DDoS detection can identify and mitigate attacks within 400-500 nanoseconds, compared to several milliseconds for control-plane solutions [19, 80].
- **Flow-Based Attacks:** Flowrest demonstrated detection of malicious flows with packet-processing latency below 450 nanoseconds across multiple security-relevant datasets [4].

This ultra-low latency detection can enable early responses from the IDS or firewall (as mentioned in the previous section), potentially preventing attacks from ever reaching their targets rather than merely limiting their impact after the attack.

4.3 Challenges: Resource Constraints

Implementing ML-based security in the data plane presents familiar technical challenges, owing to the limitations of the platform: the limited range of operations available, and the limited memory to hold look-up tables.

- **Feature Availability:** While comprehensive feature sets improve security detection accuracy, many desirable security features are challenging to implement in switches. As discussed in the previous section, payload inspection is not natively supported in P4, so if we want the ability to detect application-layer attacks we have to use complex and slow workarounds (recirculate-and-truncate). Similarly, complex temporal patterns used in security analytics are difficult to capture within switch constraints.
- **Resource Constraints vs. Model Complexity:** Security tasks can require complex models to detect sophisticated threats. However, switch hardware is severely resource-constrained. Commercial switches typically have only a few hundred megabytes of Static Random-Access Memory (SRAM) and a few megabytes of expensive Ternary Content-Addressable Memory (TCAM), shared between normal forwarding functions and security models.

This issue is particularly problematic when we consider that unlike general classification tasks, security models face adversaries actively attempting to bypass them. Robust and fault-tolerant defenses are particularly challenging to implement in a resource-constrained platform (for example, consider the poor performance of a neural network when heavily quantized).

One particular challenge is **zero-day detection**. Intrusion detection etc. using supervised ML, is not strong in identifying previously unseen attacks. Detecting novel attacks requires models capable of detecting anomalies rather than just known patterns, but a learning approach that can deal with such unknown patterns will likely require unsupervised or semi-supervised learning with substantial data; such models are hard to fit into a constrained environment.

4.4 Model Selection for Security Applications

As programmable switches are resource-constrained, we require careful model selection for security applications. Comparative evaluation across multiple security-relevant datasets [4]

so far indicates that traditional machine learning, in the form of Decision Tree (DT) and Random Forest (RF) models, outperforms more complex alternatives like Neural Networks (NN) and Binarized Neural Networks (BNN) for in-switch security tasks. Across intrusion detection datasets including CICIDS-2017 [65], UNSW-NB15 [56, 57], and NSL-KDD [70], RF models achieved F1-scores typically in the 95-100 range, compared to 84-91 for BNNs.

This performance advantage stems from several factors:

1. **Tabular Data Efficiency:** Security-relevant network features typically form tabular data, where tree-based models often outperform deep learning approaches [32].
2. **Structural Compatibility:** The logical structure of tree-based models aligns naturally with the Match & Action (M/A) pipeline architecture of programmable switches.
3. **Resource Efficiency:** Tree-based models require primarily simple comparison operations that map efficiently to switch hardware, whereas even simplified neural networks like BNNs can quickly exhaust switch resources.

For example, a basic BNN with just two layers (64 and 32 neurons) completely exhausts the resources of a Tofino ASIC [67], but still underperforms simpler tree-based models on security classification tasks. And of course, tree-based models are far superior in another regard: they are explainable, a factor critical for security applications where the rationale for a response must be justified.

4.5 Future Directions

The most direct research direction involving ML-based security in programmable networks, would be to study distillation and methods to capture powerful models on a resource-constrained platform. For example, Trustee [37] approximates a neural network with a best-fit decision tree. The challenge is to get models small enough to fit on a switch, but good enough to compete with machine learning models on the controller working in concert with the switch [25].

We note that future network security solutions could certainly benefit from hardware designed specifically to support ML-based security functions, potentially through specialized computing units integrated into programmable switches (extern logic, as seen in DeepMatch [36] for example). But the cost for this would be the loss of compatibility with standard hardware, and potentially the loss of real-time line-rate decisions. In the immediate future we are more hopeful that innovation on the software side, with possibly some limited support from hardware, will make it possible to build more robust high-performance models that still fit within switch resource constraints. Further open research problems include:

– **Unsupervised Anomaly Detection:** Current in-switch implementations focus primarily on supervised learning for known attack classes. Genos [49] represents an initial step toward unsupervised approaches, but more research is needed to enable effective zero-day attack detection within the constraints of a programmable switch.
– **Cross-Device Coordination:** We note that programmable switches have the advantage of always being part of a network. In other words, they may be computationally poor, but are rich in communication bandwidth. Distributing the load of security inference across multiple network devices could enable more comprehensive threat detection without requiring stronger computational power from individual devices. In this special case of in-network multi-agent computation, the open problem is developing efficient coordination mechanisms for cooperative inference across devices.

– **Adaptive Security Models:** Mechanisms for efficiently updating in-switch security models in response to emerging threats, without disrupting network operations, would improve responsiveness to evolving attack landscapes. The open problem here is to resist catastrophic forgetting when exposed to new challenges.

5 Discussion - Current Challenges

Programmable switches offer significant advantages for network security, but also significant challenges as a platform. As previously noted, switches can only perform a limited number of operations per packet (owing to the need to process them in real time), and the limited instruction set (no native division or floating-point operations) makes it extremely challenging to implement the algorithms used in ML inference or cryptography. Several approaches have been tried to work around this constraint:

– **Approximation**: Ding et al. [24] proposed P4Log and P4Exp algorithms to estimate logarithms and exponential functions using only P4-supported arithmetic operations, enabling entropy calculation entirely in the data plane.
– **Precomputation**: Values are precomputed and stored in match-action tables or registers, converting complex calculations into simple lookups.
– **Binarized Models**: Qin et al. [60] implemented Binarized Neural Networks in P4, converting complex ML operations into simpler bitwise operations supported by the switch.

But as noted in the previous sections, these approaches are not universally successful. And supplementing with external computation – for instance, Hauser et al. [33] offloading IPsec crypto manager functions: to the internal CPU module connected via PCIe, or to an external crypto host – compromises the performance gain from in-network computation.

The other main challenge is the limited on-switch memory available for tables and metadata, which can be as low as tens of megabytes for some switches [26]. This constrains the number of flows or packets that can be tracked, the mappings for traffic label switching or re-mapping (say, for anonymization), the state information stored, and the look-up tables used to work around the limited native operations. Attempts to reduce the memory usage include:

– **Data Structures.** Data structures like Count-Min Sketch [20], Bloom Filters, and multi-stage hash tables help maximize the use of limited memory. For instance, Chen et al. [18] stored TCP packet records in a hash table using memory arrays across pipeline stages.
– **Compressed Representations.** FlowLens [10] uses a compact representation of packet distributions (flow markers) to track network behavior with minimal memory usage. Similarly, BOLT [78] uses optimized representation to compress automata for regular-expression matching.

One possible direction of future work could focus on increased cooperation between switches, or switches and servers, as agents performing distributed computing. For instance, Kim et al. [41] suggest accessing DRAM on data center servers through RDMA-capable NICs, allowing switches to utilize external memory with minimal latency impact (1-2 microseconds). Smart allocation of memory resources, with a hierarchy of access similar to cache/ memory/ disk, would be a natural architecture for in-fabric network computation.

6 Conclusion

This survey has shown that despite tight resource constraints, programmable switches can now serve as platforms for an unexpectedly wide range of high-performance security applications. From DDoS mitigation and firewalls to cryptographic primitives and machine learning inference, the network fabric itself is becoming an active layer of defense. With this evolution, programmable switches are no longer just fast routers or packet filters, but essential building blocks for secure, autonomous, and adaptable networks.

The advantage of this new platform is clear: switches can respond orders of magnitude faster than traditional software-based solutions, and are also flexible enough to adapt to evolving threats without hardware replacement. Their key advantages are line-rate processing, visibility into network behavior, and agility in responding to new threats. But they have their own challenges: memory constraints, limited processing capabilities, deep packet inspection limitations, and (if we rely on extensions in the hardware) interoperability concerns.

Our aim in this paper is not only to survey the field, but also to draw attention to innovative approaches that address these challenges, from efficient data structures and approximation techniques to recirculate-and-truncate and look-up-table precomputation. With continuing advances in data structures, inference methods, and switch architectures, the prospect of self-defending, self-healing and self-managing networks is fast becoming a reality. By systematizing the state of the art, the current limitations, and current and possible future techniques for overcoming them, we aim to contribute to the realization of this vision.

References

1. Aristide T.-J. Akem, Beyza Bütün, Michele Gucciardo, and Marco Fiore. Henna: hierarchical machine learning inference in programmable switches. In *Proceedings of the 1st International Workshop on Native Network Intelligence*, pages 1–7, 2022.
2. Aristide T.-J. Akem, Beyza Bütün, Michele Gucciardo, and Marco Fiore. Jewel: Resource-efficient joint packet and flow level inference in programmable switches. In *IEEE INFOCOM 2024 - IEEE Conference on Computer Communications*, 2024.
3. Aristide T.-J. Akem, Guillaume Fraysse, and Marco Fiore. Encrypted traffic classification at line rate in programmable switches with machine learning. In *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, pages 1–9, 2024.
4. Aristide T.-J. Akem, Michele Gucciardo, and Marco Fiore. Flowrest: Practical flow-level inference in programmable switches with random forests. In *IEEE INFOCOM 2023 - IEEE Conference on Computer Communications*, pages 1–10, 2023.
5. Aristide T.-J. Akem, Michele Gucciardo, and Marco Fiore. Ultra-low latency user-plane cyberattack detection in sdn-based smart grids. In *Proceedings of the 15th ACM International Conference on Future and Sustainable Energy Systems*, e-Energy '24, pages 676–682, 2024.
6. A Almaini, A Al-Dubai, I Romdhani, M Schramm, and A Alsarhan. Lightweight edge authentication for software defined networks. *Computing*, 103(2):291–311, 2021.
7. Abdulsalam Almaini, Ahmed Al-Dubai, Imed Romdhani, and Martin Schramm. Delegation of authentication to the data plane in software-defined networks. In *2019 IEEE International Conferences on Ubiquitous Computing & Communications (IUCC) and Data Science and Computational Intelligence (DSCI) and Smart Computing, Networking and Services (SmartCNS)*, pages 58–65. IEEE, 2019.
8. João R. Amado, Fernando Pereira, Diogo Pissarra, Salvatore Signorello, Miguel Correia, and Fernando M. V. Ramos. Peregrine: Ml-based malicious traffic detection for terabit networks. *arXiv preprint arXiv:2403.18788*, 2024.
9. João Romeiras Amado, Fernando C. Pereira, Salvatore Signorello, Miguel Correia, and Fernando Ramos. Poster: In-network ml feature computation for malicious traffic detection. In *Proceedings of the ACM SIGCOMM 2023 Conference*, pages 1105–1107, 2023.
10. Diogo Barradas, Nuno Santos, Luís Rodrigues, Salvatore Signorello, Fernando M Ramos, and André Madeira. Flowlens: Enabling efficient flow classification for ml-based network security applications. In *Proceedings of the 28th Network and Distributed System Security Symposium*, 2021.

11. Ran Ben-Basat, Xiaoqi Chen, Gil Einziger, and Ori Rottenstreich. Efficient measurement on programmable switches using probabilistic recirculation. In *2018 IEEE 26th International Conference on Network Protocols (ICNP)*, pages 313–323. IEEE, 2018.
12. Marielly Bonfim, Maicon Santos, Kelvin Dias, and Sydney Fernandes. A real-time attack defense framework for 5g network slicing. *Software: Practice and Experience*, 50(10):1756–1775, 2020.
13. Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, et al. P4: Programming protocol-independent packet processors. *ACM SIGCOMM Computer Communication Review*, 44(3):87–95, 2014.
14. Pat Bosshart, Glen Gibb, Hun-Seok Kim, George Varghese, Nick McKeown, Martin Izzard, Fernando Mujica, and Mark Horowitz. Forwarding metamorphosis: Fast programmable match-action processing in hardware for sdn. In *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*, pages 99–110. ACM, 2013.
15. Christoph Busse-Grawitz, Roland Meier, Alexander Dietmüller, Tobias Bühler, and Laurent Vanbever. pforest: In-network inference with random forests. *arXiv preprint arXiv:1909.05680v2*, 2019.
16. Jiasong Cao, Jun Bi, Yu Zhou, and Cheng Zhang. Cofilter: A high-performance switch-assisted stateful packet filter. In *Proceedings of the ACM SIGCOMM 2018 Conference on Posters and Demos*, pages 9–11, 2018.
17. Xiaoqi Chen. Implementing aes encryption on programmable switches via scrambled lookup tables. In *Proceedings of the Workshop on Secure Programmable Network Infrastructure*, pages 8–14, 2020.
18. Xiaoqi Chen, Hyojoon Kim, Javed M Aman, Willie Chang, Ming Lee, and Jennifer Rexford. Measuring tcp round-trip time in the data plane. In *Proceedings of the Workshop on Secure Programmable Network Infrastructure*, pages 35–41, 2020.
19. Bruno Coelho and Alberto Schaeffer-Filho. Backorders: Using random forests to detect ddos attacks in programmable data planes. In *EuroP4 '22*, pages 1–7. ACM, 2022.
20. Graham Cormode and Sairam Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75, 2005.
21. Augusto da Silveira Ilha, Ângelo Cardoso Lapolli, Jonatas Adilson Marques, and Luciano Paschoal Gaspary. Euclid: A fully in-network, p4-based approach for real-time ddos attack detection and mitigation. *IEEE Transactions on Network and Service Management*, 2020.
22. Rupam Datta, Sangheon Choi, Ankur Chowdhary, and Younghee Park. P4guard: Designing p4 based firewall. In *MILCOM 2018-2018 IEEE Military Communications Conference (MILCOM)*, pages 1–6. IEEE, 2018.
23. Daniele Ding, Marco Savi, Federico Pederzolli, Mauro Campanella, and Domenico Siracusa. In-network volumetric ddos victim identification using programmable commodity switches. *IEEE Transactions on Network and Service Management*, 2021.
24. Daniele Ding, Marco Savi, and Domenico Siracusa. Estimating logarithmic and exponential functions to track network traffic entropy in p4. In *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*, pages 1–9. IEEE, 2020.
25. Roberto Doriguzzi-Corin, Luis Augusto Dias Knob, Luca Mendozzi, Domenico Siracusa, and Marco Savi. Introducing packet-level analysis in programmable data planes to advance network intrusion detection. *Computer Networks*, 239:110162, 2024.
26. Edge Networks. Wedge 100bf-32x 100gbe data center switch, 2025.
27. Andreia Febro, Huawei Xiao, and Jonathan Spring. Distributed sip ddos defense with p4. In *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–8. IEEE, 2019.
28. Karim Friday, Elias Bou-Harb, and Jorge Crichigno. A learning methodology for line-rate ransomware mitigation with p4 switches. In *NSS*, 2022.
29. Karim Friday, Elias Kfoury, Elias Bou-Harb, and Jorge Crichigno. Inc: In-network classification of botnet propagation at line rate. In *Computer Security – ESORICS*, 2022.
30. Hardik Gondaliya, G Chandra Sankaran, and Krishna M Sivalingam. Comparative evaluation of ip address anti-spoofing mechanisms using a p4/netfpga-based switch. In *Proceedings of the 3rd P4 Workshop in Europe*, pages 1–6, 2020.
31. Devashish Gosain et al. Deep4r: Deep packet inspection using recirculation in programmable data planes. In *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, 2023.
32. Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on typical tabular data? In *NeurIPS 2022*, 2022.
33. Florian Hauser, Michael Häberle, Marco Schmidt, and Michael Menth. P4-ipsec: Implementation of ipsec gateways in p4 with sdn control for host-to-site scenarios. *arXiv preprint arXiv:1907.03593*, 2019.
34. Florian Hauser, Marco Schmidt, Michael Häberle, and Michael Menth. P4-macsec: Dynamic topology monitoring and data layer protection with macsec in p4-based sdn. *IEEE Access*, 8:58845–58858, 2020.
35. Keqiang He, Junaid Khalid, Aaron Gember-Jacobson, Sourav Das, Chaithan Prakash, Aditya Akella, Li Erran Li, and Marina Thottan. Measuring control plane latency in sdn-enabled switches. In *SOSR '15*. ACM, 2015.

36. Jubril Hypolite, John Sonchack, Shlomo Hershkop, Nathan Dautenhahn, André DeHon, and Jonathan M Smith. Deepmatch: practical deep packet inspection in the data plane using network processors. In *Proceedings of the 16th International Conference on emerging Networking EXperiments and Technologies*, pages 336–350, 2020.
37. Arthur S Jacobs, Roman Beltiukov, Walter Willinger, Ronaldo A Ferreira, Arpit Gupta, and Lisandro Z Granville. Ai/ml for network security: The emperor has no clothes. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 1537–1551, 2022.
38. Theo Jepsen, Daniel Alvarez, Nate Foster, Changhoon Kim, Jeongkeun Lee, Masoud Moshref, and Robert Soulé. Fast string searching on pisa. In *Proceedings of the 2019 ACM Symposium on SDN Research*, pages 21–28, 2019.
39. Qiao Kang, Liang Xue, Adam Morrison, Yuxin Tang, Ang Chen, and Xiapu Luo. Programmable in-network security for context-aware byod policies. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 595–612, 2020.
40. Xixi Zheng Khooi, Levente Csikor, Dinil Mon Divakaran, and Min Suk Kang. Dida: Distributed in-network defense architecture against amplified reflection ddos attacks. In *2020 6th IEEE Conference on Network Softwarization (NetSoft)*, pages 277–281. IEEE, 2020.
41. Daehyeok Kim, Yibo Zhu, Changhoon Kim, Jeongkeun Lee, and Srinivasan Seshan. Generic external memory for switch data planes. In *Proceedings of the 17th ACM Workshop on Hot Topics in Networks*, pages 1–7, 2018.
42. Peng Kuang, Yong Liu, and Liangliang He. P4dad: Securing duplicate address detection using p4. In *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, pages 1–7. IEEE, 2020.
43. Jan Kučera, Diana Andreea Popescu, Gianni Antichi, Jan Kořenek, and Andrew W Moore. Seek and push: Detecting large traffic aggregates in the dataplane. *arXiv preprint arXiv:1805.05993*, 2018.
44. Ângelo Cardoso Lapolli, Jonatas Adilson Marques, and Luciano Paschoal Gaspary. Offloading real-time ddos attack detection to programmable data planes. In *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 19–27. IEEE, 2019.
45. Abdelali Laraba, Jérôme François, Isabelle Chrisment, Shihabur Rahman Chowdhury, and Raouf Boutaba. Defeating protocol abuse with p4: Application to explicit congestion notification. In *2020 IFIP Networking Conference (Networking)*, pages 431–439. IEEE, 2020.
46. Jonghwan Lee and Kamaldeep Parmar Singh. Switchtree: in-network computing and traffic analyses with random forests. *Neural Computing and Applications*, pages 1–12, 2020.
47. Guozhen Li, Mingwei Zhang, Chunming Liu, Xiao Kong, Ang Chen, Guofei Gu, and Haixin Duan. Nethcf: Enabling line-rate and adaptive spoofed ip traffic filtering. In *2019 IEEE 27th International Conference on Network Protocols (ICNP)*, pages 1–12. IEEE, 2019.
48. Jia Li, Hui Jiang, Weijia Jiang, Jason Wu, and Wenyuan Du. Sdn-based stateful firewall for cloud. In *2020 IEEE 6th International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing,(HPSC) and IEEE International Conference on Intelligent Data and Security (IDS)*, pages 157–161. IEEE, 2020.
49. R. Li, Q. Li, Y. Zhang, D. Zhao, X. Xiao, and Y. Jiang. Genos: General in-network unsupervised intrusion detection by rule extraction. In *IEEE INFOCOM 2024 - IEEE Conference on Computer Communications*, 2024.
50. Yi-Bing Lin, Tsung-Jui Huang, and Sheng-Chen Tsai. Enhancing 5g/iot transport security through content permutation. *IEEE Access*, 7:94293–94299, 2019.
51. Gang Liu, Wei Quan, Nan Cheng, Dingde Gao, Ning Lu, Hongke Zhang, and Xuemin Shen. Softwarized iot network immunity against eavesdropping with programmable data planes. *IEEE Internet of Things Journal*, 8(12):9847–9860, 2021.
52. Zaoxing Liu, Hun Namkung, Georgios Nikolaidis, Jeongkeun Lee, Changhoon Kim, Xin Jin, Vladimir Braverman, Minlan Yu, and Vyas Sekar. Jaqen: A high-performance switch-native approach for detecting and mitigating volumetric ddos attacks with programmable switches. In *30th USENIX Security Symposium*, 2021.
53. Marek Majkowski. The real cause of large ddos - ip spoofing, 2018.
54. Lukas Malina, David Smekal, Sara Ricci, Jan Hajny, Peter Cíbik, and Jan Hrabovsky. Hardware-accelerated cryptography for software-defined networks with p4. In *International Conference on Information Technology and Communications Security*, pages 271–287. Springer, 2020.
55. Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, 2008.
56. Nour Moustafa and Jill Slay. Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In *MilCIS 2015*, 2015.
57. Nour Moustafa and Jill Slay. The evaluation of network anomaly detection systems: Statistical analysis of the unsw-nb15 data set and the comparison with the kdd99 data set. *Inf. Sec. J.: A Global Perspective*, 25(1–3), 2016.

58. N Narayanan, GC Sankaran, and KM Sivalingam. Mitigation of security attacks in the sdn data plane using p4-enabled switches. In *2019 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, pages 1–6. IEEE, 2019.
59. Isaac Oliveira, Emídio Neto, Roger Immich, Ramon Fontes, Augusto Neto, Fabrício Rodriguez, and Christian Esteve Rothenberg. dh-aes-p4: On-premise encryption and in-band key-exchange in p4 fully programmable data planes. In *2021 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pages 148–153, 2021.
60. Qun Qin, Konstantinos Poularakis, Kin K Leung, and Leandros Tassiulas. Line-speed and scalable intrusion detection at the network edge via federated learning. In *2020 IFIP Networking Conference (Networking)*, pages 352–360. IEEE, 2020.
61. R Ricart-Sanchez, P Malagon, JM Alcaraz-Calero, and Q Wang. Hardware-accelerated firewall for 5g mobile networks. In *2018 IEEE 26th International Conference on Network Protocols (ICNP)*, pages 446–447. IEEE, 2018.
62. Roberto Ricart-Sanchez, Pedro Malagon, Jose M Alcaraz-Calero, and Qi Wang. Netfpga-based firewall solution for 5g multi-tenant architectures. In *2019 IEEE International Conference on Edge Computing (EDGE)*, pages 132–136. IEEE, 2019.
63. Dominik Scholz, Andreas Oeldemann, Fabien Geyer, Sebastian Gallenmüller, Henning Stubbe, Thomas Wild, Andreas Herkersdorf, and Georg Carle. Cryptographic hashing in p4 data planes. In *2019 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, pages 1–6. IEEE, 2019.
64. Michael Seufert, Konstantin Dietz, Nils Wehner, Steffen Geißler, Julian Schüler, Maximilian Wolz, Andreas Hotho, Pedro Casas, Tobias Hoßfeld, and Anja Feldmann. Marina: Realizing ml-driven real-time network traffic monitoring at terabit scale. *IEEE Transactions on Network and Service Management*, pages 1–1, 2024.
65. Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *ICISSP 2018*, pages 108–116, 2018.
66. Gürkan Simsek, Hasret Bostan, A Kemal Sarica, Emre Sarikaya, Avincan Keles, Pelin Angin, Hande Alemdar, and Ertan Onur. Dropppp: A p4 approach to mitigating dos attacks in sdn. In *International Workshop on Information Security Applications*, pages 55–66. Springer, 2019.
67. Giuseppe Siracusano and Roberto Bifulco. In-network neural networks. *arXiv preprint arXiv:1801.05731*, 2018.
68. Vibhaalakshmi Sivaraman, Srinivas Narayana, Ori Rottenstreich, S Muthukrishnan, and Jennifer Rexford. Heavy-hitter detection entirely in the data plane. In *Proceedings of the Symposium on SDN Research*, pages 164–176, 2017.
69. Lingjun Tang, Qun Huang, and Patrick PC Lee. A fast and compact invertible sketch for network-wide heavy flow detection. *IEEE/ACM Transactions on Networking*, 28(5):2350–2363, 2020.
70. Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani. A detailed analysis of the kdd cup 99 data set. In *2009 IEEE CISDA*, 2009.
71. The P4.org Architecture Working Group. P4_16 portable switch architecture (psa), 2024.
72. G. Xie, Q. Li, Y. Dong, G. Duan, Y. Jiang, and J. Duan. Mousika: Enable general in-network intelligence in programmable switches by knowledge distillation. In *IEEE INFOCOM 2022*, pages 1938–1947, 2022.
73. Guang Xie, Qing Li, Chao Cui, Peng Zhu, Dehua Zhao, Wen Shi, Zuqing Qi, Yanwei Jiang, and Xiaobo Xiao. Soter: Deep learning enhanced in-network attack detection based on programmable switches. In *2022 41st International Symposium on Reliable Distributed Systems*, 2022.
74. Jiarong Xing, Qiao Kang, and Ang Chen. Netwarden: Mitigating network covert channels while preserving performance. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 2039–2056, 2020.
75. Jiarong Xing, Wenqing Wu, and Ang Chen. Architecting programmable data plane defenses into the network with fastflex. In *Proceedings of the 18th ACM Workshop on Hot Topics in Networks*, pages 161–169, 2019.
76. Zhaoqi Xiong and Noa Zilberman. Do switches dream of machine learning? toward in-network classification. In *HotNets 2019*, pages 25–33. ACM, 2019.
77. Elisa Ordoñez Zaballa, Daniel Franco, Zhou Zhou, and Michael S Berger. P4knocking: Offloading host-based firewall functionalities to the network. In *2020 23rd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, pages 7–12. IEEE, 2020.
78. Menghao Zhang, Chen Li, Yuwei Zhang, Yang Xiang, Zhenyu Wu, and Hai Jin. Bolt: Bridging the gap between software and hardware for stateful and efficient network functions. In *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*, 2021.
79. Mingwei Zhang, Guozhen Li, Shangqing Wang, Chunming Liu, Ang Chen, Haoyu Hu, Guofei Gu, Qi Li, Minglong Xu, and Jianping Wu. Poseidon: Mitigating volumetric ddos attacks with programmable switches. In *Network and Distributed System Security Symposium*, 2020.

80. Xiang Zhang, Lin Cui, Fung Po Tso, and Weijia Jia. pheavy: Predicting heavy flows in the programmable data plane. *IEEE Trans. Netw. Serv. Manag.*, 18(4):4353–4364, 2021.
81. Cheng Zheng and Noa Zilberman. Planter: Seeding trees within switches. In *SIGCOMM '21 Poster and Demo Sessions*, pages 12–14. ACM, 2021.
82. Guangmeng Zhou, Zhongxin Liu, Chengjie Fu, Qing Li, and Ke Xu. An efficient design of intelligent network data plane. In *USENIX Security 23*, 2023.

Author Biographies

Gursimran Singh is a PhD candidate at Rochester Institute of Technology. His research focus is on network security applications in the data plane.

H.B. Acharya is an assistant professor of computer science at Oklahoma State University. His interests are in the area of Networks and Cybersecurity, including topics like data plane network security, internet interference,etc.

Minseok Kwon is a professor in the Department of Computer Science at Rochester Institute of Technology. His main research interests are in the area of computer networks, distributed computing, and network security.