Journal of Network and Computer Applications 242 (2025) 104257

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Review Article

# A survey on security applications with SmartNICs: Taxonomy, implementations, challenges, and future trends

Sergio Elizalde[a,*], Ali AlSabeh[b], Ali Mazloum[a], Samia Choueiri[a], Elie Kfoury[a], Jose Gomez[c], Jorge Crichigno[a]

[a] Molinaroli College of Engineering and Computing, University of South Carolina, Columbia, SC, USA
[b] College of Sciences and Engineering, University of South Carolina Aiken, Aiken, SC, USA
[c] Katz School of Business, Fort Lewis College, Durango, CO, USA

ARTICLE INFO

*Keywords:*
SmartNIC
Security applications
Intrusion detection systems
Volumetric attacks
Privacy
Threat model

ABSTRACT

Over the last decade, network applications have grown exponentially, demanding high-speed interconnects. Unfortunately, chip manufacturers are approaching the upper limits of silicon-based computing with slow improvements in computational performance and energy efficiency. This trend has forced the industry to shift paradigms, moving from monolithic architectures to heterogeneous, domain-specific designs. Moreover, the ever-evolving threats compromise digital services and demand more scalable and flexible solutions to ensure service continuity in production networks. Smart Network Interface Cards (SmartNICs) are a product of this new paradigm, integrating domain-specific engines and general-purpose cores to offload various network infrastructure tasks, including those related to security. This paper provides a comprehensive overview of SmartNICs, with a particular focus on their role in strengthening network defenses. It introduces SmartNIC technology and presents a taxonomy of security applications offloaded to SmartNICs, categorized into Intrusion Detection and Prevention Systems (IDS/IPS), defenses against volumetric attacks, and data confidentiality mechanisms. Additionally, the paper explores vulnerabilities associated with adopting SmartNICs in the cloud, examining the threat model and reviewing proposed remediations in the literature. Finally, it discusses challenges and future trends in SmartNIC security applications, highlighting current initiatives and open research areas.

**Contents**

* Corresponding author.
*E-mail addresses:* elizalds@email.sc.edu (S. Elizalde), ali.alsabeh@usca.edu (A. AlSabeh), amazloum@email.sc.edu (A. Mazloum), choueiri@email.sc.edu (S. Choueiri), ekfoury@email.sc.edu (E. Kfoury), jagomez@fortlewis.edu (J. Gomez), jcrichigno@cec.sc.edu (J. Crichigno).

https://doi.org/10.1016/j.jnca.2025.104257
Received 17 February 2025; Received in revised form 26 May 2025; Accepted 18 June 2025
Available online 1 July 2025

## 1. Introduction

Network security is crucial in safeguarding sensitive data and maintaining operational integrity in the digital age. As cyber threats evolve, organizations and individuals face increasing risks from cyberattacks. Implementing robust network security measures is essential to prevent unauthorized access, protect privacy, and ensure the continuity of digital services (Aslan et al., 2023). Weaknesses in network security systems create vulnerabilities that cybercriminals can exploit to access private data or disrupt services. Recent incidents underscore the importance of strong network security. For example, the MOVEit file transfer service experienced a significant security breach in 2023, which exposed the data of around 1800 networks worldwide (Gooding, 2024).

Traditionally, middleboxes have been deployed to mitigate and safeguard networks. These devices offer high-performance guarantees, making them well-suited for enterprise networks, data centers, and wide-area networks (Sherry et al., 2012). However, they are often costly, difficult to manage, and fully proprietary, which complicates the process of upgrading their functionalities.

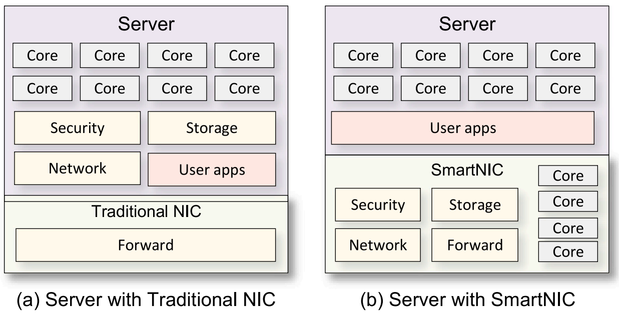


**Fig. 1.** Server infrastructure tasks offloaded to SmartNICs (Kanev et al., 2015). (a) In traditional setups, the server's CPU cores handle infrastructure tasks. (b) In modern data centers, SmartNICs take over these tasks, thereby saving CPU cycles on the host and isolating the server from the infrastructure execution environment.

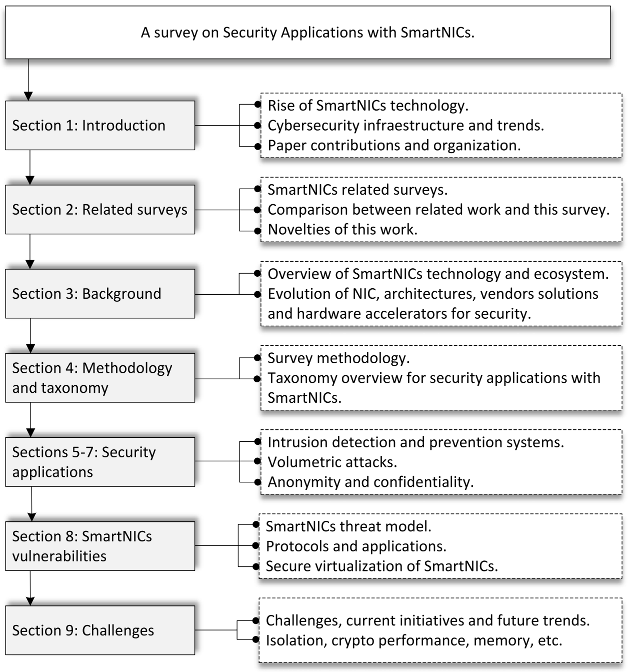

A survey on Security Applications with SmartNICs.
Section 1: Introduction
Rise of SmartNICs technology.
Cybersecurity infraestructure and trends.
Paper contributions and organization.
Section 2: Related surveys
SmartNICs related surveys.
Comparison between related work and this survey.
Novelties of this work.
Section 3: Background
Overview of SmartNICs technology and ecosystem.
Evolution of NIC, architectures, vendors solutions and hardware accelerators for security.
Section 4: Methodology and taxonomy
Survey methodology.
Taxonomy overview for security applications with SmartNICs.
Sections 5-7: Security applications
Intrusion detection and prevention systems.
Volumetric attacks.
Anonymity and confidentiality.
Section 8: SmartNICs vulnerabilities
SmartNICs threat model.
Protocols and applications.
Secure virtualization of SmartNICs.
Section 9: Challenges
Challenges, current initiatives and future trends.
Isolation, crypto performance, memory, etc.

**Fig. 2.** Proposed roadmap of the survey.

With the rise of cloud systems, virtual security equipment has emerged as a more flexible alternative (Luo et al., 2020). This modern approach utilizes Software-Defined Networking (SDN) and Network Function Virtualization (NFV) to execute security functions on general-purpose Central Processing Units (CPUs). SDN provides centralized control and high operational flexibility due to its software-based architecture. Despite these advantages, the transition to software-driven networks results in degraded packet processing performance (Pereira et al., 2024), marked by lower throughput and higher latency. Such limitations make SDN and NFV inadequate for defending against volumetric attacks, which can escalate to several Terabits per second (Tbps) in bandwidth (Lapolli et al., 2019).

To overcome the limitations of traditional approaches, programmable data plane technologies such as programmable switches and SmartNICs have emerged. Unlike traditional fixed-function switches, programmable switches allow administrators to define custom packet processing logic using network programming languages. The *de facto* language used to define the behavior of packet processing is the Programming Protocol-Independent Packet Processors (P4) (P4 Language Consortium, 2024). This flexibility facilitates innovation and rapid testing of new security solutions, making programmable switches a focal point in the research for network security applications (AlSabeh et al., 2022b; Chen et al., 2023). Following the same design principles as programmable switches, the programmability of packet processing has, in recent years, been extended to Network Interface Cards (NICs), resulting in the evolution of SmartNICs. These devices integrate domain-specific processors with general-purpose CPUs, enabling developers to program applications on the SmartNIC's CPU while offloading and accelerating specific tasks using specialized engines, such as encryption, compression, and packet filtering. This combination delivers substantial performance gains while reducing the computational burden on the host system (Xing et al., 2023).

The rising adoption of SmartNICs is clearly reflected in the global information technology ecosystem, where hyperscalers, such as Google, Amazon, and Microsoft, are designing custom SmartNICs to offload infrastructure tasks, thereby optimizing operational expenditures (Firestone et al., 2018; Dancheva et al., 2024). Manufacturers like Intel, NVIDIA, and AMD are releasing SmartNIC models that integrate domain-specific processors for networking, storage, and security, along with general-purpose CPUs for enterprise and research institutions (Sundar et al., 2023; Dastidar et al., 2023).

SmartNICs are gaining increased attention in cybersecurity as they allow operators to offload networking functions from the host, thus creating hardware isolation by running infrastructure in a separate execution environment from user applications (see Fig. 1). Moreover, network security services are being deployed on SmartNICs, including Next-Generation Firewalls (NGFWs), which offer a portfolio of security functions such as encryption and decryption, intrusion detection and prevention, and others. Furthermore, SmartNICs can provide data provenance for various applications, such as digital twins, Machine Learning (ML) training data, and botnet detection, thus enhancing trust and fidelity across different fields (Hao et al., 2023; Seo et al., 2024). In cryptography, SmartNICs are used to test quantum-safe algorithms adopted by the National Institute of Standards and Technology (NIST) (Beckwith et al., 2023; Lawo et al., 2024). The flexibility of SmartNICs, which can be continuously reprogrammed to comply with updated security standards, allows developers to release updates rapidly and mitigate security threats without requiring a complete system redesign while maintaining both performance and cost efficiency.

### 1.1. *Paper contributions*

Although SmartNICs have garnered increasing attention from both academia and industry, there is still a significant lack of comprehensive research in the area of network security. Specifically, there is no thorough survey about SmartNICs that aggregates all network security-related papers and organizes them into appropriate cybersecurity categories while simultaneously highlighting vulnerabilities and threat models. To address this gap, this paper makes four key contributions. First, it presents the advancements in SmartNICs, starting from traditional NICs, and examines how these developments enhance cybersecurity. Second, the paper introduces a taxonomy of security functions and applications offloaded to SmartNICs, detailing the techniques and methods used. Third, the paper comprehensively discusses the vulnerabilities associated with adopting SmartNICs and describes proposed solutions found in the literature, thus raising security awareness among researchers and practitioners who deploy this technology in their networks. Fourth, the paper identifies open challenges and outlines emerging trends in the SmartNIC-enabled security landscape, providing a forward-looking perspective for future research. To the best of the authors' knowledge, no previous article has provided a security-tailored comprehensive review of SmartNICs.

**Table 1**
Comparison with related surveys.

| Survey | Technology background | | | Taxonomy | | | | Threat model | Discussion | | Focus of the survey |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | Evolution | Description | Features | Overview | Literature review | Intra-category comparison | Comparison with programmable switches | | Challenges | Future directions | |
| Linguaglossa et al. (2019) | ⊙ | ⊙ | ⊙ | ⊙ | ⊙ | ○ | ⊙ | ○ | ⊙ | ● | NFV |
| Fei et al. (2020) | ⊙ | ⊙ | ⊙ | ⊙ | ⊙ | ⊙ | ● | ○ | ⊙ | ⊙ | NFV |
| Shantharama et al. (2020) | ● | ● | ⊙ | ⊙ | ⊙ | ⊙ | ⊙ | ○ | ⊙ | ⊙ | NFV |
| Freitas et al. (2022) | ○ | ⊙ | ○ | ⊙ | ⊙ | ○ | ○ | ○ | ● | ● | OS |
| Zheng et al. (2023) | ● | ⊙ | ⊙ | ⊙ | ⊙ | ○ | ⊙ | ○ | ⊙ | ⊙ | ML |
| Rosa et al. (2024) | ⊙ | ⊙ | ⊙ | ⊙ | ⊙ | ⊙ | ⊙ | ○ | ● | ● | General |
| Nickel and Göhringer (2024) | ● | ● | ● | ⊙ | ⊙ | ○ | ⊙ | ○ | ● | ⊙ | General |
| Kfoury et al. (2024) | ● | ● | ● | ⊙ | ⊙ | ⊙ | ⊙ | ○ | ● | ● | General |
| This survey | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | Security |

● Covered in this survey ○ Not covered in this survey ⊙ Partially covered in this survey.

### 1.2. Paper organization

The roadmap of this survey is depicted in Fig. 2. Section 2 compares existing surveys on SmartNICs and highlights the added value of this work. Section 3 presents an overview of SmartNICs and their architectures, features, and limitations. It also describes the advancement in hardware for security and the portfolio of security functions deployed in SmartNICs. Section 4 discusses the survey methodology and describes the proposed taxonomy. Sections 5–7 collect and analyze SmartNICs applications that address network intrusion detection and prevention systems, volumetric attacks such as heavy hitters and Distributed Denial of Services (DDoS), and confidentiality of network communications. Section 8 outlines potential vulnerabilities from adopting SmartNICs in the shared cloud, and Section 9 states the challenges and future trends in network security using SmartNICs. The abbreviations used in this article are summarized in Table 11 at the end of the article.

## 2. Related surveys

Linguaglossa et al. (2019) present a survey on performance acceleration techniques for NFV. They analyze the methods to scale and remove inherent bottlenecks in the deployment of network functions such as Network Address Translation (NAT), firewalls, and mobile functions. Moreover, the authors describe the commonly used primitives offloaded to SmartNICs and discuss how programmability increases the flexibility demanded by NFV applications. This survey briefly discusses the NGFW application deployed in SmartNICs, but it is still missing many other security applications that can be offloaded to it with a particular focus on SmartNIC.

Fei et al. (2020) analyze the performance challenges in general-purpose servers and summarize the typical performance bottlenecks when offloading NFVs. The authors review the progress in NFV acceleration and introduce a new taxonomy of state-of-the-art efforts based on various acceleration approaches. The authors discuss the surveyed works, identifying the advantages and disadvantages of each category. They also examine the products, solutions, and projects emerging in the industry. The authors also identify applications that can be implemented with SmartNICs. Additionally, they present a gap analysis to improve current solutions and highlight promising research trends for future exploration. The survey primarily focuses on NFV deployed in hardware accelerators, not including security applications of SmartNICs.

Shantharama et al. (2020) present a comprehensive survey covering commercial network accelerators and relevant research studies. The survey demonstrates that hardware acceleration can offload tasks from the general-purpose CPU and enhance its capabilities through co-processing. The authors categorize their findings into CPU hardware accelerations, memory enhancements, and interconnect improvements (including on-chip and chip-to-chip connections). They also examine hardware-accelerated infrastructures like SmartNICs, which connect general-purpose CPU platforms to networks. In terms of security, the survey covers acceleration for encryption and decryption in CPU architecture. Also, the poor isolation of NFV is discussed in virtual environments, such as kernel or user space applications. However, the survey does not address SmartNIC's security applications and benefits for overcoming these challenges.

Freitas et al. (2022) present a systematic review of accelerating technologies for fast network packet processing in Linux environments. When handling network packets, they identify the existing overheads and bottlenecks in the Linux kernel. The review introduces a taxonomy of fast packet processing solutions categorized into hardware, software, and virtualization. The authors then discuss these solutions in terms of host resource usage, high packet rates, system security, and flexibility. The survey discusses kernel security, isolation, and DoS over the Peripheral Component Interconnect Express (PCIe) link between Virtual Machines (VMs); however, it lacks literature on security and SmartNIC applications.

Zheng et al. (2023) discuss various in-network ML solutions developed using P4-programmable devices, including SmartNICs. They explore different ML models implemented within the network and examine the related challenges and solutions. In the survey, the authors presented various ML algorithms in programmable data planes and explained the implementation challenges. The authors emphasize that in-network machine learning can greatly enhance cloud computing and next-generation networks. The survey concludes with a discussion of future trends in this area. The survey focuses on ML techniques and some applications of inference for anomaly detection in P4 switches, rather than SmartNICs.

Rosa et al. (2024) review recent research efforts aimed at enabling cloud platforms to offer network acceleration as a service. It focuses on technologies such as eXpress Data Path (XDP), Data Plane Development Kit (DPDK), and Remote Direct Memory Access (RDMA), which are implemented with SmartNICs. The paper identifies four key aspects critical to integrating acceleration options in cloud computing: access interfaces, virtualization techniques, serviceability, and security. The survey discusses the security implications of network acceleration in public clouds, focusing on confidentiality, integrity, authentication, access control, and isolation, but it misses IDS/IPS applications, volumetric attacks, and the SmartNIC threat model assessment.

Nickel and Göhringer (2024) propose a taxonomy for programmable network devices, including SmartNICs, FPGAs, and switches. The survey emphasizes these devices' architectural characteristics and data processing capabilities in the context of in-network computing. Particular attention is given to FPGAs, analyzing their strengths and weaknesses in terms of power efficiency and acceleration capabilities. The taxonomy introduces three main categories: network device type, programming language/model, and data plane architecture. The programming model considers the development frameworks and languages used for network devices, while the network device type distinguishes between NICs and switches. The architecture has three subcategories: programmable ASICs, reconfigurable hardware, and System-on-Chip (SoC) designs. Regarding security, the survey briefly discusses IDS/IPS systems in FPGA-based SmartNICs and highlights encryption research directions for these systems.

The closest work to ours is a recent survey on SmartNICs by Kfoury et al. (2024). The survey begins with the evolution of legacy NICs and the technological drivers behind the emergence of SmartNICs. Then, the survey reviews the available SmartNIC hardware architectures and examines open-source and vendor-specific development environments for SmartNICs. Additionally, a taxonomy is presented to classify the literature based on security, network, and compute functions offloaded to SmartNICs.

The novelties of our work pertaining to security, and compared to Kfoury et al. (2024), are:

- Our work presents a security-focused taxonomy that covers a broader application landscape, including volumetric attacks (heavy hitters, DDoS), anomaly-based IDS/IPS systems implemented using SmartNICs, and privacy-preserving mechanisms, all mapped specifically to SmartNIC-enabled architectures.
- The authors in Kfoury et al. (2024) present a generic overview of SmartNIC applications. Although they discuss SmartNIC security-related work, the treatment of this topic is relatively brief and does not cover relevant ongoing research. Our survey is more granular and it thoroughly examines a broad range of security literature. For example, in IDS/IPS, various techniques have evolved to detect malicious activities, including signature-based and anomaly-based systems, each leveraging different techniques using SmartNICs.
- Our survey uniquely discusses the inherent security risks and attack surfaces introduced by SmartNICs themselves, which are crucial for system architects and practitioners. We complement this by analyzing remediation techniques proposed in the literature.
- The survey outlines key open research directions at the intersection of SmartNICs and network security, extracted from the reviewed literature and thoroughly annotated with ongoing efforts and future trends. These directions include topics such as isolation in multi-tenant environments, performance modeling and quantification, and addressing challenges across heterogeneous ecosystems.

Table 1 provides a comparative analysis of existing surveys, delineating key distinctions relative to the scope and contributions of this survey. In contrast to prior surveys, we focus on the recent advances in network security and SmartNICs. Additionally, we consider programmable switches as a precursor of SmartNICs and highlight their different applications and use cases in network security. We believe that our review can help a variety of readers understand why emerging network security techniques are increasingly choosing SmartNICs, and highlight future research directions on that topic.

## 3. Background

### 3.1. Evolution of NICs

Before exploring security applications, it is important to understand the foundational concepts of SmartNICs, including their evolution, architectures, and role in distributed security. Over the past two decades, NICs have progressed from traditional designs to Offload NICs and, more recently, to programmable SmartNICs. This evolution has enabled more efficient data processing and enhanced security capabilities. Table 2 summarizes the key characteristics of these NIC types.

#### 3.1.1. Traditional NICs

Traditional NICs, also known as standard or basic NICs, are hardware components that connect a computer to a network. These devices implement fundamental physical and data-link layer services, including frame serialization/deserialization, link access management, and communication error detection. Typically, these functions are performed by a dedicated fixed-function component on a dedicated chip within the NIC. On the transmitting side, the fixed-function component takes a datagram from the host, encapsulates it in a link-layer frame, and transmits the frame over the communication link according to the link-access protocol. On the receiving side, the component receives the frame and forwards it to the host via a PCIe bus. A traditional NIC has the following features:

- Basic packet processing: Traditional NICs primarily focus on moving data packets between the network and the host system's CPU. They have limited processing capabilities and rely heavily on the host CPU to handle networking tasks.
- Simple protocol handling: These NICs manage basic protocols like Ethernet and Internet Protocol (IP) but do not offload complex processing tasks from the CPU.
- Limited customization: Traditional NICs are generally fixed-function devices with minimal scope for customization or programmability.

#### 3.1.2. Offload NICs

Offload NICs integrate hardware acceleration engines to execute basic infrastructure functions that were previously managed by the network stack in the host. The goal is to free up host CPU cycles and enhance the performance. Examples of such functions are:

- TCP offload engine: Offload NICs often include a Transmission Control Protocol (TCP) offload engine that handles TCP/IP processing tasks, reducing the load on the host CPU and leaving more resources for end-user applications.
- Checksum offloading: These NICs can compute checksum for data packets, saving CPU clock cycles.
- Basic packet filtering: They can perform simple packet filtering and classification to improve network performance and security.

#### 3.1.3. SmartNICs

SmartNICs represent the latest advancement in NIC technology. Although there is no consensus on the definition of SmartNICS, NICs that perform functions beyond basic packet processing and have programmable elements may be considered SmartNICs. These devices are SoCs which typically have their own Operating System (OS), on-chip memory, general-purpose CPU cores, and programmable Domain-Specific Accelerators (Dally et al., 2020) (DSAs), see Fig. 3(a). Typical DSAs within the SmartNIC include a P4 pipeline for packet header processing, a cryptographic accelerator for encryption/decryption operations, a regular expression accelerator for pattern matching, and a compression accelerator for compression/decompression operations. Incoming packets are received (RX) and moved to the traffic manager. The packet may then be processed by one or more DSAs, as programmed by the engineer. If the packet is directed to the host device the SmartNIC is attached to, then it is forwarded via the PCI bus. Otherwise, the packet is sent to the network (TX). Fig. 3(b) shows an example of a DSA, the P4 pipeline. This accelerator processes packet headers and consists of the following components: programmable parser, programmable match-action pipeline, and programmable deparser. The programmable parser allows the developer to define packet headers based on custom or standard protocols and parse them. It is represented as a state machine. The programmable match-action pipeline executes the operations over the packet headers and intermediate results. A single match-action stage has multiple memory blocks (tables, registers) and Arithmetic Logic Units (ALUs), which allows for simultaneous lookups and actions. Since some action results may be needed for further processing (e.g., data dependencies), stages are arranged sequentially. The programmable deparser assembles the packet headers back and serializes them for transmission out of the DSA. The DSA is programmed with P4, a programming language tailored to packet header processing. The main features of SmartNICs are summarized as follows:

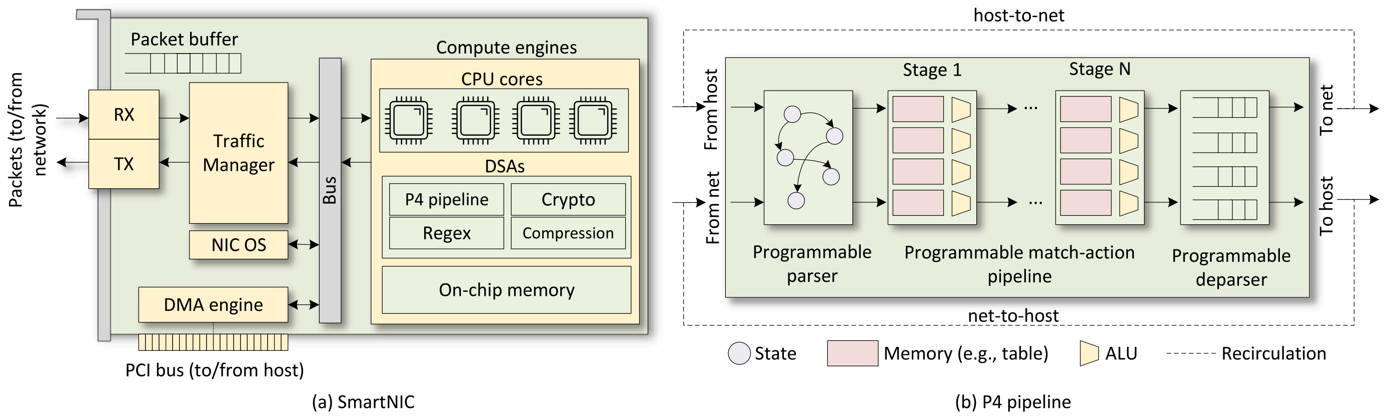


**Fig. 3.** SmartNIC architecture and programmable pipeline. (a) Packets from the network arrive on the port (RX), and from the host on the PCI bus. Similarly, packets to the network are forwarded to the port (TX), and to the host on the PCI card. Packets are then handled by the traffic manager, which moves the packet to the appropriate compute engine. Compute engines include CPU cores and DSAs. (b) The P4 pipeline is a DSA designed for packet header processing, comprising a programmable parser, a match-action pipeline, and a deparser. Packets can follow different paths: from the network to the host (net-to-host), from the host to the network (host-to-net), or be recirculated through the pipeline for additional processing.

**Table 2**
Features of traditional/Offload/Smart NICs.

| Feature | Traditional NIC | Offload NIC | SmartNIC |
|---|---|---|---|
| Infrastructure function separation | Low | Medium | High |
| Security isolation | Low | Low | High |
| General-purpose CPU | No | No | Yes |
| Acceleration engines | Low | Medium | High |
| Autonomous system | No | No | Yes |
| Protocol flexibility | Low | Medium | High |
| Standardized models | Yes | Yes | No |
| Technology maturity | High | High | Medium |

- Programmability: SmartNICs feature a programmable pipeline that enables custom packet processing. This flexibility is powered by underlying hardware technologies, including Field Programmable Gate Arrays (FPGAs), Application-Specific Integrated Circuits (ASICs), and general-purpose CPU cores.
- Advanced offloading: They can offload a wide range of tasks, including encryption/decryption, load balancing, and compression/decompression, among others.
- Enhanced security: SmartNICs can implement advanced security features, such as inline DoS mitigation, firewall functions, and IDS/IPS.
- Virtualization: SmartNICs support NFV, enabling the virtualization of network functions for more flexible and scalable network architectures.
- Autonomy: SmartNICs integrate a full network stack by running lightweight OS, thus being an independent endpoint in the network.

## 3.2. *SmartNICs architectures*

SmartNICs achieve their flexibility through various underlying technologies, which also define their architectural distinctions. A widely accepted definition characterizes SmartNICs as SoC-based devices that integrate computing engines and multi-core RISC architectures, such as ARM or MIPS. These architectures typically include hierarchical on-chip memory, comprising Static Random-Access Memory (SRAM) and Dynamic Random-Access Memory (DRAM), and Input/Output (I/O) buses for communication with internal processing units and the host system when needed. Alternatively, SmartNICs may incorporate different combinations of FPGAs, ASICs, and CPUs, offering customizable solutions that balance flexibility, performance, and power consumption based on specific application needs.

### 3.2.1. *SoC-based*

SmartNICs based on SoC architecture provide the capability to offload software components originally designed for the CPU. These SmartNICs are equipped with general ARM processors, which offer a high degree of flexibility, ease of programmability, and the ability to run new applications. An operating system typically runs on these SoC-based SmartNICs, managing NIC resources. Therefore, SoC-based SmartNICs can be considered individual computing nodes within the server, where the SmartNIC functions as an endpoint capable of controlling the full network stack. However, this architecture also brings some limitations similar to those of CPUs, such as limited parallelizability and latency overheads due to inherent architectural constraints.

### 3.2.2. *ASIC-based*

ASIC-based SmartNICs utilize application-specific integrated circuits or customized chips for packet processing. Typically, these SmartNICs incorporate specialized chips designed to perform a wide range of fixed security, networking, and storage functions. They are distinguished by their cost-effectiveness, energy efficiency, and high performance. However, their flexibility is limited by hardcoded functions established during the ASIC design phase. The functions can be configurable but not fully programmable due to the inherent underlying hardware technology.

### 3.2.3. *FPGA-based*

Instead of hardcoding into the chip the functions for networks, the logic circuitry can be programmed using programmable hardware. An FPGA is an integrated circuit composed of programmable logic interconnects, allowing for the customization of digital circuits. FPGA-based SmartNICs are characterized by their high programmability, which enables the acceleration of network functions beyond what purely software-based implementations can achieve. Typically, developers write the entire hardware description using High-Level Synthesis languages and create the corresponding operating system drivers. However, frameworks such as P4FPGA (Wang et al., 2017) allow programmers to develop and evaluate data plane applications using P4 rather than hardware description languages. Moreover, community efforts are underway to develop complete open-source projects for FPGA-based SmartNICs (Xilinx, 2021). These SmartNICs pose challenges such as high power consumption, increased cost, and a lack of dynamic updatability. However, this type of SmartNIC is currently used in hyperscale data centers (Firestone et al., 2018) due to the adaptability to incorporate new features that were not designed at the moment of SmartNIC design.

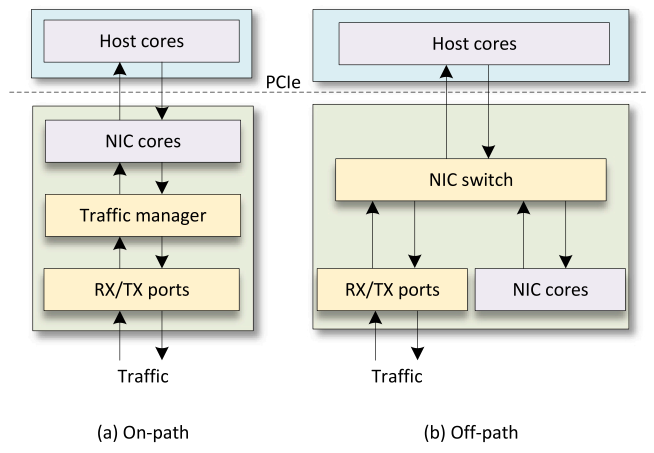


**Fig. 4.** On-path and Off-path SmartNICs. (a) The SmartNIC cores process all incoming traffic. (b) Traffic is steered by the NIC switch to the host or the SmartNIC according to offloaded rules.

### 3.3. On-path and off-path SmartNICs

Another approach to categorizing SmartNIC architectures is based on how their NIC cores interact with network traffic. These can be classified into two categories: on-path and off-path (Liu et al., 2019).

#### 3.3.1. On-path

In on-path SmartNICs (see Fig. 4(a)), the NIC cores actively handle each incoming and outgoing packet along the communication path. These SmartNICs offer low-level programmable interfaces that directly manipulate raw packets. This design ensures the offloaded code is closely situated to the network packets, enhancing efficiency. However, this approach has its drawbacks. The offloaded code competes for NIC cores with requests sent to the host, and excessive computation offloaded to the SmartNIC can significantly degrade the performance of regular networking requests directed to the host. Typically, FPGA or ASIC SmartNICs are used to deploy on-path architecture due to their higher parallelizability, reducing the impact on the traffic.

#### 3.3.2. Off-path

In this setup (see Fig. 4(b)), processing cores and memory are integrated into a separate SoC positioned adjacent to the NIC cores. The offloaded code is strategically positioned outside the critical path of the network processing pipeline. The SoC operates as an independent, fully functional end-point host with its network interface, connected to the NIC cores and the main host via an embedded switch, often referred to as a traffic manager. Depending on the forwarding rules set on the NIC switch, traffic is directed either to the main host or the SmartNIC cores. Unlike on-path SmartNICs, the offloaded code in off-path SmartNICs does not affect the host's network performance. This distinct separation allows the SoC to run a complete operating system (e.g., Linux) with a full network stack, simplifying system development and enabling offloading of complex tasks.

### 3.4. SmartNIC cybersecurity landscape

The disaggregation of networked systems (Shan et al., 2022) is redefining security architectures in modern data centers. SmartNICs enable port-level network intelligence, allowing security functions to be distributed across the infrastructure rather than centralized at perimeter devices. This is particularly relevant given the increasing volume of east–west traffic (Cisco, 2021) and the risks posed by compromised or misbehaving applications (Khan and Nencioni, 2023).

Traditionally, firewalls have served as the first line of defense by enforcing access policies and monitoring traffic (Ingham et al., 2002). However, they are often deployed at the network perimeter, leaving intra-network communication largely unmonitored. To address this, the evolution of firewalls has moved toward distributed models, where enforcement shifts from centralized appliances to the server's NIC. SmartNICs play a critical role in this transition by offloading Next-Generation Firewall (NGFW) functions and enabling deep packet inspection (DPI), intrusion detection/prevention systems (IDS/IPS), and authentication at the edge (Liang and Kim, 2022; Rivitti et al., 2024).

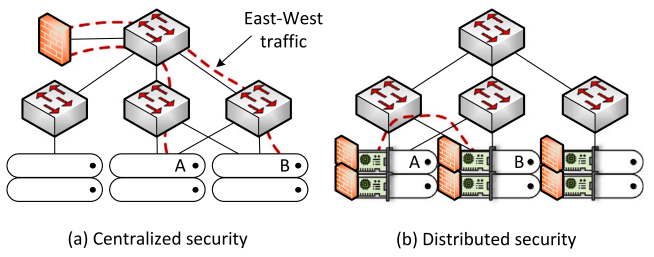


**Fig. 5.** Centralized vs. distributed security using SmartNICs. (a) Central appliances inspect east–west traffic but incur high overhead. (b) SmartNICs enable distributed inspection at each server, improving scalability and performance.

**Table 3**
Micro and network segmentation in network security (Al-Ofeishat and Alshorman, 2023)

| Feature | Network segmentation | Micro-segmentation |
|---|---|---|
| Granularity | Subnets | Individual workloads |
| Implementation | VLAN | Virtual NIC/SmartNIC |
| Zero-trust | Not supported | Supported |
| Scalability | Low | High |
| Complexity | High | Low |

#### 3.4.1. Distributed security with SmartNICs

Traditional, centralized firewalls inspect traffic at a single choke point, often introducing performance bottlenecks and traffic tromboning. In contrast, distributed firewalls push enforcement to the edge, closer to the workloads, using SmartNICs. Fig. 5 illustrates this shift from centralized to distributed inspection.

Distributed firewalls enable context-aware, workload-level policies that improve visibility and eliminate single points of failure. SmartNICs act as enforcement points, inspecting traffic inline and executing policies with minimal latency. This architecture is particularly suitable for virtualized and multi-tenant environments where performance, scalability, and isolation are critical (Yang and Chang, 2023).

#### 3.4.2. Network segmentation and micro-segmentation

Network segmentation is a widely used defensive strategy that reduces an attacker's ability to move laterally. While traditional segmentation divides traffic at the subnet level (e.g., via VLANs), it lacks granularity and offers limited visibility. In contrast, micro-segmentation, popularized by VMware (2014), secures workloads at a finer scale by applying security policies between internal components, even within the same subnet (Wagner et al., 2016). This approach is especially effective in cloud environments, where east–west traffic dominates.

SmartNICs enhance micro-segmentation by enabling security functions directly at the workload's NIC. This supports granular policy enforcement, improves scalability, and reduces complexity. Table 3 summarizes the key differences. Micro-segmentation offers finer control and improved security by isolating workloads individually, while traditional network segmentation operates at the subnet level with limited visibility and scalability.

**Table 4**
SmartNIC security services and their accelerators.

| Security services | SmartNIC security accelerators | | | | | |
|---|---|---|---|---|---|---|
| | TRNG | Hash | Regex | Sym. crypto | Asym. crypto | PKA |
| IDS/IPS | | | ✓ | | | |
| MACsec | ✓ | ✓ | | ✓ | | |
| IPSec | ✓ | ✓ | | ✓ | ✓ | ✓ |
| TLS | ✓ | ✓ | | ✓ | ✓ | ✓ |
| Storage encryption | ✓ | | | ✓ | ✓ | |

#### 3.4.3. Next-generation firewalls

Hardware designers and security companies are collaborating to develop a comprehensive security portfolio for offloading to SmartNICs. Leading firewall vendors are introducing NGFW services based on VM (Cisco Systems, 2024; Juniper Networks, 2024; Fortinet, 2024). Moreover, leading vendors such as Palo Alto Networks, VMware, and NVIDIA have developed ARM-compatible products, which can now be effectively offloaded onto SmartNICs. According to Gartner Research, NGFW should have the following features: IDS/IPS, DPI, application awareness, and support inline configurations (Gartner, 2009). Nowadays, NGFW incorporates more advanced features, including encrypted traffic control, authentication, identity-based control, and data leakage protection (Neupane et al., 2018). Vendors are moving from VM implementation to tailored SmartNIC solutions to enhance performance and security isolation (Palo Alto Networks, 2024; VMware, 2023).

### 3.5. Advances in SmartNIC for security

Infrastructure tasks such as networking, storage, and security are fundamental to data centers, campuses, and enterprise networks. Due to their repetitive and specialized nature, these tasks benefit from hardware acceleration to enhance performance and reduce power consumption. While traditionally handled by general-purpose processors with specialized instruction sets (Cheng et al., 2023; Tschofenig et al., 2015), many of these functions can be offloaded to SmartNICs.

In the context of network security, common primitives, such as encryption, authentication, and integrity checks, can be accelerated using domain-specific hardware. This includes True Random Number Generators (TRNGs) for cryptographic key generation, hardware-accelerated ciphers, regex engines, hash functions, and Public Key Accelerators (PKAs) for operations like Rivest–Shamir–Adleman (RSA), Diffie–Hellman (DH), and Elliptic Curve Cryptography (ECC). These primitives support security services across various layers of the network stack (see Table 4).

SmartNICs enhance the performance of security services such as IDS/IPS, MACsec, IPSec, TLS, and storage encryption by offloading cryptographic functions to hardware accelerators. These accelerators include regex engines, hash functions, and ciphers for symmetric and asymmetric encryption. For example, IDS/IPS systems use regex and hashing to detect anomalies, while MACsec, IPSec, and TLS secure data at various layers of the network stack. Storage encryption safeguards data at rest against breaches.

Beyond standard protocols, researchers are exploring SmartNICs for custom security applications. However, identifying which functions to offload remains a challenge due to the lack of a unified programming model. Emerging solutions propose machine learning-based detection of cryptographic functions for automatic offloading (VenkataKeerthy et al., 2022).

A limitation of fixed-function accelerators is their rigidity in adapting to evolving protocols. Modern protocols like TLS 1.3 (Rescorla, 2018) require flexibility to support various algorithms. FPGA-based SmartNICs offer a solution through reconfigurability, enabling updates to cryptographic logic (e.g., replacing Salsa20 with ChaCha20) without redesigning hardware (Bernstein et al., 2008).

## 4. Methodology and taxonomy

### 4.1. Survey methodology

This survey adopts a structured approach that begins with a broad literature review and progressively narrows the scope to focus on security applications using SmartNICs. The initial search was conducted using academic databases such as IEEE Xplore, ACM Digital Library, and Elsevier (ScienceDirect). Keywords included SmartNIC, Data Processing Unit (DPU), Infrastructure Processing Unit (IPU), programmable NIC, and network security. The search was limited to peer-reviewed papers from journals and conferences published online up to 2024. More than 50 core papers were selected, with the majority published within the last three years. The filtering focused on studies that offload network security functions and applications to SmartNICs, excluding those related to general-purpose or unrelated applications. The selected papers are categorized into distinct taxonomy groups based on the specific security functions and approaches they address. This function-driven taxonomy was derived from analyzing the primary objectives and techniques emphasized in the surveyed works. Grouping the literature in this way reflects established practices in security research and enables a clearer comparison of how SmartNICs are applied across different security domains.

### 4.2. Taxonomy overview

The taxonomy of SmartNICs security applications is proposed in Fig. 6. It categorizes the SmartNIC security applications into three main categories: (1) Intrusion Detection and Prevention Systems, (2) Volumetric Attacks, and (3) Anonymity and Confidentiality. These categories are not entirely orthogonal, meaning a single work may fit into multiple categories. For example, an IDS/IPS could detect and mitigate a volumetric attack. However, as volumetric attacks evolve and new approaches emerge to address them, such works are treated separately from others.

Each category provides background information on the discussed topic (e.g., DDoS, heavy hitters, etc.), giving the reader a brief introduction. This is followed by a literature review that covers relevant SmartNICs and security papers. Additionally, the subsection discusses the limitations of these approaches, providing both an intra-comparison and a comparison with programmable switches in terms of application use cases. Programmable switches are considered precursors to programmable NICs, as many design principles and frameworks have been inherited from this technology, and some works included in the taxonomy can be offloaded to both target devices.

## 5. Intrusion detection and prevention systems

The first category in the taxonomy presents applications and techniques for developing IDS/IPS using SmartNICs. The goal of IDS/IPS is to identify in real-time unauthorized use, misuse, and abuse of computer network systems by internal or external network entities and apply policies to mitigate it (Mukherjee et al., 1994). This section addresses two broad categories for IDS/IPS: signature-based and anomaly-based (Brown et al., 2002). The former leverages the programmability of SmartNICs for rules matching, and the latter includes traffic monitoring with machine learning and cooperative analysis for detecting network threats.

### 5.1. Signature-based

#### 5.1.1. Background

Signature-based IDS/IPS match a set of stored rules with network traffic patterns. An alarm signal or action is triggered when the extracted traffic pattern matches an intrusion signature in the database. Examples of network signatures include source and destination IP

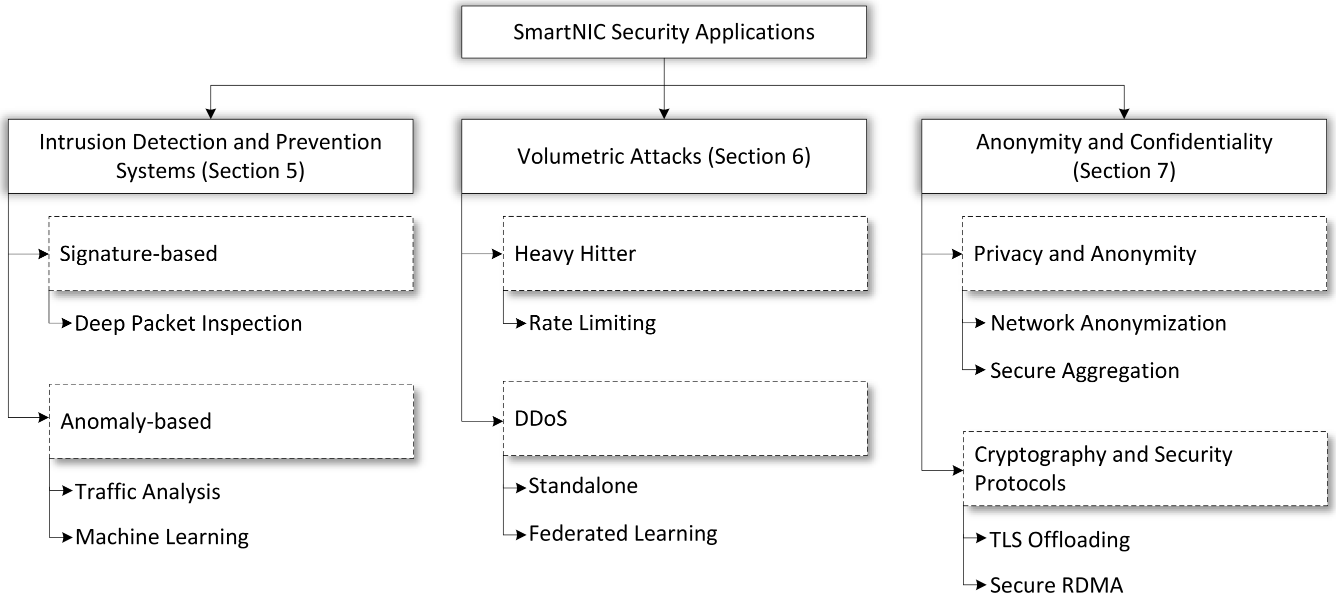


**Fig. 6.** Security taxonomy of SmartNICs based upon explored research areas.

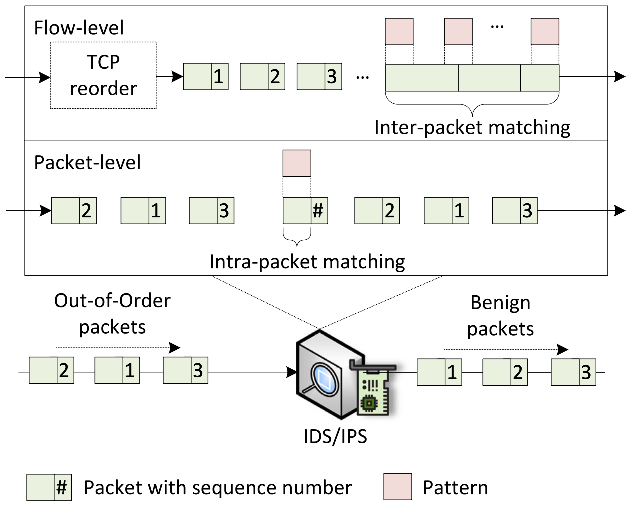


**Fig. 7.** Signature-based IDS/IPS using pattern matching over packet-level and flow-level inspection. In inter-packet matching, packets have to be reordered before matching occurs.

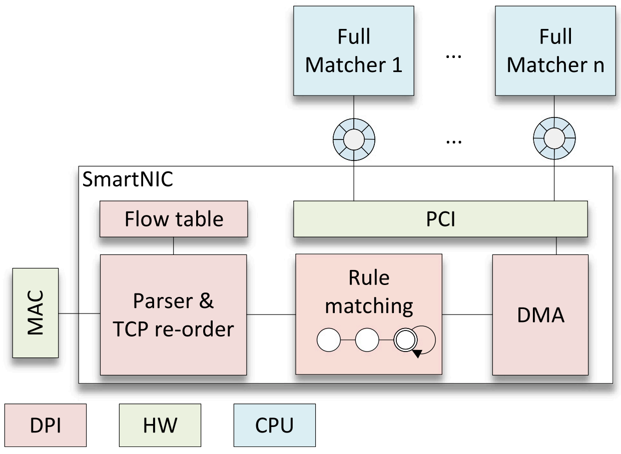


**Fig. 8.** High-level DPI architecture for IDS/IPS using FPGA-based SmartNICs (Chen et al., 2022). The SmartNIC matches sub-rules in a reduced graph. If a match exists, packets are forwarded to the CPU to complete the signature matching.

addresses, source and destination ports, requested Uniform Resource Locators (URLs), and protocol used, among others (Kumar and Sangwan, 2012). Well-known network security tools include a form of signature-based scheme such as NetSTAT (Vigna and Kemmerer, 1998), Snort (Roesch et al., 2011), and Suricata (Suricata, 2021). This software can run either in IDS mode (passive) or IPS (active) mode. In passive deployment, traffic is copied to the IDS and inspected. In active deployment, the IPS inspects the real traffic, which can forward or drop the packets based on the matching rules (Mukhopadhyay et al., 2011).

DPI is a widely used technique in modern security infrastructure that enables SmartNICs to implement signature-based IDS/IPS (Thinh et al., 2012; Orosz et al., 2018). The core purpose of DPI is to assemble and check application layer content by inspecting the packet's payload (Arshad et al., 2023). Fig. 7 shows how the matching pattern works at the packet-level and flow-level, also named stateless and stateful DPI, respectively. Signature-based IDS/IPS could detect malicious traffic by searching fixed strings or patterns in the payload. A pattern is defined by regular expressions, which are notations that describe a set of strings without explicitly enumerating them. Among the classical string matching algorithms, the most popular is the Aho–Corasick (AC) (Aho and Corasick, 1975), which can handle multiple patterns and guarantees $\mathcal{O}(n)$ time complexity for an input string of n bytes. This algorithm creates a DFA based on the input strings, thus consuming more memory as the number of rules increases (Pao et al., 2010).

Due to their reconfigurable architecture, FPGA-based SmartNICs are widely used for accelerating Deep Packet Inspection (DPI) in IDS/IPS systems. As shown in Fig. 8, packets are received at the SmartNIC and passed through a processing pipeline that extracts headers and reassembles flows based on TCP sequence numbers. The reassembled packets are then checked against DPI rules using hardware-implemented state machines for high-speed pattern matching. Matching results are forwarded to the host CPU for further inspection or action.

### 5.1.2. Literature

Leveraging a decision tree model, Jia et al. (2022) propose a high-performance Access Control List (ACL) engine on an FPGA-based SmartNIC. The rules—matching on source/destination IP addresses and ports are compiled into decision trees that are stored in the FPGA's on-chip SRAM. To accommodate deep and unbalanced tree structures, the trees are partitioned into subtrees and mapped across distributed memory interfaces, maximizing memory bandwidth utilization. A Network-on-Chip (NoC) interconnect and ring buffer are used to coordinate packet scheduling and ensure in-order delivery despite varying tree traversal latencies. This design enables the engine to sustain a throughput of up to 250 Mpps on rule sets of 100K entries. Building on this design, Xin et al. (2024) extend the FACL architecture by introducing HACL, a heterogeneous and modular ACL engine that separates decision tree traversal from rule matching. This separation enables a more structured and scalable pipeline, reducing architectural complexity by eliminating the need for recirculating paths and NoC-based scheduling. HACL maintains support for diverse decision tree algorithms (e.g., CutSplit, HyperCuts) and large rule sets without requiring FPGA reconfiguration. It achieves over 260 Mpps throughput on 100K-rule ACLs while reducing the memory footprint to approximately 2 MB through compact rule encoding and efficient engine utilization.

While the work in Jia et al. (2022) and Xin et al. (2024) achieves a flexible, high-performance ACL engine, it does not provide DPI functionality of signature matching on headers fields beyond the transport layer (i.e., payload). DeepMatch (Hypolite et al., 2020) presents a line rate DPI primitive in the data plane using a Netronome NFP-6000 SoC-based SmartNIC, showcasing the IDS application using Snort rules and QoS applications using Redis dataset. It transforms the regex rules and compiles them into deterministic finite automata (DFA) based on the AC algorithm. In IDS/IPS applications, it is a requirement to support TCP re-assembling for out-of-order (OoO) packets. DeepMatch supports both stateless intra-packet and stateful inter-packet regex matching capabilities. An updated per-flow state table is required for the latter since the packets have to be buffered and wait for serialization. The implementation details are depicted in Fig. 9. Incoming packets are buffered and dispatched to worker threads on the Flow Processor Cores (FPCs). When flow-level matching is desired, OoO packets are sent to a large external memory until they can be processed. The DeepMatch algorithm is natively implemented in Micro-C for the Netronome platform and integrated into the P4 processing pipeline with the help of P4 externs. The DFA memory maps and code are loaded using a runtime environment. DeepMatch can achieve 40 Gigabits per second (Gbps) in stateless mode and 20 Gbps in stateful mode.

Alternatively, researchers developed complete frameworks for signature-based IDS/IPS using FPGA-based SmartNICs. Pigasus (Zhao et al., 2020) implements IDS/IPS entirely in an FPGA-based SmartNIC. The architecture is mainly composed of a packet buffer, a parser, a reassembler module, a multi-string pattern matcher, and a DMA engine to move data from the SmartNICs to the CPU (see Fig. 8). This system mimics the NIC functionalities, such as primary packet processing, flow management, and data exchange with the host. Pigasus implements the packet reordering module for stateful pattern matching, similar to DeepMatch. It is performed to apply string matching on OoO packets. Pigasus leverages hierarchical filters to make its pattern-matching memory efficient. When a match is encountered, the system sends the suspicious traffic to enter full match expression mode in the CPU. It uses a memory-dense linked list data structure to track the flows and wait for serialization (see Fig. 10). Additionally, the TCP reordering module for OoO packets separates in-order, and OoO flows to improve performance, thereby avoiding additional latency for already in-order packets. Pigasus outperforms Snort running on a CPU, offering a 3x improvement in latency and requiring 65x fewer cores to maintain a 100 Gbps throughput.

Hu et al. (2023) present a real-time malicious traffic detection with the same capability of inspecting OoO packets using an FPGA-based SmartNIC. Similarly to previous works, the core functions are the TCP OoO reassembly and a hierarchical packet match. However, this work uses a Non-deterministic Finite Automata (NFA) instead of a DFA to compute the packet matching. A key insight shown in this work relates

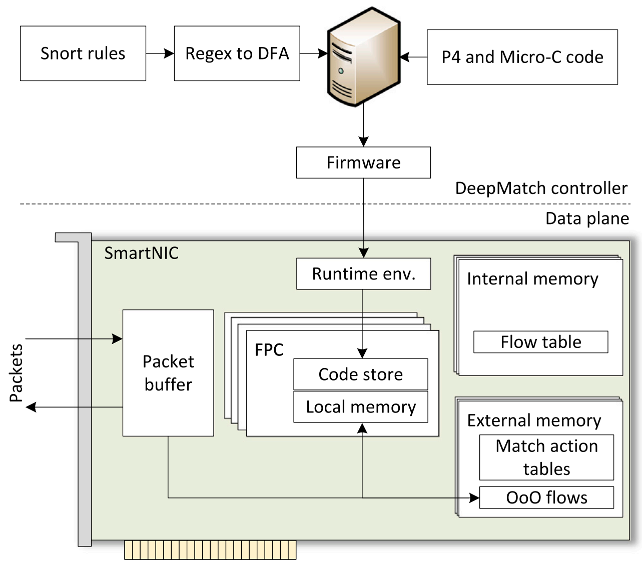


**Fig. 9.** DeepMatch is implemented on NFP-6000 to enable DPI by translating regex rules into DFAs that can be pushed to the data plane. DeepMatch uses internal and external memory to store packets that are out of order before it applies string-matching rules (Hypolite et al., 2020).

to the power consumption of every module in Fig. 8. The TCP reorder module consumes 50% of the total SmartNIC power, indicating that the hardware implementation of reordering is costly compared to the other modules.

Cheng et al. proposed Fidas (Chen et al., 2022), an FPGA-based IDS/IPS system. Following a similar approach to Pigasus, Fidas aims to develop an FPGA-based intrusion detection system that fully offloads rule pattern matching while introducing a novel traffic flow rate classification to the SmartNIC. Like Pigasus, Fidas employs a hierarchical pattern-matching approach but innovates in rule analysis and decomposition. The pattern-matching module uses a multi-level filter-based method for efficient regex processing. On the other hand, the flow rate classification module utilizes a novel dual-stack memory scheme to identify hot flows under volumetric attacks, such as DDoS or heavy hitters. This helps avoid the overhead of running DPI on flows identified as part of a volumetric attack.

In parallel to FPGA-based ACL acceleration, efforts have also been made to enhance software IDS performance using SmartNIC-assisted offloading. Sheeraz et al. (2024) integrate a Napatech SmartNIC with Snort to offload packet classification and filtering, reducing the workload on the host CPU. While the full Snort ruleset remains in host memory, the SmartNIC handles early-stage packet filtering based on IP, port, and protocol fields, effectively minimizing the volume of traffic needing deep inspection. This offloading allows the system to sustain 1 Gbps throughput under realistic traffic conditions, where only a small fraction (0.01%) of packets are malicious. The filter rules deployed on the SmartNIC are lightweight, enabling line-rate processing with minimal host CPU utilization.

### 5.1.3. *Comparison and limitations: signature-based IDS/IPS*

Signature-based IDS/IPS systems heavily rely on hardware to match a set of rules with the packet header and payload. This process often incurs overhead, primarily due to latency caused by memory access and scarcity of memory resources. To address this issue, Jia et al. (2022) propose two techniques. The first one is to fully utilize fast memory by distributing tree allocations across multiple memory interfaces, thus increasing bandwidth and reducing latency. The second one uses a greedy optimization algorithm to optimize space consumption by mapping the trees into parallel memory spaces.

**Table 5**
Comparison of signature-based IDS/IPS using SmartNICs. The reported performance varies depending on several factors, including rule set size.

| Work | Novelty | Performance | Model size | Rule set | Year |
|---|---|---|---|---|---|
| Hypolite et al. (2020) | Line-rate DPI primitive with P4 | 40 Gbps intra-packet matching | 4 MB | Snort, Redis | 2020 |
| Zhao et al. (2020) | Complete IDS/IPS in FPGA-based SmartNIC | 100 Gbps with 100K flows and 10K rules | 8 MB | Snort | 2020 |
| Chen et al. (2022) | Detection of flow rate for DDoS attacks | 100 Gbps for 800 bytes packets | 5 MB | Snort | 2022 |
| Jia et al. (2022) | High-speed ACL using decision tree | 250 Mpps with 100K ACL rules and 150 ns latency | 8.4 MB | 4-tuple ACL | 2022 |
| Hu et al. (2023) | Report per module power consumption | 3500 MB/s with 100% malicious traffic | 2 MB | Suricata | 2023 |
| Sheeraz et al. (2024) | Combining DPDK and hardware acceleration | 1 Gbps at 20K rules with low attack injection | N/A | Snort | 2024 |
| Xin et al. (2024) | Modularized ACL to decision tree | 260 Mpps with 100K ACL rules | 2.56 MB | 5-tuple ACL | 2024 |

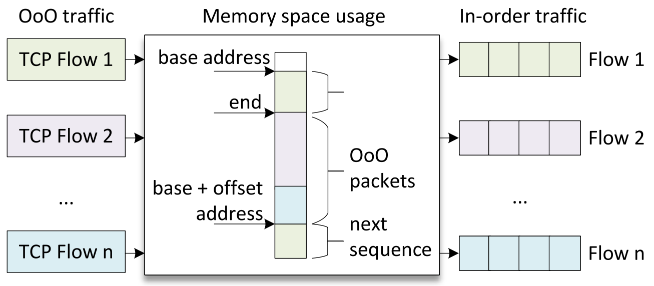


**Fig. 10.** Linked list memory usage is utilized for packet reordering (Hu et al., 2023). Flows with OoO packets are allocated in contiguous memory and referenced using pointers to the segment's starting position.

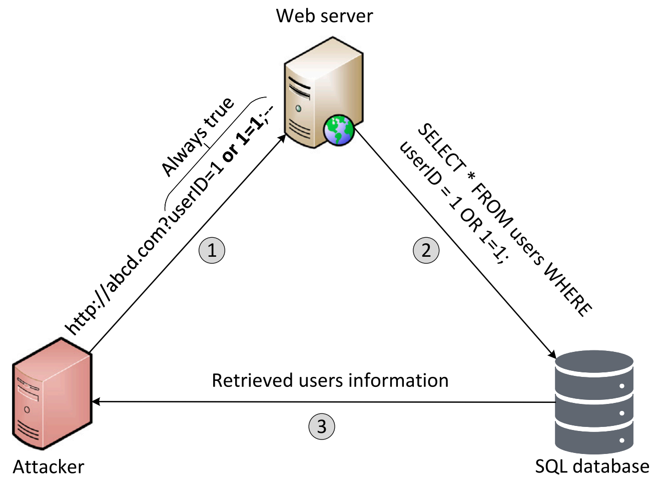

Web server
Always true
http://abcd.com?userID=1 or 1=1;--
SELECT * FROM users WHERE userID = 1 OR 1=1;
1
2
Retrieved users information
3
Attacker
SQL database

**Fig. 11.** Boolean-Based In-Band SQL Injection Attack. The attacker sends a crafted HTTP request containing a malicious condition (1 OR 1 = 1) that always evaluates to true (1). The web server incorporates this input into an malicious SQL query and forwards it to the database (2), which processes the query and returns all user records (3). The attacker then receives the retrieved data, effectively bypassing authentication and gaining unauthorized access.

DeepMatch (Hypolite et al., 2020) is also sensitive to memory latency. Although the Netronome architecture features multiple cores, multithreading cannot hide the latency overhead of reading bytes of the payload allocated in the external memory. Moreover, While up to 1024 bytes of payload can be buffered in fast memory, packets exceeding this size require additional memory access, which increases latency and degrades throughput. In contrast, FPGA-based SmartNICs benefit from deterministic memory access patterns and faster on-chip Block RAM or Ultra RAM, enabling rule lookups in a single cycle. However, this performance comes at the cost of very limited memory space, typically only a few megabytes, requiring careful model sizing and optimization.

The model size of a signature-based IDS/IPS typically corresponds to the memory required to store the DFA derived from a rule set. As Table 5 shows, this size can vary significantly depending on the number and complexity of rules. For instance, DeepMatch (Hypolite et al., 2020) employs a memory hierarchy with up to 4 MB of fast on-chip memory and 2 GB of external DRAM; however, performance degrades as the model size grows and accesses to slower external memory become more frequent. FPGA-based solutions like Pigasus (Zhao et al., 2020) and Fidas (Chen et al., 2022) keep their memory usage below 10 MB to fit within local fast memory, avoiding off-chip access that would impair performance. Conversely, hybrid systems that combine software and hardware (e.g., DPDK-based solutions) may not report precise model sizes, as part of the rule matching still depends on host-side processing.

Hu et al. (2023) evaluate IDS/IPS throughput under varying attack loads, showing that performance can range from 750 MB/s to 3500 MB/s as the share of malicious traffic increases from 20% to 100%. These insights underscore the importance of balancing rule complexity, memory constraints, and real-time performance when deploying IDS/IPS on SmartNICs. Table 5 presents a comparative overview of representative works, summarizing their novelty, throughput, model size, and rule configuration.

### 5.2. Anomaly-based

#### 5.2.1. Background

Generally, signature-based schemes have excellent detection accuracy for previously known intrusions. However, new vulnerabilities, also known as zero-day attacks, continue to emerge (Ahmad et al., 2023). These include threats with unseen signatures that have not yet been cataloged. Anomaly-based IDS/IPS systems offer an alternative by analyzing traffic behavior rather than relying on predefined signatures (Jyothsna et al., 2011). In these systems, legitimate system behavior is modeled using machine learning, statistical, or knowledge-based methods. Any significant deviation from the established model is flagged as an anomaly, which can then be classified as an intrusion (Scarfone et al., 2007). This approach assumes that malicious behavior differs measurably from normal user activity.

A concrete example is the detection of SQL injection (SQLi) attacks, where anomaly-based systems can identify abnormal database query patterns that deviate from expected application behavior. As illustrated in Fig. 11, an attacker can craft a malicious input (e.g., 1 OR 1 = 1) embedded within an HTTP request, which results in unauthorized access to user records once processed by the backend database. Detecting such abnormal request patterns in real-time can help mitigate application-layer attacks like SQLi. A widely used anomaly-based IDS is Zeek (2020), which can be combined with SmartNIC-based analysis for improved visibility. When integrated with host-based defenses, SmartNICs enable advanced threat detection by analyzing packet behavior and extracting features directly at the hardware level.

#### 5.2.2. Literature

Panda et al. (2021) develop SmartWatch, a cooperative model using P4 programmable data plane switches and Netronome SmartNICs for stateful flow tracking. The collaborative monitoring scheme uses a P4 switch to identify attack indicators at a coarse granularity. It is a “bump-in-the-wire” processing approach, wherein the P4 switch passively monitors traffic, computes switch queries, and steers traffic to the SmartNIC-Host subsystem only when further inspection is required, thus avoiding benign flows to suffer from additional latency. The SmartNIC role is to accelerate and track the flow state faster than the

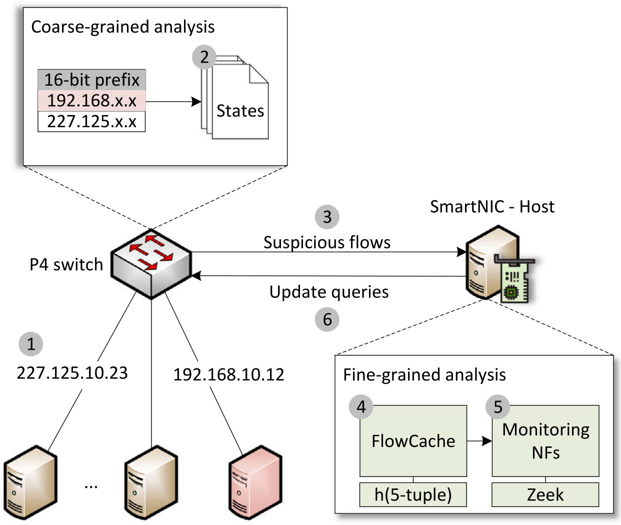


**Fig. 12.** High-level architecture of SmartWatch for cooperative monitoring (Panda et al., 2021). Multiple SSH connection attempts targeting a host are detected (1). P4 switch measures if the number of SSH connection attempts has been exceeded (2). Only a subset of the traffic is steered to the SmartNIC (3). These flows are tracked in a hash table created using a 5-tuple in the SmartNIC (4). Zeek monitors the malicious traffic in the SmartNIC and Host (5). The SmartNIC/Host subsystem updates the queries in the P4 switch (6).

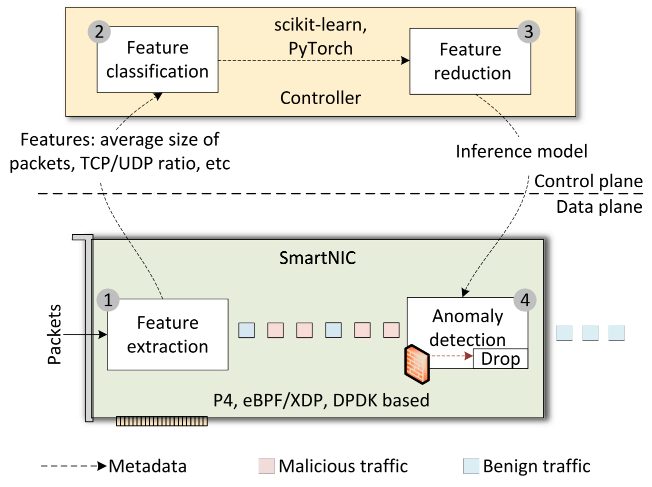


**Fig. 13.** In-network machine learning architecture for anomaly-based IDS/IPS using SmartNICs. Depending on the protocol, relevant packet fields, and metadata are extracted to add features (1). These features are then classified using supervised learning (2). Then, to reduce memory expenditure, the ratio of filtering rules and dropping benign traffic is optimized (3). The model is offloaded to the data plane, where it transmits or drops the packet according to the classification (4).

host. However, it has limited resources compared to the host in terms of both memory and processing resources. To address efficient state-tracking at high packet arrival rates, SmartWatch introduces a novel in-memory data structure and flow eviction policy on the SmartNIC. It utilizes SmartNIC's memory to design a FlowCache, comprising a hash table and ring buffers, to accommodate up to 25 million flow entries. Fig. 12 shows the proposed architecture in an SSH brute-forcing scenario. The P4 switch computes login attempts and compares them against a threshold. Only the subset of traffic that crosses the threshold is steered to the SmartNIC for further inspection in collaboration with the host. Logs are registered in the host if an anomaly is detected, and queries are updated to the switch to block the attack.

Pacífico et al. (2022) propose an IDS/IPS based on the Extended Berkeley Packet Filter (eBPF) combined with the XDP module. This approach enables the system to efficiently mitigate various types of attacks in real-time, including Structured Query Language (SQL) injection, malware, DDoS, and Peer-to-Peer traffic. The user provides the filters in a Function-as-a-Service model, executed sequentially in the hardware (i.e., SmartNIC). The scheduling algorithm follows a FIFO approach, with the SmartNIC immediately executing the first loaded filter and returning system information about the program's status to the user. Other filters are stored in a queue until the preceding filter completes execution. The filters, written in C, may include regex operations to detect specific attacks. The system achieves 10 Gbps for different filters, and the requests for adding a filter to the scheduler are about 500 ms and 900 ms on average, representing a significant overhead.

SmartNICs can be leveraged to deploy ML-based approaches to detect malicious behavior. Tasdemir et al. (2023) implement a SQL injection detector based on classical machine learning algorithms in the NVIDIA Bluefield-3 SmartNIC. Typically, specialized models are utilized for language detection and analysis, such as Natural Language Processing (NLP) (Gowtham and Pramod, 2021). However, SmartNICs lack the ML accelerators to run real-time complex models. Instead, several classical machine learning models, such as XGBoost and K-nearest neighbors, have been tested. Among the classifiers, the Passive Aggressive Classifier (Crammer et al., 2006) achieved the highest accuracy, scoring over 98%, outperforming all other models tested.

Xavier et al. (2021) developed an IDS based on decision trees implemented in P4 on Netronome SmartNICs. Using P4 to deploy the machine learning algorithm directly into the data plane drastically enhances such a system's performance. To overcome the limitations of the P4 language, which has limited computation capabilities, the authors translated the decision tree trained to a series of if-else statements that can be encoded in the P4 language. Moreover, the authors report a trade-off between per-flow and per-packet classification, stating that per-packet classification provides sufficient performance for distinguishing between benign and attack traffic. However, when higher accuracy is needed to identify specific attacks, per-flow classification is required. Fig. 13 shows the general architecture of implementing ML classifiers for anomaly detection and mitigation using SmartNICs.

Wu et al. (2022) proposed ONLAD-IDS, an on-device sequential learning approach with a supervised anomaly detector. Unlike purely supervised or unsupervised learning, the semi-supervised approach uses both labeled and unlabeled data to train the classifier. Additionally, it leverages sequential learning to update the model during runtime. The ONLAD-IDS architecture includes packet sniffing, a feature extractor, feature selection using Analysis of Variance (ANOVA), and the ONLAD model. A key advantage of this model is its continuous adaptive learning, which facilitates maintaining and updating the inference model at runtime.

### 5.2.3. *Comparison and limitations: anomaly-based IDS/IPS*

Anomaly-based IDS/IPS systems leverage SmartNICs either for fine-grained statistics collection (e.g., SmartWatch) or to offload inference models for real-time traffic classification. These approaches vary from standalone deployments, as seen in ONLAD-IDS (Wu et al., 2022; Tasdemir et al., 2023; Xavier et al., 2021), to cooperative architectures, like SmartWatch (Panda et al., 2021), which involve programmable switches, hosts, and SmartNICs working together to improve performance and granularity.

Some works explore serverless IDS architectures, where sequential filters run on SmartNICs across multiple tenants (Pacífico et al., 2022). However, this model introduces queuing and communication delays, especially under high traffic or attack conditions, such as volumetric DoS, where delayed execution can affect mitigation effectiveness. Despite these challenges, it provides a viable approach for multi-tenant SmartNIC usage while preserving data isolation.

**Table 6**
Comparison of anomaly-based IDS/IPS using SmartNICs.

| Work | Technique | Attack vector | | | | | Performance | Year |
|---|---|---|---|---|---|---|---|---|
| | | Brute force | Scan | DoS | SQLi | Malware | | |
| Panda et al. (2021) | Traffic analysis | ✓ | ✓ | | | | 43 Mpps with 64-byte packet | 2021 |
| Xavier et al. (2021) | Decision tree | ✓ | ✓ | ✓ | | | Above 95% and 100 ns delay | 2021 |
| Wu et al. (2022) | Semisupervised ML | ✓ | | ✓ | | ✓ | 90% detection rate and 1500 pkts/ms | 2022 |
| Pacífico et al. (2022) | Custom filters | | | ✓ | ✓ | ✓ | 10 Gbps with 64-byte packet | 2022 |
| Tasdemir et al. (2023) | Classical ML | | | | ✓ | | 95% detection rate and 0.0003 ms delay | 2023 |

Two main design patterns emerge across the reviewed systems. First, SoC-based SmartNICs like BlueField support familiar environments (e.g., Python), enabling direct deployment of classical ML models (Tasdemir et al., 2023). Second, data-plane-centric solutions run simplified or compiled models (e.g., decision trees) directly in the SmartNIC pipeline, offering lower latency and higher throughput (Xavier et al., 2021). Feature selection techniques, such as ANOVA-based ranking, are commonly applied to reduce dimensionality and model complexity, ensuring real-time classification. While many classical ML systems rely on static pre-trained models, ONLAD-IDS (Wu et al., 2022) introduces a key innovation by supporting sequential learning and online model updates directly on the SmartNIC. This enables the IDS to adapt dynamically to concept drift and emerging threats with minimal computational overhead, eliminating the need for full retraining or host involvement.

These systems target a broad spectrum of threats, including brute-force login attempts, DoS/DDoS floods, malware activity, and SQL injection. Table 6 summarizes representative anomaly-based IDS/IPS works, highlighting the techniques used, attack coverage, and key performance metrics.

### 5.3. Comparison with programmable switches: IDS/IPS

IDS/IPS leverage various techniques to achieve their goals, with two primary approaches being DPI and ML. Both techniques can be implemented on SmartNICs and programmable switches for intrusion detection and mitigation. Despite these similarities, there is a clear distinction between the use cases of programmable switches and SmartNICs.

ML detection schemes have been deployed to detect malware in programmable switches (Lee et al., 2024; Kapourchali et al., 2024). These works follow a similar approach, where incoming packets are parsed and hashed to create a flow. The payload of the packets corresponding to a flow is then sent to the control plane, where the switch CPU runs the ML model to classify the packets as either malicious or benign. After successful malware detection, the flow is blacklisted. Other works propose complete packet classification within the data plane, eliminating the need for the control plane to execute the ML algorithm, thus reducing the latency. In these cases, the control plane is primarily responsible for populating table entries and defining classifier logic, which is typically limited to decision tree algorithms (Poddar and Babu, 2022).

Moreover, instead of classifying packets using ML, DPI can be performed directly on programmable switches to detect malware. Due to inherent hardware constraints, the programmable switches are generally limited to inspecting packet headers. To address this, researchers and developers often rely on techniques such as packet recirculation, replication, and cloning to iteratively inspect traffic, achieving the required level of security. However, this comes at the cost of performance degradation (Gupta et al., 2023a). The computational and memory demands required for DPI are not well-suited for programmable switches. Consequently, these switches have primarily been used for security applications like parsing Domain Name System (DNS) headers, performing HTTP inspections, or building stateful firewalls by analyzing URLs (AlSabeh et al., 2022a; Gupta et al., 2023b). More complex rule sets, such as those found in Snort, present significant challenges and are not extensively used.

In contrast, SmartNICs support DPI applications relevant to signature-based IDS/IPS, which inspect packet payloads to match pre-existing signatures rather than focusing solely on packet headers. This enables SmartNICs to provide security beyond what programmable switches can offer, extending protection to the application layer and accommodating a wider range of variable packet headers and protocols. SmartNICs are particularly well-suited for DPI because they integrate accelerators that address the limitations of resource scarcity found in programmable switches.

Additionally, SmartNICs offer a broader range of ML algorithms for malware detection, as they have more memory and cores available. SoC-based SmartNICs, which run lightweight operating systems and can leverage Python for programming, provide ease of development and open up numerous possibilities for innovation. This flexibility allows researchers to defend against various attacks, including SQLi, which typically requires NLP schemes (Tasdemir et al., 2023).

### 5.4. IDS/IPS: summary and lessons learned

SmartNICs notably enhance the performance of network intrusion detection systems. FPGA-based SmartNICs are well suited for signature-based detection, and SoC-based SmartNICs offer more flexibility for machine learning anomaly-based detection. The key takeaways are:

- Offloading DPI to FPGA-based SmartNIC boosts the performance and reduces the CPU cycles to perform packet reordering and string matching. More importantly, this solution offers more scalability than increasing the number of CPU cores to keep up with the high-speed network environment.
- The performance of intrusion detection systems can be enhanced by collaborative inspection using programmable switches, SmartNICs, and the host. The hierarchical inspection reduces the CPU usage, only receiving the suspicious traffic that the SmartNICs cannot process (it represents around 5% in a representative dataset Stratosphere Lab, 2016).
- SmartNICs can run machine learning models leveraging anomaly-based IDS/IPS. However, the limited computational power of the ARM cores leads to suboptimal performance and necessitates that the programmer devise novel quantization and optimization technique. Current state-of-the-art widespeard SmartNICs lack ML accelerators and ML models generally run better on GPUs rather than CPUs.
- DPI over encrypted flows is ineffective due to the confidentiality of the data. One alternative is to bypass these flows or perform traffic analysis using anomaly-based methods (i.e., traffic analysis). Another approach is to trust the SmartNIC to decrypt all incoming traffic before the inspection.

## 6. Volumetric attacks

The second category in the taxonomy presents applications and techniques that leverage SmartNICs to defend the network against volumetric attacks. Emerging commodity SmartNICs can facilitate the detection and mitigation of these attacks while maintaining high accuracy and throughput. The advantageous position at the network edge and the multiple core architectures allow them to process packets at line rate. Furthermore, the high-bandwidth communication channel between the SmartNIC and the CPU permits offloading the rule filtering operations. This section categorizes volumetric attacks into heavy hitter attacks and DDoS attacks.

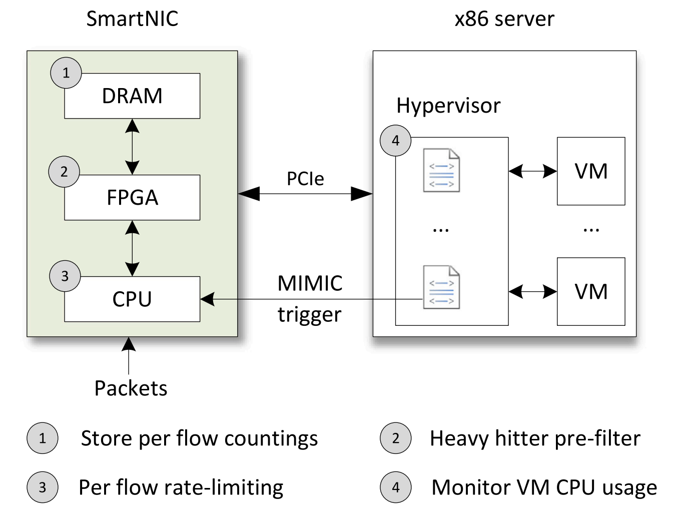


**Fig. 14.** Architecture for Heavy Hitter mitigation using SmartNIC and x86 server (Song et al., 2022).

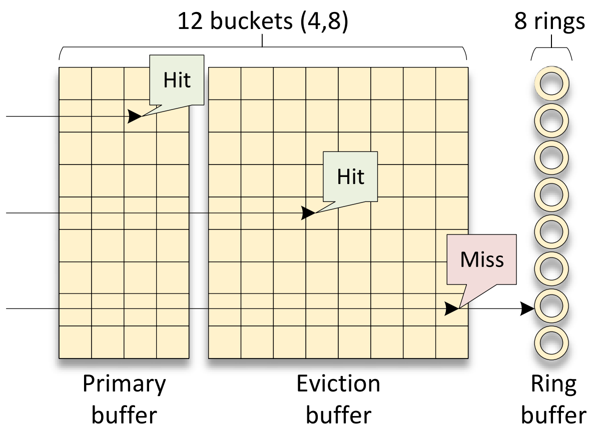


**Fig. 15.** Multi-level caching mechanism for HH using FlowCache design. The FlowCache data structure is allocated in a contiguous region of memory at compile time in DRAM memory.
*Source:* Reproduced from Panda et al. (2021).

## *6.1. Heavy hitter*

### *6.1.1. Background*

Heavy hitters (HHs) are flows that contribute significantly to the overall traffic on a link (Sivaraman et al., 2017). They require special attention, as they can disrupt the service of other flows or signal malicious activities, such as Denial of Service (DoS) attacks. Despite their impact, these large traffic volumes represent less than 10% of all flows in a data center network (Benson et al., 2010). However, east–west traffic is increasing with the emergence of new applications such as Generative AI and the Internet of Things (IoT), making it more challenging to detect these threats (Hsieh et al., 2024). Distinguishing between heavy hitters and mice flows (i.e., the number of packets in a flow below a defined threshold) quickly is essential for DoS detection, flow-size routing, and QoS. In the ever-growing multi-tenant cloud, resource sharing exacerbates this scenario, where CPU overload may lead to packet loss and higher latency, directly affecting all users' QoS. SmartNICs can detect HH by analyzing network traffic and mitigate the issue by rate-limiting or blacklisting heavy flows. This section introduces heavy hitter detection in the data plane using SmartNICs. Existing works leverage P4 with programmable hardware, which can also be implemented in SmartNICs.

### *6.1.2. Literature*

Standard HH detection algorithms violate programmable hardware constraints, as they require too many memory accesses per packet and are not well-suited for pipelined architectures. To address this issue, Turkovic et al. (2020) propose Sequential Zeroing, which fully leverages line-speed packet processing on programmable hardware by utilizing data plane programmability to detect heavy hitters. It emulates a sliding window for heavy hitter detection in high-speed data streams, minimizing both false positive and false negative rates. The sliding window is implemented through multiple consecutive sketches, with only the last sketch updated when a new packet arrives. Each sketch is divided into multiple stages, with the number of counters decreasing at each stage. Unlike other schemes that estimate packet counts for entire flows, Sequential Zeroing filters out lower-frequency flows in the earlier stages, allowing only higher-frequency flows (i.e., potential heavy hitters) to advance to the next stage. This layered structure reduces the memory required in the final stages, lowering the chance of collisions and optimizing memory usage. The multi-stage model aligns with the operational principles of programmable hardware, enabling efficient use of its limited resources. Sequential zeroing outperforms other state-of-the-art algorithm such as Count-min (Cormode and Muthukrishnan, 2005), HashPipe (Sivaraman et al., 2017) and PRECISION (Ben-Basat et al., 2018).

MIMIC (Song et al., 2022) proposed a new per-flow backpressure protocol and a heavy-hitter detection system to rate-limit selected heavy hitters and protect other tenants in x86-based server. The detection system uses a hierarchical memory design, leveraging both on-chip SRAM and off-chip DRAM to manage high concurrency without losing flow information. A pre-filtering procedure is added for rapid detection leveraging FPGA acceleration. The CPU queries the FPGA on demand to prevent CPU overload from frequent heavy-hitter reporting by the FPGA. The backpressure protocol is non-invasive, maintaining tenant privacy and allowing controllable rate-limiting through the novel use of meter tables. The SmartNIC serves as a mediator to facilitate heavy-hitter detection and per-flow backpressuring. Fig. 14 shows the architecture for HH mitigation in the cloud. CPU overloading at x86 server will trigger MIMIC on SmartNIC. SmartNIC CPU gets a set of heavy hitters from pre-filtering in the FPGA to be used by the backpressure protocol to determine the appropriate heavy hitters for rate-limiting.

Zhou et al. propose P4RSS (Zou et al., 2023), where the SmartNIC is positioned upstream of the CPU to perform stateful traffic load balancing and queuing flows based on real-time CPU utilization. Off-the-shelf x86 servers, commonly used as middleboxes in edge and public clouds, often employ Receive Side Scaling (RSS) to distribute packet flows across multiple CPU cores. However, RSS can lead to inter-core load imbalance by conducting stateless hashing without considering CPU core utilization, potentially overloading a single core with multiple heavy-hitter flows. By placing a P4-compatible device in front of the CPU, P4RSS performs stateful traffic load balancing based on real-time core utilization monitoring. It also offloads flow affinity maintenance and heavy hitter throttling to the P4-compatible device, freeing up CPU resources. P4RSS can be implemented as hyper-converged server switches or P4-based SmartNICs. Evaluation results show that P4RSS reduces the standard deviation of CPU core utilization by 22%–53% compared to RSS, enhancing middlebox stability and enabling higher CPU utilization without overprovisioning. The evaluation in a SmartNIC demonstrates that P4RSS outperforms RSS in preventing overload and optimizing server utilization.

SmartWatch (Panda et al., 2021) performs fine-grained traffic measurement leveraging their novel FlowCache. It utilizes both host and SmartNIC to track the flows in a lossless manner. This data structure consists of a hash table and ring buffers (see Fig. 15). Ingress packets update the hash table using the corresponding 5-tuple hash. The ring

**Table 7**
Comparison between HH works using SmartNICs.

| Work | SmartNIC architecture | Data structure | Mitigation |
|---|---|---|---|
| Panda et al. (2021) | SoC-based | Custom (FlowCache) | Dropping |
| Song et al. (2022) | FPGA-based | Hash table | Rate limiting |
| Turkovic et al. (2020) | SoC-based | Custom (Modulo Sketching) | N/A |
| Zou et al. (2023) | SoC-based | Hash table | Rate limiting |

buffers accommodate evictions from the hash table, up to 64 K entries. Using this scheme, SmartWatch supports heavy hitter and heavy change detection in a window of 5 s. The HH detection is triggered when a predefined threshold is exceeded, set at 0.001 percent of the total packets received during the monitoring interval, which can range from 2 million to 64 million packets.

### 6.1.3. Comparison and limitations: heavy hitter

Table 7 compares representative approaches to heavy-hitter detection and mitigation using programmable hardware, particularly SmartNICs. SmartWatch (Panda et al., 2021) introduces FlowCache, which, while scalable, is limited to tracking up to 25 million flows without packet loss. Sequential Zeroing (Turkovic et al., 2020) offers a memory-efficient approach to heavy-hitter detection by leveraging modulo sketching and periodic counter resets, but it comes with several limitations. Its fixed sliding window size requires careful tuning and may not adapt well to dynamic or bursty traffic patterns, potentially leading to missed detections. The method is designed purely for detection and does not incorporate any built-in mitigation mechanisms like rate limiting. Additionally, its accuracy can degrade in multi-tenant environments due to hash collisions and counter reuse, and it is sensitive to the quality of the hash function used. These constraints make Sequential Zeroing less suitable for environments requiring adaptive, real-time traffic management.

P4RSS (Zou et al., 2023) and MIMIC (Song et al., 2022) represent two SmartNIC-based approaches for mitigating heavy traffic in multi-tenant environments, but both exhibit design trade-offs. P4RSS introduces load-aware intra-server flow scheduling by dynamically steering packets to the least-loaded CPU cores while preserving flow affinity. However, it assumes homogeneous core performance and lacks support for dynamic flow migration, which may reduce adaptability under sudden load spikes. Its reliance on accurate CPU load feedback and the added overhead from ECN-based feedback and token-bucket enforcement also introduce complexity. Similarly, MIMIC focuses on protecting CPU resources by implementing SmartNIC-aided flow backpressure. It uses meter tables to detect and rate-limit heavy flows, enforcing fairness across tenants. While MIMIC offers fine-grained rate control and integrates well with ECN for congestion feedback, it also inherits limitations such as hardware-dependent tuning and potential queuing delays at the SmartNIC. Both systems require coordination with host-side software or control planes, which may affect real-time responsiveness and scalability across heterogeneous deployments.

## 6.2. DDoS

### 6.2.1. Background

DDoS attacks are a prevalent cybersecurity threat that aims to disrupt online services by overwhelming target systems with malicious traffic from different sources. Generally, these sources are legitimate devices connected to the Internet but are under the control of an attacker (i.e., botnets). These attacks are motivated by financial gain, hacktivism, and competitive advantage, resulting in a monetary loss that exceeds 22$ million per year in the US and continues to grow (Federal Bureau of Investigation, 2023).

Various types of DDoS attacks target both the network and application layers, and they can be broadly categorized into flooding attacks and protocol/logic attacks (Srivastava et al., 2011). Flooding attacks generate a vast number of packets and send them to the victim, overwhelming the system and preventing it from processing legitimate user requests. Examples of flooding attacks include SYN, UDP, and ICMP floods. In contrast, protocol-based attacks are more sophisticated and require less traffic. These attacks exploit vulnerabilities in the protocol stack to disrupt the network. For instance, application-layer DDoS attacks like HTTP floods establish HTTP sessions with the server, occupying available connections and overwhelming it with illegitimate requests (Haseeb-Ur-Rehman et al., 2023).

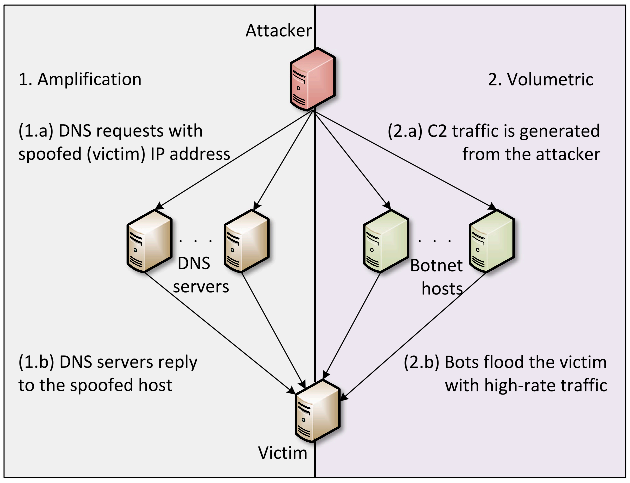

Attacker
1. Amplification
2. Volumetric
(1.a) DNS requests with spoofed (victim) IP address
(2.a) C2 traffic is generated from the attacker
. . .
DNS servers
. . .
Botnet hosts
(1.b) DNS servers reply to the spoofed host
(2.b) Bots flood the victim with high-rate traffic
Victim

**Fig. 16.** Common DDoS attack types. (1) Amplification: the attacker sends DNS requests with the victim's spoofed IP address, causing DNS servers to flood the victim with replies. (2) Volumetric: the attacker commands a botnet to generate high-rate traffic that overwhelms the victim's resources.

One particularly dangerous variant is the amplification DDoS attack. In this attack, the adversary spoofs the victim's IP address and sends small requests to third-party servers (e.g., DNS or NTP), which reply with much larger responses directed at the victim. This amplification effect can generate traffic volumes many times greater than the original request. Another common form is the botnet-based volumetric attack, where a network of compromised devices is remotely coordinated by the attacker via a command-and-control (C2) channel. Through C2 traffic, the attacker issues commands to launch a synchronized flood toward the target. As illustrated in Fig. 16, (1) shows amplification using spoofed requests, while (2) depicts a botnet flooding the victim directly.

### 6.2.2. Literature

Miano et al. (2019) propose offloading part of the DDoS attack detection algorithm to SmartNICs. This is accomplished by implementing a technique that tracks the top K malicious talkers and deploys filtering rules using eBPF/XDP on a Netronome SmartNIC. Their work demonstrates that DDoS mitigation relying on flow-based filtering schemes is less effective than signature-based approaches, where filtering rules are defined by combining data packet fields across multiple protocol layers.

Hinic et al. (2024) implemented a wire-speed DDoS mitigation system using the BlueField-2 SmartNIC. It leverages the embedded switch and DOCA Flow APIs to offload detection and mitigation of TCP SYN flood attacks directly onto the SmartNIC hardware. DOCA Flow provides a set of libraries that enable accelerated hardware use, allowing efficient packet processing and filtering. Using pipeline counters,

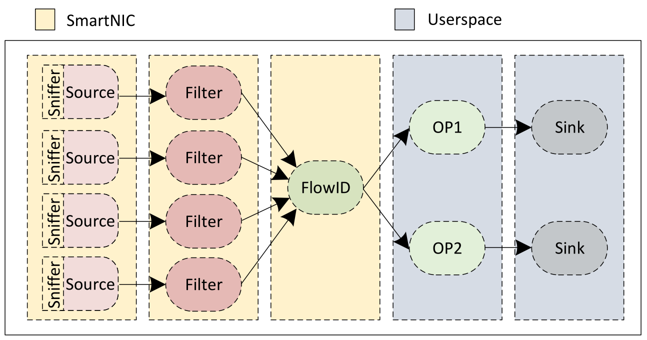


**Fig. 17.** High-level architecture of WindFlow tailored to SYN flood and port scanning defenses.
*Source:* Reproduced from Miano et al. (2024).

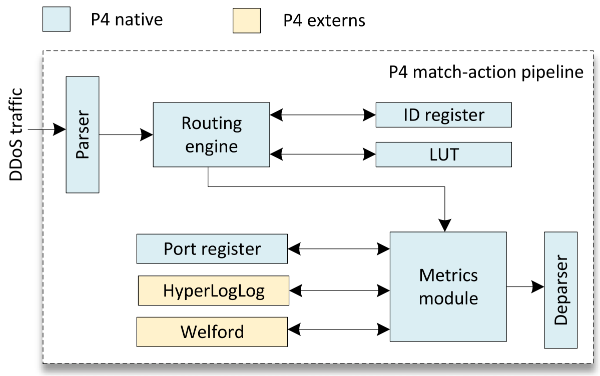


**Fig. 18.** DDoS mitigation framework proposed in Patetta et al. (2024).

the system detects SYN floods by monitoring per-flow statistics and identifying abnormal SYN-to-TCP ratios. Once an attack is detected, mitigation is enforced directly in hardware by dynamically updating filtering rules, ensuring minimal latency and CPU overhead. This approach enables efficient traffic classification and mitigation while maintaining high throughput up to 100 Gbps.

Dimolianis et al. (2021) propose a source IP-agnostic DDoS traffic classification and filtering schema that identifies malicious packet signatures via supervised ML methods and generates signature-based filtering rules. The mechanism continuously monitors the network and extracts relevant packet header features to create signatures using eBPF/XDP. The signatures collected are classified and reduced using ML and Pareto multi-objective optimization. Specifically, the system is tested using Random Forest and Multilayer Perceptrons algorithm. Reducing the number of malicious signatures drastically improves the throughput and lookup time in the mitigation module. The proof of concept was evaluated in a DNS volumetric attack scenario and compared against an IP-based mechanism, where the IP filtering performs better with low-rate DDoS but has scalability issues when the number of IP addresses increases.

Miano et al. (2024) employ WindFlow (stream processing library) on SmartNICs for network telemetry and analytics using eBPF. The implementation of the streaming algorithm in SmartNICs is depicted in Fig. 17. The restrictions imposed on the SmartNIC hardware capabilities necessitate that the authors split their design into two portions: (1) tasks involving repetition and intensive computations are offloaded to the SmartNIC, and (2) tasks running in the userspace that receive packets and additional metadata from the SmartNIC to perform the remaining of the processing. First, the SmartNIC captures packets from one or more network interfaces, parses them to extract the relevant fields (e.g., protocol, source and destination IPs and ports, etc.), and generates tuples. The tuples are later filtered based on user-defined rules (e.g., only TCP packets with SYN or SYN-ACK flags for SYN flood detection). Later, a map is used to compute the Flow Identifier (FlowID). In the userspace, the security application is implemented. For instance, in the context of SYN flood detection, the algorithm would count the number of incomplete TCP handshakes every few seconds. The authors showcase the effectiveness of their system on two security applications, namely detecting SYN flooding and port scanning.

Some works use FPGA-based SmartNICs to detect and mitigate DDoS attacks. Patetta et al. (2024) propose an anomaly-detection approach that leverages port-based features to detect botnet spreading. A change detection algorithm is tuned to report anomalies by analyzing long-term port usage. This work utilizes HyperLogLog to estimate cardinality based on hash functions and Welford's algorithm to estimate the mean and variance with a reduced memory footprint. Fig. 18 shows the P4 match-action pipeline implementations with their corresponding modules. The SmartNIC acts as a programmable data plane and network probe, managed by a controller to retrieve the metrics. The implementation copes with hardware constraints (e.g., division approximation) to overcome resource limitations.

Similarly, Salopek and Mikuc (2023) investigated a hybrid approach combining high-speed software filtering and FPGA-based SmartNICs. The role of the SmartNIC is to generate metadata while processing the packets for further usage in the Restricted Feature-set Packet Filter (RFPF), which is a software filter, and to perform a pre-filtering using an LPM algorithm that can be decomposed in stages. The filter rule balancing hardware and software uses a heuristic that considers the system's status. The SmartNIC passes the results of the prefiltering to the software that performs the packet drop or forwards it to the network. The improvement of using SmartNIC in the systems yields 20% fewer CPU cycles in random traffic and almost 30% in specific traffic.

Instead of standalone defenses at the end host, DDoS detection and mitigation for collaborative networks (e.g., Autonomous Systems) are proposed. The Federated Learning (FL) approach trains learning models collaboratively with distributed profiles of cyber threats. These profiles are not shared between nodes in the network, thus maintaining data privacy. The Federate Average (FedAvg) is the popular method for parameter aggregation. The aggregation tasks are offloaded to a SmartNIC, leveraging the multi-thread processors for multiple aggregations at a time.

Dimolianis et al. (2022) propose a Federated Learning schema for collaborative privacy-aware DDoS detection, addressing limitations in current multi-domain cyber security efforts. A third trusted party coordinates by aggregating machine learning models from collaborators based on private attacks and benign traces without exchanging sensitive data. Detected attacks are mitigated by efficient and scalable firewalls implemented within the eBPF/XDP data plane programmability framework. Unlike related efforts where collaborators exchange coarse-grained data or predefined static rules and focus solely on attack data, the proposed FL scheme enables DDoS detection using both benign and attack data without exposing private information. It creates machine learning models based on Multilayer perceptrons (MLP) with generalization capabilities to identify previously unseen benign and attack packets, leveraging an anomaly detection scheme.

### 6.2.3. *Comparison and limitations: DDoS defenses*

Table 8 compares SmartNIC-based DDoS mitigation systems in terms of their attack vector, detection scheme, filtering method, SmartNIC architecture, and limitations. Across the literature, three main axes of differentiation emerge: deployment scope (standalone vs. collaborative), detection strategy (entropy-based, anomaly-based, or ML classification), and filtering implementation (e.g., P4, eBPF/XDP, DOCA Flow).

**Table 8**
DDoS defenses comparison.

| Paper | Attack vector | Detection scheme | Filtering | Rules based on | SmartNIC architecture | Limitations |
|---|---|---|---|---|---|---|
| Miano et al. (2019) | Volumetric | Entropy | XDP | IP SRC/DST | SoC-based | Requires a load balancer between SmartNIC and host for optimal performance. |
| Dimolianis et al. (2021) | Amplification | Signature | XDP | DNS packets field | SoC-based | Methodology tailored to Amplification DDoS only. |
| Salopek and Mikuc (2023) | Volumetric | N/A | RFPF | Protocol, IP SRC/DST, Port | FPGA-based | Utilize CPU cores for filtering in software. |
| Dimolianis et al. (2020) | Volumetric | Symmetry analysis | P4 | 5-tuple | SoC-based | Heavily depends on accurate packet metrics. |
| Patetta et al. (2024) | SYN Flood | Anomaly | P4 | Ports, IP SRC/DST | FPGA-based | Cannot listen to multiple ports for detection. |
| Lai et al. (2020) | Volumetric | Entropy | N/A | Protocol, IP SRC/DST, Port | FPGA-based | Relies on the controller for detection using LSTM-RNN. |
| Hinic et al. (2024) | SYN Flood | Anomaly | DOCA Flow | Protocol, IP SRC/DST | SoC-based | Rigid match rules (IP spoofing vulnerable). |

Standalone SmartNIC-based defenses, such as those by Miano et al. (2019), Hinic et al. (2024), and Salopek and Mikuc (2023), commonly aim for high-throughput detection and mitigation, but they face notable limitations. Approaches relying on static rule matching, such as those using DOCA Flow or FPGA-based filters, are often rigid and susceptible to evasion techniques like IP spoofing. Systems that offload pre-filtering to the SmartNIC frequently require coordination with host-side software for final decision-making, introducing latency and potential bottlenecks. Additionally, rule-space limitations in frameworks like eBPF/XDP or P4 restrict scalability when handling large blacklists or complex match conditions.

In contrast to standalone SmartNIC-based defenses, collaborative frameworks such as those by Dimolianis et al. (2022) and Patetta et al. (2024) distribute detection and mitigation responsibilities across network domains. Dimolianis et al. (2022) employ federated learning for privacy-preserving DDoS detection across autonomous systems but face limitations such as reliance on a trusted aggregator and the performance constraints of eBPF/XDP, including limited rule space and inflexible update mechanisms. Patetta et al. (2024) focus on detecting botnet activity by aggregating long-term port-level features from distributed SmartNICs, yet their approach depends on periodic synchronization and centralized analysis, which can delay response under high-speed attack conditions. These collaborative strategies highlight the trade-offs between distributed visibility and coordination overhead. From an architectural standpoint, filtering mechanisms based on eBPF/XDP and P4 are programmable but constrained by memory map limitations, which challenge the scalability of blacklist-based defenses. Mitigation techniques such as top-K flow offloading (Miano et al., 2019) and IP-agnostic signature matching (Dimolianis et al., 2021) help address these issues, while FPGA-based SmartNICs offer low-latency lookups but often lack advanced protocol handling and visibility features like port mirroring.

Overall, SmartNIC-based DDoS defenses are evolving from fixed-rule filtering toward adaptive, context-aware systems. However, achieving scalable, low-latency mitigation still requires balancing between SmartNIC resource constraints, detection accuracy, and cross-domain coordination.

### 6.3. Comparison with programmable switches: volumetric defenses

Detecting and mitigating volumetric attacks in the data plane provides several orders of magnitude higher throughput and reduced latency than traditional methods, making programmable data plane devices ideal candidates for defending against such attacks. However, the roles of programmable switches and SmartNICs are distinct regarding traffic visibility and the resources available to run these applications.

Programmable switches are optimized for line-rate packet forwarding, limiting the complexity of per-packet operations. While P4-based defenses have been proposed to detect DDoS and heavy-hitter flows (AlSabeh et al., 2022b), these typically rely on memory-efficient sketches that reduce tracking granularity. The limited on-chip memory must be shared with forwarding functions, which can restrict defense accuracy or scalability. In practice, this necessitates spreading detection logic across multiple switches or pipeline stages (Bruschi et al., 2020).

In contrast, SmartNICs are deployed at the network edge and offer greater flexibility for volumetric attack mitigation. Their programmability, access to DRAM, and ability to run stateful filtering make them well-suited for detecting and mitigating high-volume attacks before they impact core systems. SmartNICs support low-latency inline filtering via frameworks like DOCA Flow and eBPF/XDP (Hinic et al., 2024; Miano et al., 2019), enable distributed defenses through federated learning (Dimolianis et al., 2022), and overcome the flow-tracking limitations of switches by leveraging greater memory capacity (Dimolianis et al., 2021). These architectural advantages position SmartNICs as a complementary and often more versatile alternative to switches in defending against large-scale DDoS attacks, capable of filtering malicious traffic before it reaches the core network, since SmartNICs are deployed closer to the traffic origination point (e.g., end devices or edge servers).

### 6.4. Volumetric attacks: summary and lessons learned

SmartNICs significantly enhance the detection and mitigation of heavy hitters and DDoS attacks. SmartNIC flexibility allows the deployment of different catching data structures that permit the count of flows effectively, tracking the amount of data traversing the network without compromising performance. Moreover, the strategic position of SmartNIC in the network edge allows for rate limiting and mitigating DDoS attacks by high-speed filtering before the DDoS propagates in the network. The key takeaways are:

- SmartNICs flexibility allows the deployment of innovative data structures in the data plane for summarizing and characterizing large volumes of data. Examples of these are Sequential Zeroing (Turkovic et al., 2020), SmartWatch (Panda et al., 2021), and MIMIC (Song et al., 2022).
- SmartNICs allow rate-limiting heavy hitters, preventing packet loss due to CPU overload. Also, it can distribute flows that are aware of the CPU workload context, optimizing CPU usage.
- SmartNICs, with their high-rate packet processing capabilities, effectively mitigate DDoS attacks by filtering out malicious traffic.
- SmartNICs support the detection of DDoS via standalone or collaborative schemes (e.g., federated learning). The latter helps to reduce the burden on the network by distributing mitigation across the network.

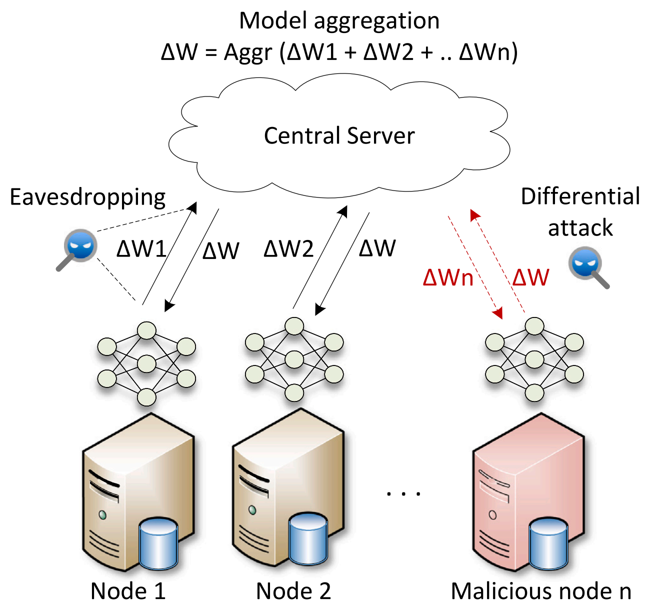


**Fig. 19.** Attacks on federated learning during model aggregation. A central server aggregates updates ($\Delta W_1$, $\Delta W_2$, …, $\Delta W_n$) from distributed nodes. Malicious nodes may perform eavesdropping or differential attacks to infer private data from shared model updates.

## 7. Anonymity and confidentiality

The third category in the taxonomy focuses on the applications that leverage SmartNICs to ensure communications privacy. The first subsection describes the anonymization techniques and privacy protection frameworks running on SmartNICs. The second subsection describes cryptographic suites and enhanced security protocols using authentication schemes with SmartNICs.

### *7.1. Privacy and anonymity*

#### *7.1.1. Background*

Modern communication relies on the Internet to exchange data across the globe. Consequently, user activity is often monitored to extract personal information for security or commercial purposes. Privacy protection and anonymization at the network layer aim to prevent data leakage by obfuscating user identities as packets traverse the network. This is commonly achieved by modifying or encrypting packet headers to make it infeasible for adversaries to associate packets with specific users, even when payloads remain unencrypted.

Technologies like Blockchain and FL are gaining traction for privacy-preserving applications. Blockchain is widely used in cryptocurrencies (Nakamoto et al., 2008) and has expanded into sectors such as healthcare and transportation (Sunny et al., 2022). In these domains, SmartNICs can enhance both security and performance by offloading cryptographic operations such as encryption, hashing, and consensus protocols from the host CPU.

Similarly, in FL, SmartNICs can accelerate secure aggregation and communication processes by handling model update encryption, local inference, or gradient processing. This offloading reduces latency and helps preserve privacy in distributed learning environments. However, even in FL settings, privacy risks remain. As illustrated in Fig. 19, attackers may conduct eavesdropping or differential attacks during model aggregation, attempting to infer sensitive data from the transmitted or injected model updates ($\Delta W_1$, $\Delta W_2$, …, $\Delta W_n$). Emerging cryptographic techniques, such as homomorphic encryption and secure multiparty computation, offer promising solutions, enabling computation on encrypted data and minimizing the risk of information leakage even within the SmartNIC.

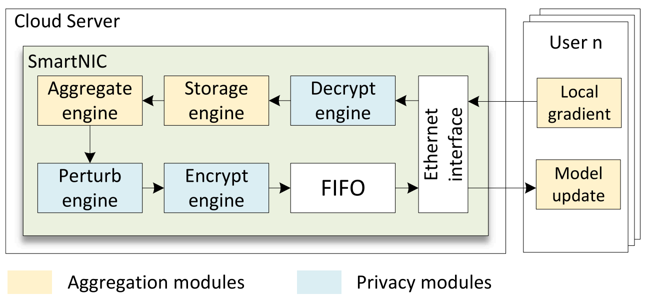


**Fig. 20.** Secure aggregation scheme using SmartNIC (Zang et al., 2022).

#### *7.1.2. Literature*

Micovic et al. proposed LISPP (Mićović et al., 2023), a system which leverage Format Preserving Encryption (FPE) for privacy protection on the network layer. FPE enables the encryption of bits of arbitrary lengths while maintaining the same alphabet, making it suitable for encrypting packet header fields without affecting their size. The algorithm used in this work is the FF3-1 specification (Dworkin, 2019), running on a Netronome SmartNIC. SmartNICs are used on the server side to obfuscate packet headers, such as IP and port source/destination, thus providing a tunneling effect for packets while traversing the external network. The authors implement it in three stages: purely in P4, mixed P4 and Micro C, and purely C, with the latter achieving line-rate performance.

Similarly, Sacco et al. (2023) propose P4FL, a federated learning system for P4-compatible SmartNICs. The solution utilizes the Multiprotocol Label Switching (MPLS) packet field to define and transmit model parameters and gradients. However, due to the limitations of P4 in performing division and floating-point arithmetic, certain operations are offloaded to the CPU rather than handled within the P4 pipeline. While this work effectively implements a federated learning system, the gradients, and parameters are transmitted in clear text over the network, making it vulnerable to differential attacks (infer node information by injecting predefined gradients and analyzing the output model)

Therefore, Zang et al. (2022) propose a secure aggregation scheme for FL using FPGA-based SmartNICs. Fig. 20 shows high-level components of the design both on the user and server side. On the user side, a global network model and initial parameters are downloaded from the server. Then, the gradients are calculated and encrypted to send over the network to the server. On the server, multiple encrypted gradients are received from n users. SmartNICs decrypt the local gradients and aggregate them to obtain the global gradient. Additionally, Gaussian noise is added to preserve the privacy of the local gradients. After encrypting the global gradient, the users receive and update the model for further training.

Similarly, Pan et al. (2024) propose Flagger, an FL aggregation system that leverages Homomorphic Encryption (HE) to preserve the privacy of the data on the SmartNIC. In this approach, the SmartNIC operates over an encrypted ciphertext to aggregate without decrypting the gradients. The encryption leverages an additive property of the HE to operate over it without decrypting the gradients, thus improving data privacy even from the SmartNIC execution environment. Flagger reduces aggregation time by 400% compared with CPU-centric aggregator architecture using the same cryptographic algorithm.

In IoT scenarios, authentication and anonymity are essential to protect device-to-device (D2D) communications. Bakar et al. (2024) propose DPUAUT, a SmartNIC-based D2D authentication mechanism, also called the intelligent swarm systems broadly used in IoT. It proposes a lightweight authentication scheme for secure communication in autonomous IoT networks, utilizing Physical Unclonable Functions (PUFs) to boost security against physical attacks. The scheme enhances

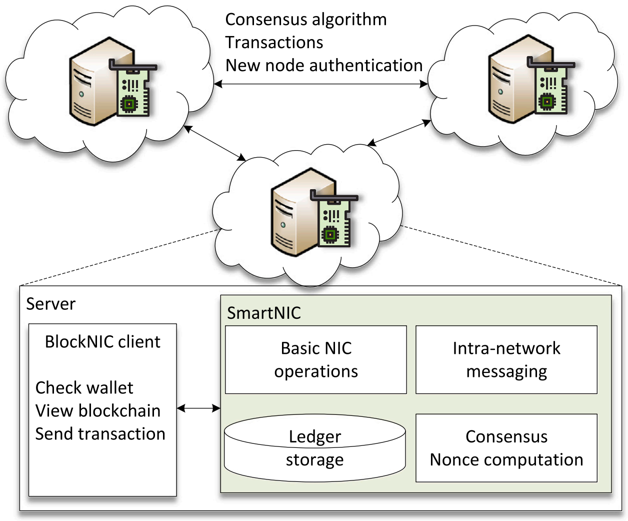


**Fig. 21.** Multi-node abstraction for Blockchain using SmartNICs (Kapoor et al., 2023).

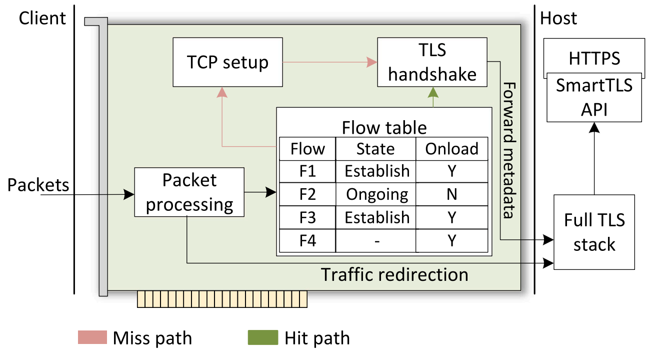


**Fig. 22.** Architecture of the SmartTLS. *Source:* Reproduced from Kim et al. (2020).

performance by offloading PUF-verifier logic to a SmartNIC, which reduces authentication latency, improves scalability, and strengthens security. In terms of privacy, the IDs of the devices are masquerading utilizing a session-specific random number along with the corresponding ID, thus making it unfeasible for an adversary to determine if the messages are for the same session or device. The authors conduct informal and formal security analyses, including the Real-or-Random model, to verify the scheme's robustness.

Kapoor et al. (2023) proposed a SmartNIC blockchain deployment called BlockNIC. The system comprises three main modules: transactions, blocks, and the blockchain. The transaction stores the data, including the information on the amount of currency, time, sender IP address, and signature. The Block stores up to 20 transactions; it includes the index, time, and hash of the previous block. In this system, the consensus algorithm is the proof of work (Sriman et al., 2021), which follows "the easy-to-verify, hard-to-find" principle. Finally, the blockchain stores all the metadata related to the blocks, such as the number of blocks, transactions, and hash of the last operation. Fig. 21 shows a high-level abstraction of a BlockNIC. Each node is a bare-metal server equipped with a SmartNIC, and the SmartNIC continuously runs the BlockNIC program to mine and maintain its own copy of the ledger. SmartNICs effectively offload the blockchain processing from the main server infrastructure using a C-based blockchain implementation.

### 7.1.3. Limitations and comparison: privacy and anonymity

Several SmartNIC-based solutions aim to enhance privacy and anonymity through in-network encryption, FL, and blockchain offloading, yet each introduces specific trade-offs. Stateless encryption schemes like LISPP (Mićović et al., 2023) avoid mapping tables and preserve routing functionality, but key rotations can cause packet drops unless perfectly synchronized across all ingress and egress points of the network. P4FL (Sacco et al., 2023) enables gradient exchange in-network using MPLS headers, yet the lack of encryption exposes updates to differential attacks or eavesdropping. This can lead to privacy leakage (e.g., membership inference). Secure aggregation systems such as Zang et al. (2022) rely on trusted SmartNIC environments to decrypt and perturb gradients, offering lower overhead than HE but still requiring on-device decryption. Flagger (Pan et al., 2024) advances privacy by operating on ciphertext using HE across DPUs, but its complex coordination and high hardware demands introduce scalability challenges.

Kapoor et al. (2023) offload the entire blockchain infrastructure to SmartNICs. However, this approach has two main limitations. First, the computation is performed using the wimpy ARM cores on the SmartNICs, which are then compared to a bare-metal server. Typically, blockchain computation is offloaded to GPUs or FPGAs instead due to the high parallel computation required. Second, the implementation in C, along with the use of OpenSSL for user-space cryptography, does not fully exploit the capabilities of the SmartNIC, leading to suboptimal performance. Overall, while SmartNICs provide unique edge-computing advantages, their constrained resources, coordination complexity, and trust assumptions remain key bottlenecks.

## 7.2. Cryptography and security protocols

### 7.2.1. Background

In the data center, data can exist in one of three states: in use, in transit (or in motion), and at rest. Safeguarding against unauthorized access is paramount in each state. Data in motion refers to data being transmitted across networks and stored at rest on physical media such as databases or storage arrays. Encryption is essential to protect against data breaches and leakage in both states. SmartNICs can help reduce the CPU cycles required for these encryption tasks, particularly for data in motion. While data traverses a network, it is vulnerable to threats like eavesdropping, packet modification, and corruption. Cryptographic services are employed to mitigate these risks and provide security. TLS, defined in RFC 5246 (Dierks and Rescorla, 2008), is a protocol that ensures privacy between communicating applications and their users by encrypting network connections at the application layer to provide secure end-to-end communication.

Moreover, protocols can benefit from the crypto capabilities of SmartNICs to enhance security. Currently, data centers are equipped with high throughput (10 Gbps) and low-latency (microsecond scale) distributed storage, increasing the network tax on the CPU for I/O-related tasks (Xu et al., 2015). Solid States Disk (SSD) uses specific protocols like Nonvolatile Memory Express (NVMe), serving intensive I/O applications such as relational and non-relational databases (Guz et al., 2017). An extension to this protocol, NVMe over Fabrics (NVMe-oF), enables access to disaggregated memory over the network, providing scalability and achieving near-local latency and throughput (Ng et al., 2024). Emerging technologies such as RDMA and its infrastructures, including InfiniBand (IBA) and RDMA over Converged Ethernet (RoCE), address this issue by offloading I/O tasks to RDMA-capable SmartNICs, bypassing the CPU and allowing direct memory access. Unfortunately, studies have shown that RDMA one-sided operations are subject to security risks, such as packet eavesdropping, injection, and tampering (Chen et al., 2024). To address these threats, SmartNIC programmability allows researchers to investigate protocol enhancements that provide authentication and confidentiality over RDMA. Securing RDMA is a crucial research topic due to its extensive deployment in cloud environments and the demands of new data-hungry applications that need remote storage access with minimal performance overhead.

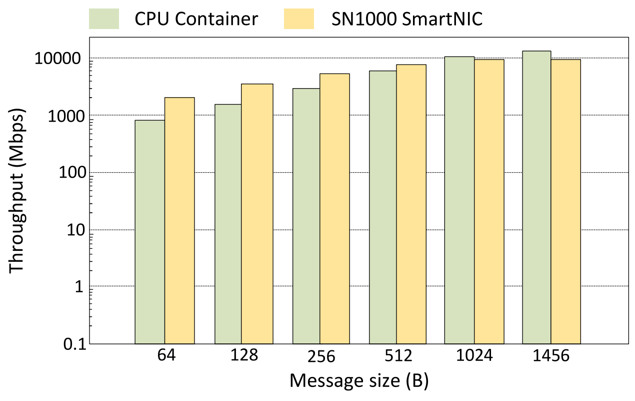


**Fig. 23.** SmartNIC vs. CPU with accelerated instruction for TLS offload. *Source:* Reproduced from Hussain et al. (2023).

### 7.2.2. Literature

TLS operations are divided into key exchange and encrypted data transfer, using asymmetric and symmetric encryption respectively. The key exchange has a high performance overhead, especially when connections are short-lived sessions. The bulk encryption of the data has a relative performance overhead, depending on the size of the delivered data. SmartTLS (Kim et al., 2020) aims to offload the handshake process on SmartNICs while leaving the CPU for encrypting/decrypting the data transfer. The SmartNIC serves as a TCP endpoint to track connections in a flow table while redirecting traffic to the host depending on hit-or-miss rules. Fig. 22 shows the architecture of SmartTLS in detail. When a new packet arrives, the packet processing module determines whether the packet will be forwarded to the host (traffic redirection) or processed by the SmartNIC. The SmartNIC then checks the connection state in its flow table. If the packet has the SYN flag activated, the SmartNIC sets up the TCP connection. If a match is found in the table (hit path), the SmartNIC either initiates the key exchange or, if the SmartNIC is overloaded (Onload = Y), delegates the task to the CPU.

Zhao et al. (2023) evaluated SoC-based SmartNICs executing these primitives in real-world use cases such as VPN tunneling, user authentication, and secure web services. Their results demonstrated reduced latency due to offloading, but also revealed throughput limitations under heavy workloads. This evaluation highlights the performance trade-offs in TLS operations across different deployment contexts and reinforces the benefits of offloading asymmetric cryptographic primitives to SmartNIC hardware.

SmartNICs can be used to completely offload the TLS suite. Typically, TLS runs in the user space, as user code is considered safer (a crash in a user process only affects that process). Kernel TLS (kTLS) leverages the kernel space to enhance the performance of TLS. This enables the use of TLS offload in SmartNICs, such as in Bluefield (Tuaf and Gilboa, 2020). These SmartNICs incorporate ASIC accelerators to offload crypto tasks that save CPU cycles on the host, thus boosting the performance over software-based kTLS implementations. Novais and Verdi (2024) studied the performance of kTLS and provided key insights for offloading kTLS. The performance trade-off is clear between selecting different offloading schemes: Software kTLS and Inline. Hardware offloading (inline) shows the best latency and power consumption performance.

Kottur et al. (2022) implemented the ChaCha20 algorithm on a Netronome SmartNIC and tested its performance. ChaCha20, an enhanced version of Salsa20, is supported in TLS version 3, making it relevant for today's applications. The authors used crypto externs to implement ChaCha20 on the SmartNIC. They discuss several limitations tightly coupled with the constrained resources of the SmartNIC. For example, the placement of crypto code alongside other offloaded applications depends on the available CPU and memory resources within the

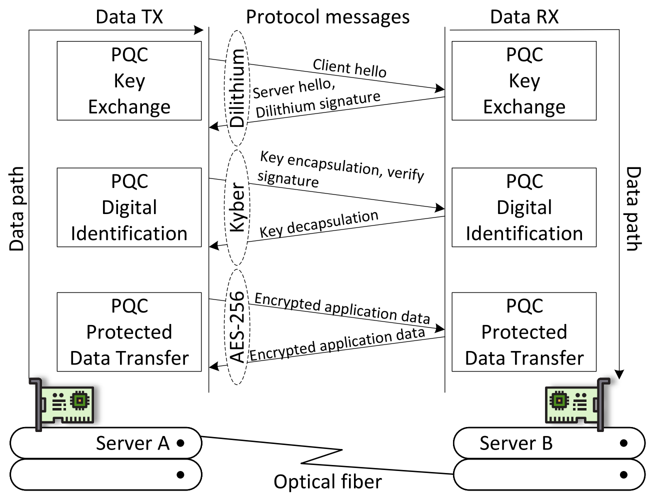

Data TX
Protocol messages
Data RX
PQC Key Exchange
PQC Key Exchange
Dilithium
Client hello
Server hello, Dilithium signature
Data path
Data path
PQC Digital Identification
PQC Digital Identification
Kyber
Key encapsulation, verify signature
Key decapsulation
PQC Protected Data Transfer
PQC Protected Data Transfer
AES-256
Encrypted application data
Encrypted application data
Server A
Server B
Optical fiber

**Fig. 24.** PQC resilient end-to-end encryption using SmartNICs (Aguilera et al., 2023).

SmartNIC. Additionally, the approach cannot handle messages longer than 256 bytes due to memory constraints in the parser. Furthermore, the lack of support for modulus operations in the hardware prevents the system from using ChaCha's standard authentication algorithm.

Hussain et al. (2023) proposed a microservice architecture for in-network security on an FPGA-based SmartNIC. The framework aims to provide scalability to security services for 5G/6G networks and server applications such as HTTP secure (HTTPS) and TLS. Decomposing the security algorithm in microservices and running it independently helps code reusability, hardware maintenance, debugging, and management of components for scaling. Fig. 23 shows the performance of offloading TLS to an FPGA-based SmartNIC. The end-to-end latency improves between 88% to 97% with the SmartNICs. However, the CPU of the FPGA is a bottleneck for packet sizes above 512 bytes, hence reducing the throughput.

Pismenny et al. (2021) propose a scheme for autonomous offloading layer 5 protocols such as TLS. The approach considered in this work does not need to change the existing TCP/IP stack, making no need to recompile the source code of applications. Offloading both encryption/decryption and authentication, the paper shows that the benefit of transmitting is higher than receiving, while both alienating the CPU utilization by 1.7x. Also, this work is extended to support other applications on top of layer 4 and pretend to offload tasks autonomously to the NIC without changing the software stack and recompiling software.

Emerging networking faces two big concerns: the necessity of augmenting the security (i.e., longer encryption key size) and the urge to produce new algorithms for Post Quantum Computing (PQC). The era of PQC poses new challenges to developing secure channels using traditional cryptography algorithms. SmartNIC programmability allows developers and security experts to easily deploy new algorithms or fix the ones that are no longer secure, for example, by augmenting the key size. Aguilera et al. (2023) propose a SmartNIC-based end-to-end PQC (see Fig. 24). SmartNICs execute all the networking functions required to establish a link with Dilithium for digital signature and Kyber for key encapsulation method, which have been proven to be resilient against PQC threats, combined with classic Advanced Encryption Standard (AES) with 256-bit block size encryption. The high processing requirements of PQC algorithms are met by offloading their functions to dedicated onboard ARM processors, which provide hardware accelerations. The evaluation results indicate enhancements in CPU cycle efficiency for Dilithium key functions, with improvements ranging from five to 31 times. Similarly, Kyber key functions exhibit improvement rates between eight and 57 times (see Table 9).

Moreover, SmartNIC can be used to provide security to RDMA protocol. Several studies highlight the vulnerabilities of the off-the-shelf

**Table 9**
Performance characteristics of cryptographic algorithms offloaded to SmartNICs.

| Work | Algorithm | Throughput | Latency | Notes |
|---|---|---|---|---|
| Kim et al. (2020) | RSA-2048/4096, AES-GCM | 13.3K TLS handshakes/s | 8 ms (99% @ 16 flows) | Offloads TLS handshake using PKA |
| Zhao et al. (2023) | AES-GCM, SHA-256 | 739 Mbps, 0.5 GB/s | 0.36 ms | Lower latency for short-lived flows; limited throughput under load. |
| Novais and Verdi (2024) | AES-GCM | 9.36 Gbps in line | 4 ms | Metrics increases with the number of connections |
| Hussain et al. (2023) | AES-GCM | 8.6 Gbps | 5.89 μs | Modular AES offload for microservices |
| Kottur et al. (2022) | ChaCha20 | Up to 6.8M msg/s | 21–170 μs | P4 and micro C implementation |
| Pismenny et al. (2021) | AES-GCM | 3.3x higher | 74% lower | Compared to Linux OpenSSL AES-GCM |
| Aguilera et al. (2023) | Kyber, Dilithium | 17 Mbps, 2 Mbps | N/A | Post-quantum key exchange (Kyber), digital signature (Dilithium) |

RDMA protocol (Tsai et al., 2019; Rothenberger et al., 2021; Taranov et al., 2022). The main vulnerabilities of the protocol include the lack of encryption and authentication mechanisms, a weak CRC checksum that uses known seeds or polynomials, a small packet sequence number (24 bits), and the absence of logging for one-sided operations, which makes detecting attacks challenging. Simpson et al. (2020) propose a framework for securing RDMA and describe the challenges to achieving security. The paper identifies three key vulnerabilities: lack of isolation, inauditable read and write operations, and susceptibility to DoS. Current virtualization techniques are insufficient for adversary isolation, highlighting the need for an IDS/IPS to address this challenge. Additionally, the read operation does not leave any trace, making it impossible to audit in the event of an attack. Lastly, an attacker could disrupt the entire network by injecting spoofed packets. To mitigate these issues, the authors propose solutions such as encrypting RDMA, using SmartNICs to log one-sided operations to prevent unauthorized reads, and employing micro-segmentation to isolate traffic.

Zeta (Chang and Mukherjee, 2024) introduces a zero-trust, transparent add-on for RDMA, addressing gaps in traditional SmartNIC implementations. Most SmartNICs support inline IPSec to ensure data secrecy and authenticity for RDMA applications. However, IPSec alone obscures the benefits of zero-trust principles and limits granular policy control at the IP layer. For instance, if multiple RDMA applications share the same IP address, applying differentiated policies to each traffic flow becomes challenging. Zeta resolves this limitation by incorporating an access control unit into SmartNIC. When two RDMA endpoints attempt to establish a session, they must negotiate parameters. Zeta enhances security by blocking connection requests if the two applications are not permitted to communicate, effectively implementing access control. Using the SmartNIC's CPU cores, Zeta processes packets on a per-flow basis. Given that this path may be slower and could impact performance, Zeta offloads subsequent packets within the same bidirectional session to the SmartNIC's flow offload engine (the fast path) following the initial negotiation. Since IPSec lacks zero-trust capabilities and remains vulnerable to impersonation attacks, Zeta leverages SmartNIC's PKA to tag packets with public/private key pairs, reinforcing security. Testing Zeta with real-world applications shows that while its cryptographic verification adds a session startup latency of 1.5 ms, it minimally impacts end-to-end performance, causing less than a 1% reduction in throughput and a 5% increase in latency.

Taranov et al. (2020) propose sRDMA, a secure RDMA protocol that addresses the vulnerabilities highlighted by ReDMArk (Rothenberger et al., 2021). sRDMA is designed to provide efficient authentication and encryption for RDMA networks, which typically transmit messages in plaintext. This plaintext transmission makes RDMA networks vulnerable to eavesdropping, allowing attackers in the same network to read or write arbitrary memory locations based on the information they

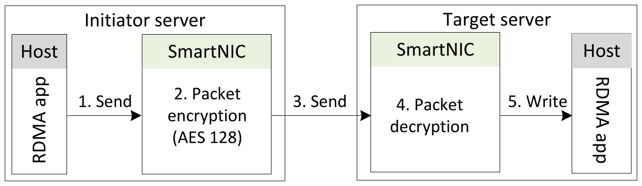


**Fig. 25.** General overview of sRDMA (Taranov et al., 2020). QP connections are established between host/SmartNIC and SmartNIC/SmartNIC prior to exchange data.

intercept. To enhance RDMA security, sRDMA introduces a secure reliable connection queue pair (QP) that utilizes symmetric cryptography for data authentication and encryption. Rather than relying on IPSec to encrypt the entire RDMA packet, sRDMA adds a new header field immediately before the payload, defining authentication and secrecy for the header itself. Additionally, it provides an option to encrypt the payload, though this introduces extra overhead. Fig. 25 illustrates the sRDMA write workflow, showing how the protocol uses SmartNICs at the endpoints between hosts to ensure robust data authenticity. Performance evaluations of sRDMA demonstrate improved efficiency over IPSec, particularly in nonce calculation and header size.

### 7.2.3. Limitations and comparison: cryptography and security protocols

SmartNICs enable efficient offloading of cryptographic operations, reducing CPU load and improving latency for protocols like TLS, PQC, and RDMA. In the context of TLS, offloading handshake and encryption tasks to SmartNICs significantly improves performance for short-lived sessions, as shown in SmartTLS (Kim et al., 2020) and kTLS inline implementations (Novais and Verdi, 2024). However, using the SmartNIC as a full TCP endpoint introduces overhead, especially under high concurrency or limited NIC-side compute resources. Autonomous and microservice-based approaches (Pismenny et al., 2021; Hussain et al., 2023) further enhance scalability, though they demand careful workload orchestration to prevent bottlenecks. For emerging cryptographic algorithms, SmartNIC programmability allows execution of ChaCha20 (Kottur et al., 2022) and post-quantum cryptography (PQC) schemes like Kyber and Dilithium (Aguilera et al., 2023). These enable future-proof secure channels, but implementation is constrained by missing arithmetic primitives (e.g., modulus) and high computational overhead, especially for PQC key operations. Lastly, SmartNICs improve RDMA protocol security beyond traditional IPSec by enabling lightweight, flexible offloads. sRDMA introduces extra overhead due to the additional authentication headers, which may impact performance in latency-sensitive deployments, while Zeta's access control logic, though effective, exposes an attack surface through SmartNIC eSwitch table exhaustion during session setup (Taranov et al., 2020; Chang

**Table 10**
Threats at different domain levels.

| Domain level | Threats | Remediations |
|---|---|---|
| Vendor | Hardware trojans in third-party chips<br>Physical attacks (voltage, power, etc.) | Hardware root of trust<br>Pre and Post silicon validation |
| Service provider | Data tampering and privacy<br>Malicious or faulty programs | Data at rest/in motion encryption<br>Program attestation |
| Users | Privilege escalation | Resource virtualization |

and Mukherjee, 2024). Overall, while SmartNICs provide powerful primitives for in-network cryptography, deployment must carefully consider performance, compatibility, and resource constraints across diverse protocols and workloads.

### *7.3. Comparison with programmable switches: anonymity and confidentiality*

Programmable switches are limited in their ability to support cryptographic operations due to constrained instruction sets and minimal memory per stage. Implementing even symmetric encryption like AES in P4 requires complex workarounds such as scrambled lookup tables (SLTs), which significantly reduce throughput, dropping from terabits per second to a few gigabits (Chen, 2020). These implementations often require packet recirculation, large table sizes, and rely on XOR and permutation operations rather than true cryptographic primitives, sacrificing security for feasibility (Mazloum et al., 2025).

In contrast, SmartNICs offer enhanced flexibility and compute capabilities for end-to-end cryptographic workloads, making them well-suited for in-network privacy and anonymity services. Equipped with ARM cores, DRAM, and PKAs (e.g., accelerators for AES, SHA), they can efficiently execute FPE, support secure FL aggregation, and perform HE-based operations. These capabilities enable stateful cryptographic operations without significantly degrading packet processing performance, functionality that lies beyond the reach of resource-constrained programmable switches. As a result, SmartNICs can enforce privacy-preserving learning (Zang et al., 2022; Pan et al., 2024), anonymize IP-layer metadata through in-network obfuscation (Mićović et al., 2023), and support lightweight cryptographic authentication for IoT devices (Bakar et al., 2024), offering a versatile platform for secure, high-speed edge computing.

### *7.4. Confidentiality and anonymity: summary and lessons learned*

SmartNICs can handle end-to-end encryption with hardware accelerators and secure protocols such as RDMA. However, some aspects should be considered when offloading cryptographic algorithms to SmartNICs. The key takeaways are:

- Offloading the complete TLS suite to a SmartNIC alleviates the CPU burden on the host. However, key exchange imposes significant overhead on SmartNICs, particularly in short-lived sessions. In such cases, collaboration with the host CPU can enhance performance.
- SmartNIC flexibility allows the implementation of new cryptographic algorithms (Nir and Langley, 2018) for TLS suite, and PQC algorithm (Aguilera et al., 2023). The performance of these algorithms can be enhanced by adding ASIC accelerators trading off with flexibility.
- RDMA-compatible SmartNICs can handle secure remote access by leveraging authentication and encryption, enabling one-sided login transactions for forensic analysis.
- SmartNICs can process data at the network edge, isolating it from the data center. This enables privacy enforcement, for example, by leveraging Federated Learning to train machine learning models without exposing raw data.

## 8. SmartNICs vulnerabilities

Adopting SmartNICs introduces advanced capabilities for network processing, potentially offloading many applications, including security. However, it can also pose security risks inherent to the hardware and software design. Understanding the threat model and identifying vulnerabilities is crucial for enhancing the robustness of networks that rely on SmartNICs.

### *8.1. Threat model*

Due to the ever-evolving threat landscape, the concept of a Trusted Execution Environment (TEE) has emerged, shifting the trust paradigm to individual devices rather than network boundaries. Technologies such as TrustZone (Alves, 2004; ARM, 2009) implements TEE on ARM cores. It enables software and hardware isolation between a secure world (trusted) and a normal world (untrusted). However, TrustZone is not well suited for virtualization (Hua et al., 2017; Cicero et al., 2018), facing many challenges due to hardware restrictions. Following the same principle, researchers are proposing threat models and frameworks to address the vulnerabilities of SmartNICs.

Lal et al. (2023) proposed a SmartNIC threat model in which the heterogeneous architecture of SmartNICs for confidential computing is analyzed in detail. SmartNIC ecosystem is composed of three domain levels: (1) Vendors, (2) Cloud Service Providers (CSP), and (3) Users. Vendors are companies that develop software and hardware, such as operating systems and chip designs. The CSPs own and deploy the devices in their infrastructure to offer offloading capabilities and access to the OS and telemetry functions. The users offload their applications to the SmartNICs and expect the confidentiality of their data in every state. The scenario is as follows: users aim to run network functions in a virtualized manner atop the SmartNIC hardware. The SmartNIC is attached to a host owned by the CSP. The host and the SmartNIC operate an OS provided and managed by the vendor and the CSP, respectively.

At the vendor level, they must provide strong security guarantees by design. Generally, vendors include third-party chips in their designs to reduce the development time. For example, the BlueField-2 incorporates TITAN IC for regex acceleration and Rambus PKA (Rambus, 2024). These third-party components are no longer trustable because the chips may be compromised at any stage of the production chain and exploited by adversaries. Table 10 lists the threats and corresponding remediation techniques in every domain level considered.

### *8.2. Third-party modules*

The evolution of NICs from simple designs to more complex architectures has led the industry to rely on third-party modules to meet time-to-market constraints. Growing reliance on these third-party modules from untrusted entities severely affects the security and trustworthiness of SmartNICs, especially for mission-critical infrastructure. Hardware trojans can potentially leak information such as the secret key of a communication channel. Also, it could drive the chip to not safe operating conditions; such an attack creates a violation of peak power and temperature contrast (also called thermal virus). Other

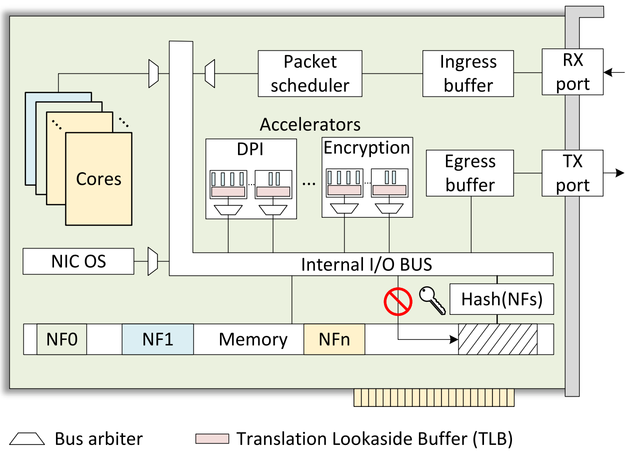


**Fig. 26.** SmartNIC architecture for network function isolation. Restricted memory allocations can only be accessed via a hardware root of trust.
*Source:* Reproduced from Zhou et al. (2024).

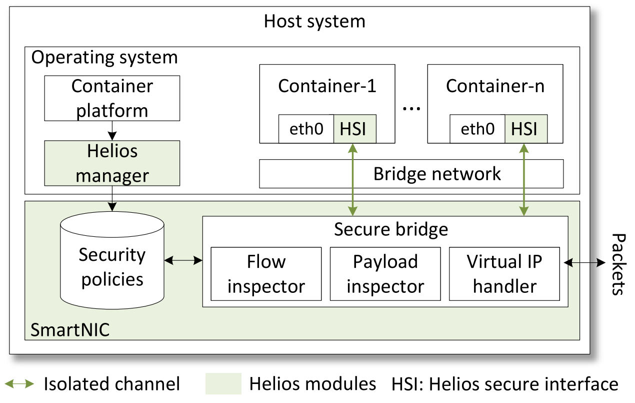


**Fig. 27.** Helios proposed architecture for container isolation (You et al., 2023).

vulnerabilities are the debug/test interfaces that adversaries can exploit to leak confidential information.

*Remediation techniques:* Detecting hardware Trojans in third-party modules is challenging due to the absence of a golden reference for comparison. To address this, some vendors are incorporating hardware Roots of Trust to ensure secure boot, cryptographic key storage, and authentication. Additionally, PCB-level protections are being implemented to defend against physical attacks (Mishra et al., 2017).

### *8.3. Multi-tenant and virtualization*

Efforts have been made to isolate and guarantee performance in the cloud (Grant et al., 2020; Huang et al., 2024; Khalilov et al., 2024), and other works test the performance degradation of isolating virtual network functions (Ni, 2022). Also, SmartNICs have been proposed as a source of trust in telemetry applications using sketches (Cheng et al., 2024). While performance isolation is essential, ensuring the security of network functions running on SmartNICs is even more critical for maintaining high-security standards.

*Remediation techniques:* Similar to the threat model proposed in Lal et al. (2023), S-NIC (Zhou et al., 2024) addresses the problem by pervasively virtualizing hardware accelerators, enforcing single-owner semantics for each line in on-NIC cache and RAM, and providing dedicated bus bandwidth for each network function. The goal is to provide a virtual SmartNIC, isolating host and NIC-level software. This design eliminates side channels involving shared hardware states, and each network function has the illusion of having a private SmartNIC. This offers hardware-guaranteed isolation both from other hosts and data center managers. Fig. 26 shows the S-NIC architecture for an SoC-based SmartNIC, in which the resources are carefully isolated with hardware-enforced security. The concrete attacks the author mitigates are packet corruption, stealing of DPI rules, and I/O bus denial of service. These mitigations are related to vendors' SmartNICs models, such as Marvell LiquidIO, Netronome Agilio, and NVIDIA Bluefield.

UniSec (Yan et al., 2019) proposes a unified framework API for developers that abstracts the functions implemented in SmartNIC by hardware–software co-design. The modularization process can take one of the three models: (1) Stateless header-based, (2) Stateful header-based, and (3) Payload-based. Moreover, UniSec designed a separate software security function running in the host and a hardware security function running in the SmartNIC, and they complement each other by orchestrating chained security functions, avoiding unnecessary hops between different devices. With the well-defined API, programmers only focus on the core logic, leaving resource management and matching code with accelerators to UniSec. However, this approach does not provide isolation between the disaggregated resources. SmartNICs can achieve high performance using zero-copy and dedicated CPU cores to process packets but trading off with security. Ni et al. (2019) discuss the security limitations of NFVs running on SmartNICs and propose a memory isolation solution based on P4 packet steering. In this work, the memory pool of SmartNICs is isolated using a routing function to steer packets right into their corresponding memory regions.

Helios (You et al., 2023) extended hardware to isolate containers and comprehensively run security engines in the SmartNIC. Fig. 27 shows the general overview of the architecture. Helios aims to enhance container security in virtualized environments by physically isolating communication channels through a secure bridge, replacing the legacy software-based network access control (e.g., security proxy). Helios is deployed in hardware, inspecting the east–west traffic in headers and payload data between the containers and the outside world. Helios leveraged micro-segmentation architecture to reduce the attack surface between containers by reducing network visibility with the flow inspector, blocking packets that violate the security policy maps established within the containers. Moreover, an enhanced version of Helios is presented in You et al. (2024). The Hyperion system follows the same architecture as Helios but extends the concept to nodes with multiple containers instead of a single node isolation. Hyperion adds a load balancer to enforce fairness among all the containers that share resources in the SmartNIC.

Open Radio Access Networks (O-RAN) are also adopting virtualization, and SmartNICs play a crucial role in offloading security infrastructures for the next generation of wireless communications. Achieving high performance in terms of latency, throughput, and power consumption while integrating heterogeneous applications remains a challenging task in open-source architectures. Additionally, enhancing security often appears to conflict with improving performance. Haas et al. (2022) proposed a hardware and operating system co-design to ensure secure integration of hardware and software components within O-RAN architectures. Their approach focuses on securing critical resources such as accelerators, memory, and communication interfaces at the chip level.

### *8.4. IDS/IPS bypassing*

Signature-based IDS/IPS systems deployed on SmartNICs are vulnerable to Algorithmic Complexity Attacks (ACAs), where adversaries craft input traffic that forces the system into worst-case processing paths. These attacks, unlike volumetric DDoS, do not require high traffic rates, small volumes of carefully designed packets can cause disproportionate resource consumption. In systems like Pigasus (Zhao et al., 2020), which rely on DPI and TCP reassembly, adversaries can inject highly OoO packets that trigger excessive memory lookups and

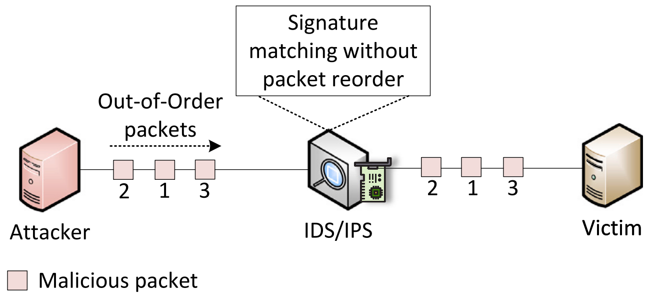


**Fig. 28.** Packet splitting attack bypassing a signature-based IDS (Zhao et al., 2022). Stateless DPI cannot identify malicious payloads spread across out-of-order packets.

list traversals (Zhao et al., 2022). This leads to elevated per-packet job sizes and causes innocent flows to be delayed or dropped when resources become constrained.

Such attacks are particularly effective against flow reassembly engines that buffer packets until an in-order byte stream is reconstructed. Excessive disorder (high OoOness) results in large buffer growth, degrading performance or bypassing inspection entirely when thresholds are exceeded. Fig. 28 illustrates how malicious payloads split across disordered segments may evade stateless DPI engines. Although SmartNICs offer higher on-board memory compared to traditional fixed-function devices, sustained ACA traffic can still exhaust these resources.

*Remediation techniques:* To mitigate this, adaptive flow-level scheduling mechanisms like SurgeProtector (Atre et al., 2022) prioritize lightweight, in-order traffic and penalize flows exhibiting abnormal behavior. By estimating the job size (e.g., reassembly effort or number of regex rules triggered) and applying weighted scheduling, the system deprioritizes adversarial flows while preserving throughput for benign ones. This approach has been shown to reduce the displacement factor (i.e., the ratio of innocent traffic dropped to attacker traffic injected) by over 90%, while maintaining fairness and low overhead.

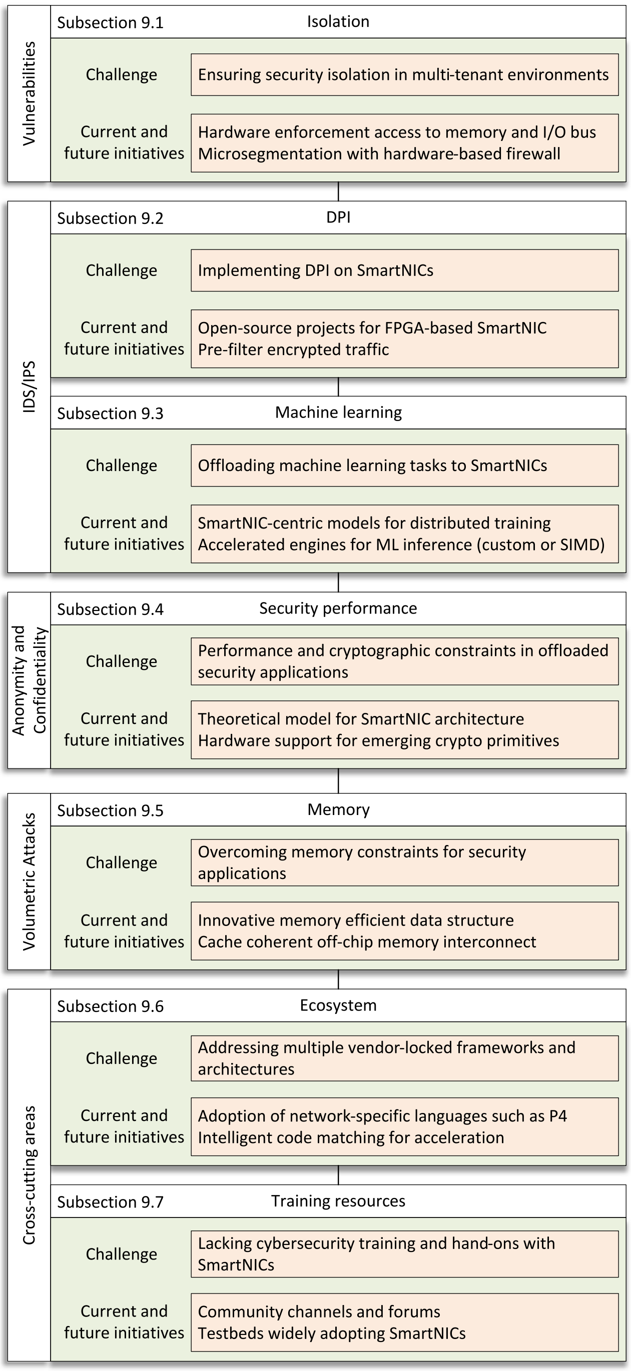
## 9. Challenges and future trends

This section presents the challenges associated with SmartNICs and their impact on security applications. Furthermore, several current initiatives and future directions are presented to address these challenges. Fig. 29 outlines the challenges and future trends.

### 9.1. Ensuring security isolation in multi-tenant environments

The heterogeneous architecture of SmartNICs, markedly different from traditional monolithic CPU designs, introduces new security challenges in virtualized, multi-tenant environments. These challenges arise primarily from inadequate isolation among disaggregated resources such as shared memory (e.g., SRAM, DRAM), I/O buses, and hardware accelerators. In public cloud deployments, where containers or network functions may be co-resident on the same SmartNIC, robust access isolation is essential to prevent privilege escalation, code injection, DPI rule theft, DoS attacks, and packet manipulation (Zhou et al., 2024; Lal et al., 2023).

*Current and future initiatives:* Several efforts in the literature have addressed the challenge of enforcing security isolation in SmartNIC-based virtualized environments by targeting different system layers. Architectural approaches like those proposed by Lal et al. (2023) define requirements for SmartNICs to support trusted execution, including secure host interfaces, cryptographic protections, and hardware-enforced access control. Others, such as Ni et al. (2019), propose software-defined memory isolation using packet steering to restrict memory access among co-resident functions. Helios (You et al., 2023) explores container-level isolation by using SmartNICs as a hardware root of



| | | | Challenge | Current and future initiatives |
|---|---|---|---|---|
| Vulnerabilities | Subsection 9.1 | Isolation | Ensuring security isolation in multi-tenant environments | Hardware enforcement access to memory and I/O bus<br>Microsegmentation with hardware-based firewall |
| IDS/IPS | Subsection 9.2 | DPI | Implementing DPI on SmartNICs | Open-source projects for FPGA-based SmartNIC<br>Pre-filter encrypted traffic |
| IDS/IPS | Subsection 9.3 | Machine learning | Offloading machine learning tasks to SmartNICs | SmartNIC-centric models for distributed training<br>Accelerated engines for ML inference (custom or SIMD) |
| Anonymity and Confidentiality | Subsection 9.4 | Security performance | Performance and cryptographic constraints in offloaded security applications | Theoretical model for SmartNIC architecture<br>Hardware support for emerging crypto primitives |
| Volumetric Attacks | Subsection 9.5 | Memory | Overcoming memory constraints for security applications | Innovative memory efficient data structure<br>Cache coherent off-chip memory interconnect |
| Cross-cutting areas | Subsection 9.6 | Ecosystem | Addressing multiple vendor-locked frameworks and architectures | Adoption of network-specific languages such as P4<br>Intelligent code matching for acceleration |
| Cross-cutting areas | Subsection 9.7 | Training resources | Lacking cybersecurity training and hand-ons with SmartNICs | Community channels and forums<br>Testbeds widely adopting SmartNICs |

**Fig. 29.** Overview of challenges and future trends in SmartNICs security.

trust, while Zhou et al. (2024) push isolation down to the microarchitectural level with techniques such as memory fingerprinting and bus arbitration. These solutions, while varied, reflect a shared goal of preventing cross-tenant interference and ensuring secure execution in multi-tenant environments.

More recently, Giantsidi et al. introduced TNIC (Giantsidi et al., 2025), a SmartNIC architecture that embeds a formally verified root of trust in the data plane to provide host-independent security guarantees. By supporting non-equivocation and transferable authentication, TNIC enables verifiable message delivery and strong isolation in Byzantine environments (i.e., actors may behave maliciously or unpredictably). It achieves these guarantees with minimal hardware overhead and delivers up to 6× higher throughput than traditional TEE-based systems.

These developments highlight a clear trend: SmartNICs are evolving beyond programmable packet processors to become secure, verifiable execution platforms. Future SmartNIC designs will increasingly integrate formal methods, hardware-enforced isolation, and lightweight trust anchors to meet the demands of secure, multi-tenant computing. However, realizing this vision will require a holistic co-design of hardware and software that balances isolation, compatibility, and performance.

### *9.2. Implementing DPI on SmartNICs*

General-purpose CPUs are not well-suited for DPI and struggle to manage large volumes of bulk traffic efficiently. SmartNICs offer a promising solution capable of performing DPI at line rate, thereby reducing CPU utilization and energy consumption. However, implementing DPI in the data plane poses challenges due to the demands of high-speed networks, which require low latency, high throughput, and adherence to memory constraints. An essential step before performing DPI is to reorder TCP packets for accurate matching. However, this process exposes the system to DoS attacks by injecting a high percentage of unsequenced packets, known as ACAs (Zhao et al., 2022; Afek et al., 2016). Furthermore, the effectiveness of DPI is diminished in two scenarios, under the prevalence of encrypted traffic (e.g., HTTP and VPN) and under a DDoS attack where the packets are malicious, thus requiring a mechanism to filter those packets and avoid waste of resources.

*Current and future initiatives:* Researchers are developing open-source frameworks, such as Pigasus and its improved version, Pigasus 2.0 (Zhao et al., 2022), that support inline DPI using Snort rules. These can be implemented using FPGA-based SmartNICs, which deliver line-rate performance. Additionally, Fidas (Chen et al., 2022) not only proposed string-matching but also identified a synergy to run a flow-rate classifier in parallel to detect and mitigate DDoS attacks. This approach optimizes DPI for benign flows rather than analyzing many malicious flows under a DDoS attack.

When dealing with encrypted traffic, inspecting payload becomes obsolete, so mainly two approaches are proposed to overcome this issue (Papadogiannaki et al., 2022). A straightforward approach to handling encrypted traffic is through filtering. Košař et al. (2023) introduce a hardware TLS pre-filter that eliminates encrypted traffic from the IDS pipeline, significantly enhancing performance without compromising rule types or IDS functionalities. This functionality can be easily added to existing frameworks. Nevertheless, the challenge remains in developing more advanced techniques for inspecting encrypted network traffic without compromising data privacy. Future directions can integrate SmartNICs with the efforts that have been done to preserve DPI rules, privacy, and data confidentiality, such as in Ning et al. (2020, 2019), Canard et al. (2017) and Kim et al. (2021). This would boost the performance and efficiency of SmartNIC-based IDS/IPS solutions while maintaining the controllability of open-source frameworks and privacy of the end users.

### *9.3. Offloading machine learning tasks to SmartNICs*

The integration of machine learning into intrusion detection and prevention systems (IDS/IPS) has significantly enhanced their ability to detect zero-day attacks and analyze encrypted traffic. To meet real-time requirements in high-speed networks, there is growing interest in offloading ML inference to SmartNICs. This allows for packet-level decision-making closer to the data source while reducing the computational burden on the host CPU.

However, SmartNICs impose architectural constraints that complicate ML model deployment. These include limited memory, the absence of floating-point units, and restricted support for complex arithmetic.

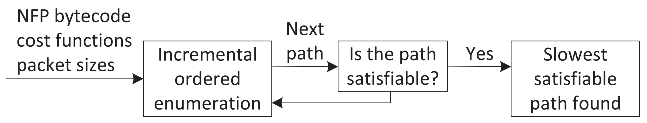


**Fig. 30.** Performance bottleneck estimation workflow using satisfiability-guided analysis (Krude et al., 2021).

As a result, models must be quantized to integer-only operations, with inference logic typically implemented using lightweight matrix multiplications, decision trees (e.g., a series of if-else statements), or lookup tables. Feature extraction must also rely on integer arithmetic, typically derived from flow-level metadata such as packet size, inter-arrival time, or protocol flags.

Frameworks like DOCA do not provide built-in ML inference APIs, making integration into SmartNIC dataplanes more challenging. While some alternatives enable SmartNICs to access GPUs via GPUDirect (Zhang et al., 2023) for training or inference, this approach introduces additional latency and PCIe bandwidth overhead, making it less suitable for latency-sensitive applications.

*Current and future initiatives:* Recent implementations have translated machine learning models, such as random forests and small neural networks, into conditional logic and integer-based vector operations, making them suitable for SmartNIC environments (Xavier et al., 2021). Unexplored areas, such as leveraging single instruction, multiple data (SIMD) capabilities within SoC-based SmartNICs, show promise for embedding classical ML models (e.g., neural networks, SVMs), potentially outperforming host CPU execution in both latency and efficiency. A growing body of research has also explored offloading ML inference to the data plane using P4-programmable switches (Datta et al., 2024; Lee and Singh, 2020), including the development of toolchains for ML offloading (Zhang et al., 2024), which opens opportunities for similar techniques to be adopted in SmartNIC environments.

For training, SmartNICs are gaining attention in distributed systems. A fully SmartNIC-based cluster has been proposed as a replacement for host servers (Park et al., 2024), creating new possibilities for enhancing inter-node communication protocols. For example, Hè (2024) demonstrate how SmartNICs can compress gradients in transit, optimizing communication during distributed training.

As threats become increasingly evasive and encrypted traffic continues to rise, ML-augmented SmartNICs are poised to become essential for scalable, low-latency IDS/IPS in modern data centers. Realizing this vision, however, will require continued advances in hardware-aware model optimization, development tooling, and secure, trustworthy deployment pipelines.

### *9.4. Performance and cryptographic constraints in offloaded security applications*

Offloading security functions such as encryption, hashing, and traffic inspection to SmartNICs enhances system performance by processing data closer to the network. However, two key challenges persist: limitations in executing cryptographic operations on constrained hardware, and the lack of systematic tools for performance prediction across heterogeneous SmartNIC platforms.

SmartNICs often restrict operations to 32-bit integers and lack support for floating point, exponentiation, or logarithmic functions, which limits accurate implementation of advanced cryptographic primitives. These constraints necessitate offloading complex operations to general-purpose CPUs, adding latency (Kfoury et al., 2024; AlSabeh et al., 2022b). Viegas et al. (2021) evaluated hash functions on the Netronome NFP4000 and observed that only a subset (`crc32-custom`, `random`, `csum16`) maintains line-rate throughput, while others reduce performance by up to 23%.

*Current and future initiatives:* On the performance modeling side, Krude et al. (2021) proposed a throughput estimation method based on satisfiability-guided path enumeration (see Fig. 30), achieving high accuracy with an error under 1.7%. Guo et al. (2023) introduced LogNIC, a packet-centric execution model that evaluates SmartNIC behavior across compute and memory domains, while Wu et al. (2024) developed Tomur, a predictive framework that models NF resource contention with a 3.7% average error. These works collectively underscore the need for hardware support for emerging crypto primitives and architecture-independent models that enable predictable and efficient deployment of offloaded security functions.

### 9.5. Overcoming memory constraints for security applications

SmartNICs face significant memory limitations compared to host servers, which can impact their ability to support memory-intensive security applications. While server memory can scale to hundreds of gigabytes, SmartNICs typically provide much less, ranging from a few megabytes of on-chip memory (e.g., BRAM or TCAM) to several gigabytes of off-chip DRAM. The trade-off between access speed and capacity is a central design constraint: on-chip memory offers fast, single-cycle access but is scarce, while off-chip DRAM adds latency and increases power and cost.

In security workloads, memory is crucial for tracking flow state, storing counters, and maintaining rule sets. DDoS defenses rely on fast access to prefix counters, while IDS/IPS systems store and evaluate partial signature rules. To adapt, developers design compact, efficient data structures optimized for SmartNIC memory hierarchies. For example, SmartWatch (Panda et al., 2021) introduces FlowCache, a hash table combined with ring buffers to support lossless state tracking for up to 25 million flows. This architecture enables shared buffering between SmartNIC and host for applications such as traffic analysis and heavy hitter detection. Similarly, Turkovic et al. (2020) propose modulo sketching and sequential zeroing to track flow statistics with minimal memory footprint, improving scalability using match-action lookups.

*Current and future initiatives:* To address the limitations of on-chip memory, recent research has explored offloading to external memory and redesigning memory hierarchies. For example, SmartDIMM (Patel et al., 2024) introduces a hybrid architecture that integrates FPGA-based packet processing closer to DRAM, reducing data movement and improving latency for memory-bound workloads. While promising, such off-chip and remote memory solutions pose challenges in high-speed environments, particularly under real-time constraints. Similarly, Ji et al. (2023) propose offloading memory-intensive kernel operations to SmartNICs. Their design leverages RDMA to access host memory directly, significantly reducing the load on host CPU cycles. As SmartNICs increasingly support complex security functions, addressing memory bottlenecks will demand not only architectural innovation but also the development of memory-efficient data structures and collaborative memory management strategies.

### 9.6. Addressing multiple vendor-locked frameworks and architectures

The SmartNICs ecosystem is mainly composed of two actors. The first one is the vendors that develop hardware and software frameworks for their products. The second is the CSPs, which adopt these technologies and offer services to clients. SmartNICs often require different programming approaches, typically using proprietary compilers to match the underlying architecture of the device. The diversity of development tools reduces collaboration and code reuse while increasing application development time and cost. Moreover, an essential feature in the cybersecurity domain is the controllability and compatibility of deployed technologies with future generations and previous devices. Commercial software frameworks such as DOCA (NVIDIA, 2024) raise issues of software controllability and vendor lock-in, making it more challenging to migrate from one vendor to another or to adopt new technologies (Chen et al., 2022).

*Current and future initiatives:* To innovate in the network field without being tightly restricted to the vendor, researchers typically employ network programming languages such as P4 (P4 Language Consortium, 2024), DPDK (The Linux Foundation, 2024), or kernel-based packet processing technologies such as eBPF/XDP (Linux Foundation, 2024). These technologies utilize proprietary SDK APIs to maximize the use of accelerators tailored to specific architectures and vendors. Several initiatives are underway to develop standardized APIs for SmartNIC programming, but these APIs still lack vendor-specific functions necessary for fully leveraging hardware accelerators. Researchers are turning to program analysis techniques to extract offloading insights. For example, Qiu et al. (2021) proposed a porting technique based on machine learning and code analysis. The core idea is to identify which software components can leverage accelerators to enhance performance significantly and to estimate the required core count for running specific network functions.

Adopting P4 to enhance code reusability through its high-level abstraction syntax allows developers to offload applications onto different target devices (e.g., SmartNICs or switches). Choueiri et al. (2024) integrated P4-DPDK and SmartNICs to detect heavy hitters using count-min sketches. Their work demonstrated the feasibility of leveraging the P4 programming language to program CPU cores for implementing security functions. However, this abstraction can obscure opportunities for low-level performance optimization, as the compilers for these languages are not yet mature enough to guarantee the correctness of the output programs. To address this challenge, Xing et al. (2023) introduced an optimization framework called Pipeleon. This framework utilizes profiling to apply performance optimization techniques on P4 programmable SmartNICs, considering the underlying architecture of these devices, such as multicores and ASICs. The results demonstrate improved latency and throughput. A promising research direction is to enhance the maturity of existing frameworks that fully leverage the underlying architecture of SmartNICs through a hardware–software co-design approach.

### 9.7. Addressing gap in cybersecurity training

Nowadays, only 3% of organizations are incorporating effective countermeasures against cyber threats, according to Cisco reports (Cisco, 2023), mainly due to a lack of infrastructure and qualified cybersecurity experts. SmartNIC is a promising technology that can bridge this gap, but it faces many challenges, particularly finding courses and training materials related to cybersecurity. Adopting new technology like SmartNICs involves an inherent learning curve for beginners and advanced users. The available documentation and guided tutorials for running applications on SmartNICs are often tailored to vendor-specific SDKs or proprietary frameworks. These vendors offer training material tailored to their products (NVIDIA, 2021). However, open-source SDKs are appearing as an alternative for SoC-based SmartNICs such as DPDK (The Linux Foundation, 2024), and open-source vendor-agnostics platforms for FPGA-based SmartNICs such as NetFPGA (NetFPGA, 2021), Pigasus open-source IDS platforms (Zhao et al., 2020), and more. Developers need hands-on experience to gain a deeper understanding of SmartNICs and reliable testbeds to test their innovative solutions.

*Current and future initiatives:* To address this shortcoming, testbeds worldwide, such as FABRIC (Baldin et al., 2019), CloudLab (Ricci et al., 2014), and Chameleon (Keahey et al., 2019), have incorporated SmartNICs into their networks, thus providing the needed infrastructure for cyber training. However, while the community shares updates and advancements regarding SmartNICs on YouTube channels and forums, these resources are often not comprehensive enough and do not thoroughly cover cybersecurity fundamentals. There is a need for detailed manuals and tutorials for implementing security applications, as well as

**Table 11**
Abbreviations used in this survey.

| Abbreviations (A–M) | | Abbreviations (N–Z) | |
|---|---|---|---|
| AC | Aho–Corasick | NAT | Network Address Translation |
| ACA | Algorithm Complexity Attacks | NFV | Network Function Virtualization |
| ACL | Access Control List | NFA | Non-Deterministic Finite Automata |
| AES | Advanced Encryption Standard | NFP | Netronome Flow Processor |
| ALU | Arithmetic Logic Unit | NGFW | Next-Generation Firewall |
| ANOVA | Analysis of Variance | NIC | Network Interface Card |
| API | Application Programming Interface | NLP | Natural Language Processing |
| ASIC | Application Specific Integrated Circuit | NoC | Network on Chip |
| C2 | Command and Control | NVMe | Non-Volatile Memory Express |
| CPU | Central Processing Unit | NVMe-oF | NVMe over Fabric |
| CSP | Cloud Service Provider | O-RAN | Open Radio Access Networks |
| D2D | Device to Device | OoO | Out of Order |
| DDoS | Distributed Denial of Service | OS | Operating System |
| DFA | Deterministic Finite Automata | P4 | Programming Protocol-independent Packet Processor |
| DH | Diffie–Hellman | PCIe | Peripheral Component Interconnect Express |
| DNS | Domain Name System | PDP | Programmable Data Plane |
| DoS | Denial of Service | PKA | Public Key Accelerator |
| DPDK | Data Plane Development Kit | PQC | Post Quantum Computing |
| DPI | Deep Packet Inspection | PUF | Physical Unclonable Function |
| DPU | Data Processing Unit | QoS | Quality of Service |
| DRAM | Dynamic Random-Access Memory | QP | Queue Pair |
| DSA | Domain-Specific Accelerators | RDMA | Remote Direct Memory Access |
| ECC | Elliptic Curve Cryptography | Regex | Regular Expression |
| eBPF | Extended Berkeley Packet Filter | RFPF | Restricted Feature-set Packet Filter |
| FL | Federated Learning | RoCE | RDMA over Converged Ethernet |
| FPC | Flow Processor Core | RSA | Rivest–Shamir–Adleman |
| FPE | Format Preserving Encryption | RSS | Receive Side Scaling |
| FPGA | Field Programmable Gate Array | SDK | Software Development Kit |
| HE | Homomorphic Encryption | SDN | Software-Defined Network |
| HH | Heavy Hitter | SLT | Scrambled Lookup Table |
| HTTP | Hypertext Transfer Protocol | SoC | System on a Chip |
| HTTPS | Hypertext Transfer Protocol Secure | SQL | Structured Query Language |
| IBA | InfiniBand | SQLi | SQL Injection |
| IDS/IPS | Intrusion Detection and Prevention System | SRAM | Static Random-Access Memory |
| IoT | Internet of Things | SSD | Solid State Disk |
| IP | Internet Protocol | TCP | Transmission Control Protocol |
| IPSec | Internet Protocol Security | TEE | Trusted Execution Environment |
| IPU | Infrastructure Processing Unit | TLS | Transport Layer Security |
| kTLS | Kernel TLS | TRNG | True Random Number Generator |
| MACsec | Media Access Control Security | URL | Uniform Resource Locator |
| ML | Machine Learning | VM | Virtual Machine |
| MLP | Multilayer Perceptron | VPN | Virtual Private Network |
| MPLS | Multiprotocol Label Switching | XDP | eXpress Data Path |

hands-on labs that comprehensively cover cybersecurity fundamentals and advanced topics. These resources should include instructions for configuring basic security applications such as firewalls, ACLs, and URL filters and more advanced countermeasures against DDoS attacks, DoS slow attacks, and offloading IDS/IPS services on the SmartNIC. Additionally, the offloading of workloads related to encryption and decryption, mainly for security suites such as TLS and IPSec, to SmartNICs should be thoroughly covered.

## 10. Conclusion

This paper presents a comprehensive survey of SmartNICs, with particular emphasis on security applications. First, the survey provides background information on various SmartNIC architectures and advancements in hardware designed for offloading network security infrastructure. Then, state-of-the-art related works are presented. Motivated by the lack of security-centric work on SmartNICs, the survey offers a taxonomy of SmartNIC security applications, categorizing them into three types: Intrusion Detection and Prevention Systems, Volumetric Attacks, and Anonymity and Confidentiality. The survey also evaluates the vulnerabilities of SmartNIC architectures, from threat models to various attack scenarios, and presents remediation techniques discussed in the literature. The paper concludes with a discussion of current challenges and future trends in SmartNIC security, highlighting research directions worthy of further investigation.

## CRediT authorship contribution statement

**Sergio Elizalde:** Writing – review & editing, Writing – original draft, Investigation, Conceptualization. **Ali AlSabeh:** Writing – review & editing, Investigation, Conceptualization. **Ali Mazloum:** Writing – review & editing. **Samia Choueiri:** Writing – review & editing. **Elie Kfoury:** Writing – review & editing, Conceptualization. **Jose Gomez:** Writing – review & editing, Writing – original draft. **Jorge Crichigno:** Writing – review & editing, Project administration, Funding acquisition, Conceptualization.

## Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Sergio Elizalde reports financial support was provided by National Science Foundation. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgments

This work is supported by the National Science Foundation (NSF), Office of Advanced Cyberinfrastructure (OAC), under grant numbers 2118311, 2403360, and 2346726.

## Data availability

No data was used for the research described in the article.

## References

Afek, Y., Bremler-Barr, A., Harchol, Y., Hay, D., Koral, Y., 2016. Making DPI engines resilient to algorithmic complexity attacks. IEEE/ACM Trans. Netw. 24 (6), 3262–3275.

Aguilera, A.C., Clemente, X.A.i., Lawo, D., Monroy, I.T., Olmos, J.V., 2023. First end-to-end PQC protected DPU-to-DPU communications. Electron. Lett. 59 (17), e12901.

Ahmad, R., Alsmadi, I., Alhamdani, W., Tawalbeh, L., 2023. Zero-day attack detection: a systematic literature review. Artif. Intell. Rev. 56 (10), 10733–10811.

Aho, A.V., Corasick, M.J., 1975. Efficient string matching: an aid to bibliographic search. Commun. ACM 18 (6), 333–340.

Al-Ofeishat, H.A., Alshorman, R., 2023. Build a secure network using segmentation and micro-segmentation techniques. Int. J. Comput. Digit. Syst. 16 (1), 1499–1508.

AlSabeh, A., Kfoury, E., Crichigno, J., Bou-Harb, E., 2022a. P4ddpi: Securing p4-programmable data plane networks via dns deep packet inspection. In: NDSS Symposium 2022.

AlSabeh, A., Khoury, J., Kfoury, E., Crichigno, J., Bou-Harb, E., 2022b. A survey on security applications of P4 programmable switches and a STRIDE-based vulnerability assessment. Comput. Netw. 207.

Alves, T., 2004. Trustzone: Integrated hardware and software security. Inf. Q. 3, 18–24.

ARM, A., 2009. Security Technology Building a Secure System Using Trustzone Technology. ARM Limited.

Arshad, S., Zanib, R., Akram, A., Saeed, T., 2023. A short review on faster and more reliable tcp reassembly for high-speed networks in deep packet inspection. In: 2023 1st International Conference on Advanced Innovations in Smart Cities. ICAISC, IEEE, pp. 1–6.

Aslan, Ö., Aktuğ, S.S., Ozkan-Okay, M., Yilmaz, A.A., Akin, E., 2023. A comprehensive review of cyber security vulnerabilities, threats, attacks, and solutions. Electronics 12 (6), 1333.

Atre, N., Sadok, H., Chiang, E., Wang, W., Sherry, J., 2022. SurgeProtector: Mitigating temporal algorithmic complexity attacks using adversarial scheduling. In: Proceedings of the ACM SIGCOMM 2022 Conference. pp. 723–738.

Bakar, R.A., Paolucci, F., Cugini, F., Castoldi, P., Olmos, J.J.V., 2024. DPUAUT: Secure authentication protocol with SmartNiC integration for trustworthy communications in intelligent swarm systems. IEEE Access.

Baldin, I., Nikolich, A., Griffioen, J., Monga, I.I.S., Wang, K.-C., Lehman, T., Ruth, P., 2019. Fabric: A national-scale programmable experimental network infrastructure. IEEE Internet Comput. 23 (6), 38–47.

Beckwith, L., Abdulgadir, A., Azarderakhsh, R., 2023. A flexible shared hardware accelerator for NIST-recommended algorithms CRYSTALS-Kyber and CRYSTALS-Dilithium with SCA protection. In: Cryptographers' Track At the RSA Conference. Springer, pp. 469–490.

Ben-Basat, R., Chen, X., Einziger, G., Rottenstreich, O., 2018. Efficient measurement on programmable switches using probabilistic recirculation. In: 2018 IEEE 26th International Conference on Network Protocols. ICNP, IEEE, pp. 313–323.

Benson, T., Akella, A., Maltz, D.A., 2010. Network traffic characteristics of data centers in the wild. In: Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement. pp. 267–280.

Bernstein, D.J., et al., 2008. ChaCha, a variant of salsa20. In: Workshop Record of SASC. Vol. 8, Citeseer, pp. 3–5.

Brown, D.J., Suckow, B., Wang, T., 2002. A Survey of Intrusion Detection Systems. Department of Computer Science, University of California, San Diego.

Bruschi, V., Basat, R.B., Liu, Z., Antichi, G., Bianchi, G., Mitzenmacher, M., 2020. DISCOvering the heavy hitters with disaggregated sketches. In: Proceedings of the 16th International Conference on Emerging Networking EXperiments and Technologies. pp. 536–537.

Canard, S., Diop, A., Kheir, N., Paindavoine, M., Sabt, M., 2017. BlindIDS: Market-compliant and privacy-friendly intrusion detection system over encrypted traffic. In: Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security. pp. 561–574.

Chang, H., Mukherjee, S., 2024. Zeta: Transparent zero-trust security add-on for RDMA. In: IEEE INFOCOM 2024-IEEE Conference on Computer Communications. IEEE, pp. 1041–1050.

Chen, X., 2020. Implementing AES encryption on programmable switches via scrambled lookup tables. In: Proceedings of the Workshop on Secure Programmable Network Infrastructure. pp. 8–14.

Chen, X., Shi, Z., Qian, L., 2024. SEC-RDMA: a scheme to enhance security for RDMA one-sided operations. In: Third International Conference on High Performance Computing and Communication Engineering (HPCCE 2023). 13073, SPIE, pp. 112–120.

Chen, X., Wu, C., Liu, X., Huang, Q., Zhang, D., Zhou, H., Yang, Q., Khan, M.K., 2023. Empowering network security with programmable switches: A comprehensive survey. IEEE Commun. Surv. Tutor. 25 (3), 1653–1704.

Chen, J., Zhang, X., Wang, T., Zhang, Y., Chen, T., Chen, J., Xie, M., Liu, Q., 2022. Fidas: fortifying the cloud via comprehensive FPGA-based offloading for intrusion detection: industrial product. In: Proceedings of the 49th Annual International Symposium on Computer Architecture. pp. 1029–1041.

Cheng, Z., Apostolaki, M., Liu, Z., Sekar, V., 2024. TRUSTSKETCH: Trustworthy sketch-based telemetry on cloud hosts. In: The Network and Distributed System Security Symposium. NDSS.

Cheng, H., Großschädl, J., Marshall, B., Page, D., Pham, T., 2023. RISC-V instruction set extensions for lightweight symmetric cryptography. IACR Trans. Cryptogr. Hardw. Embed. Syst. 193–237.

Choueiri, S., Mazloum, A., Kfoury, E.F., Crichigno, J., Scalable heavy hitter detection: A DPDK-based software approach with P4 integration.

Cicero, G., Biondi, A., Buttazzo, G., Patel, A., 2018. Reconciling security with virtualization: A dual-hypervisor design for ARM TrustZone. In: 2018 IEEE International Conference on Industrial Technology. ICIT, IEEE, pp. 1628–1633.

Cisco, 2021. Cisco visual networking index: Forecast and trends, 2017–2022. [Online]. Available: https://tinyurl.com/askfxvz9.

Cisco, 2023. Cisco Cybersecurity Readiness Index. Tech. Rep., Cisco Systems, Inc., San Jose, CA, URL https://newsroom.cisco.com/c/dam/r/newsroom/en/us/interactive/cybersecurity-readiness-index/documents/Cisco_Cybersecurity_Readiness_Index_FINAL.pdf. (Accessed 28 June 2024).

Cisco Systems, I., Cisco Firepower NGFW Virtual, [Online]. Available: https://www.cisco.com/c/en/us/products/collateral/security/firepower-ngfw-virtual/threat-defense-virtual-ngfwv-ds.html.

Cormode, G., Muthukrishnan, S., 2005. An improved data stream summary: the count-min sketch and its applications. J. Algorithms 55 (1), 58–75.

Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., Singer, Y., Warmuth, M.K., 2006. Online passive-aggressive algorithms. J. Mach. Learn. Res. 7 (3).

Dally, W.J., Turakhia, Y., Han, S., 2020. Domain-specific hardware accelerators. Commun. ACM 63 (7), 48–57.

Dancheva, T., Alonso, U., Barton, M., 2024. Cloud benchmarking and performance analysis of an HPC application in Amazon EC2. Clust. Comput. 27 (2), 2273–2290.

Dastidar, J., Riddoch, D., Moore, J., Pope, S., Wesselkamper, J., 2023. The AMD 400-G adaptive SmartNIC system on chip: a technology preview. IEEE Micro 43 (3), 40–49.

Datta, S., Kotha, A., et al., 2024. XNetIoT: An extreme quantized neural network architecture for IoT environment using P4. IEEE Trans. Netw. Serv. Manag..

Dierks, T., Rescorla, E., 2008. The Transport Layer Security (TLS) Protocol Version 1.2. Request for Comments: 5246, RFC 5246, URL https://www.rfc-editor.org/info/rfc5246.

Dimolianis, M., Kalogeras, D.K., Kostopoulos, N., Maglaris, V., 2022. Ddos attack detection via privacy-aware federated learning and collaborative mitigation in multi-domain cyber infrastructures. In: 2022 IEEE 11th International Conference on Cloud Networking (CloudNet). IEEE, pp. 118–125.

Dimolianis, M., Pavlidis, A., Maglaris, V., 2020. A multi-feature ddos detection schema on P4 network hardware. In: 2020 23rd Conference on Innovation in Clouds, Internet and Networks and Workshops. ICIN, IEEE, pp. 1–6.

Dimolianis, M., Pavlidis, A., Maglaris, V., 2021. Signature-based traffic classification and mitigation for DDoS attacks using programmable network data planes. IEEE Access 9, 113061–113076.

Dworkin, M., 2019. Recommendation for block cipher modes of operation: Methods for format-preserving encryption. NIST Special Publication 800-38G Rev. 1 (Initial Public Draft), https://csrc.nist.gov/pubs/sp/800/38/g/r1/ipd.

Federal Bureau of Investigation, 2023. Internet crime report.

Fei, X., Liu, F., Zhang, Q., Jin, H., Hu, H., 2020. Paving the way for NFV acceleration: A taxonomy, survey and future directions. ACM Comput. Surv. 53 (4), 1–42.

Firestone, D., Putnam, A., Mundkur, S., Chiou, D., Dabagh, A., Andrewartha, M., Angepat, H., Bhanu, V., Caulfield, A., Chung, E., et al., 2018. Azure accelerated networking: Smartnics in the public cloud. In: 15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18). pp. 51–66.

Fortinet, I., 2024. FortiGate virtual appliances datasheet. [Online]. Available: https://www.fortinet.com/content/dam/fortinet/assets/data-sheets/fortigate-vm.pdf.

Freitas, E., de Oliveira Filho, A.T., do Carmo, P.R., Sadok, D., Kelner, J., 2022. A survey on accelerating technologies for fast network packet processing in Linux environments. Comput. Commun. 196, 148–166.

Gartner, 2009. Defining the next-generation firewall. [Online]. Available: https://www.gartner.com/en/documents/1204914.

Giantsidi, D., Pritzi, J., Gust, F., Katsarakis, A., Koshiba, A., Bhatotia, P., 2025. TNIC: A Trusted NIC Architecture: A hardware-network substrate for building high-performance trustworthy distributed systems. In: Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2. pp. 1282–1301.

Gooding, D., Critical MOVEit vulnerability puts huge swaths of the Internet at severe risk, [Online]. Available: https://tinyurl.com/mrbft7s6.

Gowtham, M., Pramod, H., 2021. Semantic query-featured ensemble learning model for SQL-injection attack detection in IoT-ecosystems. IEEE Trans. Reliab. 71 (2), 1057–1074.

Grant, S., Yelam, A., Bland, M., Snoeren, A.C., 2020. Smartnic performance isolation with fairnic: Programmable networking for the cloud. In: Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication. pp. 681–693.

Guo, Z., Lin, J., Bai, Y., Kim, D., Swift, M., Akella, A., Liu, M., 2023. LogNIC: A high-level performance model for SmartNICs. In: Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture.

Gupta, S., Gosain, D., Kwon, M., Acharya, H., 2023a. DeeP4R: Deep packet inspection in P4 using packet recirculation. In: IEEE INFOCOM 2023-IEEE Conference on Computer Communications. IEEE, pp. 1–10.

Gupta, S., Gosain, D., Kwon, M., Acharya, H.B., 2023b. Predictable internet clients and in-switch deep packet inspection. In: 2023 32nd International Conference on Computer Communications and Networks. ICCCN, IEEE, pp. 1–10.

Guz, Z., Li, H., Shayesteh, A., Balakrishnan, V., 2017. NVMe-over-fabrics performance characterization and the path to low-overhead flash disaggregation. In: Proceedings of the 10th ACM International Systems and Storage Conference. pp. 1–9.

Haas, S., Hasler, M., Pauls, F., Köpsell, S., Asmussen, N., Roitzsch, M., Fettweis, G., 2022. Trustworthy computing for O-RAN: Security in a latency-sensitive environment. In: 2022 IEEE Globecom Workshops (GC Wkshps). IEEE, pp. 826–831.

Hao, J., Jin, M., Li, Y., Piao, T., Hu, H., Xi, X., Chen, J., 2023. Power data traceability mechanism based on data processing unit. In: 2023 IEEE 11th Joint International Information Technology and Artificial Intelligence Conference. ITAIC, 11, IEEE, pp. 1928–1933.

Haseeb-Ur-Rehman, R.M.A., Aman, A.H.M., Hasan, M.K., Ariffin, K.A.Z., Namoun, A., Tufail, A., Kim, K.-H., 2023. High-speed network DDoS attack detection: A survey. Sensors 23 (15), 6850.

Hè, H., 2024. FPGA-based SmartNIC for Distributed Machine Learning (Master's thesis). ETH Zurich.

Hinic, S., Bakar, R.A., Marotta, A., Paolucci, F., Cugini, F., 2024. Wire-speed DDoS attack mitigation using hardware acceleration of programmable DPUs. In: 2024 IEEE Global Communications Conference. GLOBECOM, IEEE.

Hsieh, K., Wong, M., Segarra, S., Mani, S.K., Eberl, T., Panasyuk, A., Netravali, R., Chandra, R., Kandula, S., 2024. NetVigil: Robust and low-cost anomaly detection for East-West data center security. In: 21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24). pp. 1771–1789.

Hu, Z., Hasegawa, H., Yamaguchi, Y., Shimada, H., 2023. Realtime malicious traffic detection targeted for TCP out-of-order packets based on FPGA. IEEE Access.

Hua, Z., Gu, J., Xia, Y., Chen, H., Zang, B., Guan, H., 2017. vTZ: Virtualizing ARM TrustZone. In: 26th USENIX Security Symposium (USENIX Security 17). pp. 541–556.

Huang, Z., Tan, Y., Zhu, Y., Tan, H., Li, K., 2024. MTDA: Efficient and fair DPU offloading method for multiple tenants. IEEE Trans. Serv. Comput..

Hussain, L., Rawat, M., Yadav, N.K., Darak, S., Tammana, P., Shah, R., 2023. Microservice-based in-network security framework for FPGA NICs. In: 2023 IEEE/ACM 23rd International Symposium on Cluster, Cloud and Internet Computing Workshops (CCGridW). IEEE, pp. 328–330.

Hypolite, J., Sonchack, J., Hershkop, S., Dautenhahn, N., DeHon, A., Smith, J.M., 2020. 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. pp. 336–350.

Ingham, K., Forrest, S., et al., 2002. A History and Survey of Network Firewalls. Tech. Rep., University of New Mexico.

Ji, H., Mansi, M., Sun, Y., Yuan, Y., Huang, J., Kuper, R., Swift, M.M., Kim, N.S., 2023. $\{STYX\}$: Exploiting $\{SmartNIC\}$ capability to reduce datacenter memory tax. In: 2023 USENIX Annual Technical Conference (USENIX ATC 23). pp. 619–633.

Jia, C., Li, C., Li, Y., Hu, X., Li, J., 2022. FACL: A flexible and high-performance ACL engine on FPGA-based SmartNIC. In: 2022 IFIP Networking Conference (IFIP Networking). IEEE, pp. 1–9.

Juniper Networks, I., vSRX virtual firewall datasheet, [Online]. Available: https://www.juniper.net/content/dam/www/assets/datasheets/us/en/security/vsrx-virtual-firewall-datasheet.pdf.

Jyothsna, V., Prasad, R., Prasad, K.M., 2011. A review of anomaly based intrusion detection systems. Int. J. Comput. Appl. 28 (7), 26–35.

Kanev, S., Darago, J.P., Hazelwood, K., Ranganathan, P., Moseley, T., Wei, G.-Y., Brooks, D., 2015. Profiling a warehouse-scale computer. In: Proceedings of the 42nd Annual International Symposium on Computer Architecture. pp. 158–169.

Kapoor, E., Jampani, G., Choi, S., 2023. BlockNIC: SmartNIC assisted blockchain. In: 2023 Silicon Valley Cybersecurity Conference. SVCC, IEEE, pp. 1–8.

Kapourchali, R.F., Mohammadi, R., Nassiri, M., 2024. P4httpGuard: detection and prevention of slow-rate DDoS attacks using machine learning techniques in P4 switch. Clust. Comput. 1–18.

Keahey, K., Riteau, P., Stanzione, D., Cockerill, T., Mambretti, J., Rad, P., Ruth, P., 2019. Chameleon: a scalable production testbed for computer science research. In: Contemporary High Performance Computing. CRC Press, pp. 123–148.

Kfoury, E.F., Choueiri, S., Mazloum, A., AlSabeh, A., Gomez, J., Crichigno, J., 2024. A comprehensive survey on smartnics: Architectures, development models, applications, and research directions. IEEE Access.

Khalilov, M., Chrapek, M., Shen, S., Vezzu, A., Benz, T., Di Girolamo, S., Schneider, T., De Sensi, D., Benini, L., Hoefler, T., 2024. OSMOSIS: Enabling multi-tenancy in datacenter SmartNICs. In: 2024 USENIX Annual Technical Conference (USENIX ATC 24). pp. 247–263.

Khan, M.M.I., Nencioni, G., 2023. Resource allocation in networking and computing systems: A security and dependability perspective. IEEE Access.

Kim, J., Camtepe, S., Baek, J., Susilo, W., Pieprzyk, J., Nepal, S., 2021. P2DPI: practical and privacy-preserving deep packet inspection. In: Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security. pp. 135–146.

Kim, D., Lee, S., Park, K., 2020. A case for smartnic-accelerated private communication. In: Proceedings of the 4th Asia-Pacific Workshop on Networking. pp. 30–35.

Košař, V., Šišmiš, L., Matoušek, J., Kořenek, J., 2023. Accelerating IDS using TLS pre-filter in FPGA. In: 2023 IEEE Symposium on Computers and Communications. ISCC, IEEE, pp. 436–442.

Kottur, S.Z., Kadiyala, K., Tammana, P., Shah, R., 2022. Implementing chacha based crypto primitives on programmable smartnics. In: Proceedings of the ACM SIGCOMM Workshop on Formal Foundations and Security of Programmable Network Infrastructures. pp. 15–23.

Krude, J., Rüth, J., Schemmel, D., Rath, F., Folbort, I.-H., Wehrle, K., 2021. Determination of throughput guarantees for processor-based smartnics. In: Proceedings of the 17th International Conference on Emerging Networking EXperiments and Technologies.

Kumar, V., Sangwan, O.P., 2012. Signature based intrusion detection system using SNORT. Int. J. Comput. Appl. Inf. Technol. 1 (3), 35–41.

Lai, Y.-K., Huang, P.-Y., Lee, H.-P., Tsai, C.-L., Chang, C.-S., Nguyen, M.H., Lin, Y.-J., Liu, T.-L., Chen, J.H., 2020. Real-time ddos attack detection using sketch-based entropy estimation on the netfpga sume platform. In: 2020 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC). IEEE, pp. 1566–1570.

Lal, R., Anderson, J.B., Jackson, A., 2023. Data processing unit's entry into confidential computing. In: Proceedings of the 12th International Workshop on Hardware and Architectural Support for Security and Privacy. pp. 56–63.

Lapolli, Â.C., Marques, J.A., Gaspary, L.P., 2019. Offloading real-time ddos attack detection to programmable data planes. In: 2019 IFIP/IEEE Symposium on Integrated Network and Service Management. IM, IEEE, pp. 19–27.

Lawo, D.C., Frantz, R., Aguilera, A.C., i Clemente, X.A., Podleś, M., Imaña, J.L., Monroy, I.T., Olmos, J.V., 2024. Falcon/kyber and dilithium/kyber network stack on nvidia's data processing unit platform. IEEE Access.

Lee, J.-H., Singh, K., 2020. Switchtree: in-network computing and traffic analyses with random forests. Neural Comput. Appl. 1–12.

Lee, A.Y.-P., Wang, M.I.-C., Hung, C.-H., Wen, C.H.-P., 2024. PS-IPS: Deploying intrusion prevention system with machine learning on programmable switch. Future Gener. Comput. Syst. 152, 333–342.

Liang, J., Kim, Y., 2022. Evolution of firewalls: Toward securer network using next generation firewall. In: 2022 IEEE 12th Annual Computing and Communication Workshop and Conference. CCWC, IEEE, pp. 0752–0759.

Linguaglossa, L., Lange, S., Pontarelli, S., Rétvári, G., Rossi, D., Zinner, T., Bifulco, R., Jarschel, M., Bianchi, G., 2019. Survey of performance acceleration techniques for network function virtualization. Proc. IEEE 107 (4), 746–764.

Linux Foundation, eBPF and XDP: Extending and evolving the linux kernel, BPF.io Official Documentation. Available: https://ebpf.io.

Liu, M., Cui, T., Schuh, H., Krishnamurthy, A., Peter, S., Gupta, K., 2019. Offloading distributed applications onto smartnics using ipipe. In: Proceedings of the ACM Special Interest Group on Data Communication. pp. 318–333.

Luo, J.-L., Yu, S.-Z., Peng, S.-J., 2020. SDN/NFV-based security service function tree for cloud. IEEE Access 8, 38538–38545.

Mazloum, A., AlSabeh, A., Kfoury, E.F., Crichigno, J., Domain Name Security Inspection at Line Rate: TLS SNI Extraction in the Data Plane Using P4 and DPDK.

Miano, S., Doriguzzi-Corin, R., Risso, F., Siracusa, D., Sommese, R., 2019. Introducing smartnics in server-based data plane processing: The ddos mitigation use case. IEEE Access 7, 107161–107170.

Miano, S., Lettieri, G., Antichi, G., Procissi, G., 2024. Accelerating network analytics with an on-NIC streaming engine. Comput. Netw. 110231.

Mićović, M., Radenković, U., Vuletić, P., 2023. Network layer privacy protection using format-preserving encryption. Electronics 12 (23), 4800.

Mishra, P., Tehranipoor, M., Bhunia, S., 2017. Security and trust vulnerabilities in third-party IPs. Hardw. IP Secur. Trust. 3–14.

Mukherjee, B., Heberlein, L.T., Levitt, K.N., 1994. Network intrusion detection. IEEE Netw. 8 (3), 26–41.

Mukhopadhyay, I., Chakraborty, M., Chakrabarti, S., et al., 2011. A comparative study of related technologies of intrusion detection & prevention systems. J. Inf. Secur. 2 (01), 28.

Nakamoto, S., et al., 2008. Bitcoin. A Peer- To- Peer Electron. Cash Syst. 21260.

NetFPGA, NetFPGA, [Online]. Available: https://netfpga.org/.

Neupane, K., Haddad, R., Chen, L., 2018. Next generation firewall for network security: a survey. In: SoutheastCon 2018. IEEE, pp. 1–6.

Ng, D., Lin, A., Kashyap, A., Li, G., Lu, X., 2024. NVMe-oPF: Designing efficient priority schemes for NVMe-over-fabrics with multi-tenancy support. In: 2024 IEEE International Parallel and Distributed Processing Symposium. IPDPS, IEEE, pp. 519–531.

Ni, Z., 2022. A SmartNIC-Based Network Processing Framework for High Performance NFV Platforms (Ph.D. thesis). The George Washington University.
Ni, Z., Liu, G., Afanasev, D., Wood, T., Hwang, J., 2019. Advancing network function virtualization platforms with programmable nics. In: 2019 IEEE International Symposium on Local and Metropolitan Area Networks. LANMAN, IEEE, pp. 1–6.
Nickel, M., Göhringer, D., 2024. A survey on architectures, hardware acceleration and challenges for In-Network computing. ACM Trans. Reconfigurable Technol. Syst..
Ning, J., Huang, X., Poh, G.S., Xu, S., Loh, J.-C., Weng, J., Deng, R.H., 2020. Pine: Enabling privacy-preserving deep packet inspection on TLS with rule-hiding and fast connection establishment. In: Computer Security–ESORICS 2020: 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14–18, 2020, Proceedings, Part I 25. Springer, pp. 3–22.
Ning, J., Poh, G.S., Loh, J.-C., Chia, J., Chang, E.-C., 2019. PrivDPI: Privacy-preserving encrypted traffic inspection with reusable obfuscated rules. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. pp. 1657–1670.
Nir, Y., Langley, A., 2018. ChaCha20 and Poly1305 for IETF Protocols. Tech. Rep..
Novais, F.A., Verdi, F.L., 2024. Unlocking security to the board: An evaluation of SmartNIC-driven TLS acceleration with kTLS. In: NOMS 2024-2024 IEEE Network Operations and Management Symposium. IEEE, pp. 1–9.
NVIDIA, Free DLI Course: Introduction to DOCA for DPUs, [Online]. Available: https://developer.nvidia.com/blog/free-dli-course-introduction-to-doca-for-dpus/.
NVIDIA, DOCA: Data center infrastructure on chip architecture, NVIDIA Developer Documentation. Available: https://developer.nvidia.com/doca.
Orosz, P., Tóthfalusi, T., Varga, P., 2018. FPGA-assisted DPI systems: 100 Gbit/s and beyond. IEEE Commun. Surv. Tutor. 21 (2).
P4 Language Consortium, P4: Programming protocol-independent packet processors, P4.org Official Documentation. Available: https://p4.org.
Pacífico, R.D., Vieira, M.A., Duarte, L.F., Nacif, J.A., 2022. Function as a service offloaded to a SmartNIC. In: 2022 IEEE Latin-American Conference on Communications. LATINCOM, IEEE, pp. 1–6.
Palo Alto Networks, I., Palo Alto networks firewalls, [Online]. Available: https://docs.aviatrix.com/previous/documentation/latest/firewall-and-security/index.html.
Pan, X., An, Y., Liang, S., Mao, B., Zhang, M., Li, Q., Jung, M., Zhang, J., 2024. Flagger: Cooperative acceleration for large-scale cross-silo federated learning aggregation. In: 2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture. ISCA, IEEE, pp. 915–930.
Panda, S., Feng, Y., Kulkarni, S.G., Ramakrishnan, K., Duffield, N., Bhuyan, L.N., 2021. Smartwatch: Accurate traffic analysis and flow-state tracking for intrusion prevention using smartnics. In: Proceedings of the 17th International Conference on Emerging Networking EXperiments and Technologies. pp. 60–75.
Pao, D., Lin, W., Liu, B., 2010. A memory-efficient pipelined implementation of the aho-corasick string-matching algorithm. ACM Trans. Archit. Code Optim. (TACO) 7 (2), 1–27.
Papadogiannaki, E., Tsirantonakis, G., Ioannidis, S., 2022. Network intrusion detection in encrypted traffic. In: 2022 IEEE Conference on Dependable and Secure Computing. DSC, IEEE, pp. 1–8.
Park, S.J., Govindan, R., Shen, K., Culler, D., Özcan, F., Kim, G.-W., Levy, H., 2024. Lovelock: Towards smart nic-hosted clusters. ACM SIGENERGY Energy Inform. Rev. 4 (5), 172–179.
Patel, N., Mamandipoor, A., Nouri, M., Alian, M., 2024. SmartDIMM: In-memory acceleration of upper layer protocols. In: 2024 IEEE International Symposium on High-Performance Computer Architecture. HPCA, IEEE, pp. 312–329.
Patetta, M., Secci, S., Taktak, S., Line-rate botnet detection with smartnic-embedded feature extraction, Available at SSRN 4717964.
Pereira, F., Ramos, F.M., Pedrosa, L., 2024. Automatic parallelization of software network functions. In: 21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24). pp. 1531–1550.
Pismenny, B., Eran, H., Yehezkel, A., Liss, L., Morrison, A., Tsafrir, D., 2021. Autonomous NIC offloads. In: Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems. pp. 18–35.
Poddar, R., Babu, H., 2022. Decision tree based IoT attack detection in programmable data plane using P4 language. In: International Conference on Advanced Information Networking and Applications. Springer, pp. 671–683.
Qiu, Y., Xing, J., Hsu, K.-F., Kang, Q., Liu, M., Narayana, S., Chen, A., 2021. Automated smartnic offloading insights for network functions. In: Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles. pp. 772–787.
Rambus, I., 2024. Public key encryption IP-85. https://www.rambus.com/security/crypto-accelerator-cores/pke-ip-85/. (Accessed 20 July 2024).
Rescorla, E., 2018. The Transport Layer Security (TLS) Protocol Version 1.3. Request for Comments 8446, RFC Editor, URL https://www.rfc-editor.org/info/rfc8446.
Ricci, R., Eide, E., Team, C., 2014. Introducing CloudLab: Scientific infrastructure for advancing cloud architectures and applications. Login:: Mag. USENIX SAGE 39 (6), 36–38.
Rivitti, A., Tulumello, A., Bianchi, G., 2024. The case for ultra high speed portable network security filters. In: ITASEC 2024: The Italian Conference on CyberSecurity.
Roesch, M., et al., 2011. Snort-network intrusion detection & prevention system.
Rosa, L., Foschini, L., Corradi, A., 2024. Empowering cloud computing with network acceleration: A survey. IEEE Commun. Surv. Tutor..
Rothenberger, B., Taranov, K., Perrig, A., Hoefler, T., 2021. ReDMArk: Bypassing RDMA security mechanisms. In: 30th USENIX Security Symposium (USENIX Security 21). pp. 4277–4292.
Sacco, A., Angi, A., Marchetto, G., Esposito, F., 2023. P4FL: An architecture for federating learning with In-Network processing. IEEE Access.
Salopek, D., Mikuc, M., 2023. Enhancing mitigation of volumetric DDoS attacks: A hybrid FPGA/Software filtering datapath. Sensors 23 (17), 7636.
Scarfone, K., Mell, P., et al., 2007. Guide to intrusion detection and prevention systems (idps). NIST Spec. Publ. 800 (2007), 94.
Seo, H., Shin, S., Lee, S., 2024. BotFence: A framework for network-enriched botnet detection and response with SmartNICs. IEEE Access.
Shan, Y., Lin, W., Guo, Z., Zhang, Y., 2022. Towards a fully disaggregated and programmable data center. In: Proceedings of the 13th ACM SIGOPS Asia-Pacific Workshop on Systems. pp. 18–28.
Shantharama, P., Thyagaturu, A.S., Reisslein, M., 2020. Hardware-accelerated platforms and infrastructures for network functions: A survey of enabling technologies and research studies. IEEE Access 8, 132021–132085.
Sheeraz, M., Durad, M.H., Tahir, S., Tahir, H., Saeed, S., Almuhaideb, A.M., 2024. Advancing Snort IPS to achieve line rate traffic processing for effective network security monitoring. IEEE Access 12, 61848–61859.
Sherry, J., Hasan, S., Scott, C., Krishnamurthy, A., Ratnasamy, S., Sekar, V., 2012. Making middleboxes someone else's problem: Network processing as a cloud service. ACM SIGCOMM Comput. Commun. Rev. 42 (4), 13–24.
Simpson, A.K., Szekeres, A., Nelson, J., Zhang, I., 2020. Securing $\{RDMA\}$ for $\{High-Performance\}$ datacenter storage systems. In: 12th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 20).
Sivaraman, V., Narayana, S., Rottenstreich, O., Muthukrishnan, S., Rexford, J., 2017. Heavy-hitter detection entirely in the data plane. In: Proceedings of the Symposium on SDN Research. pp. 164–176.
Song, E., Yu, N., Pan, T., Fu, Q., Xu, L., Wei, X., Qiao, Y., Lu, J., Dong, Y., Xie, M., et al., 2022. Mimic: Smartnic-aided flow backpressure for cpu overloading protection in multi-tenant clouds. In: 2022 IEEE 30th International Conference on Network Protocols. ICNP, IEEE, pp. 1–11.
Sriman, B., Ganesh Kumar, S., Shamili, P., 2021. Blockchain technology: Consensus protocol proof of work and proof of stake. In: Intelligent Computing and Applications: Proceedings of ICICA 2019. Springer, pp. 395–406.
Srivastava, A., Gupta, B.B., Tyagi, A., Sharma, A., Mishra, A., 2011. A recent survey on ddos attacks and defense mechanisms. In: International Conference on Parallel Distributed Computing Technologies and Applications. Springer, pp. 570–580.
Stratosphere Lab, Malware capture facility project, [Online]. Available: https://www.stratosphereips.org/datasets-mixed.
Sundar, N., Burres, B., Li, Y., Minturn, D., Johnson, B., Jain, N., 2023. 9.4 An In-depth look at the intel IPU E2000. In: 2023 IEEE International Solid-State Circuits Conference. ISSCC, IEEE, pp. 162–164.
Sunny, F.A., Hajek, P., Munk, M., Abedin, M.Z., Satu, M.S., Efat, M.I.A., Islam, M.J., 2022. A systematic review of blockchain applications. Ieee Access 10, 59155–59177.
Suricata, S., 2021. Suricata: open source IDS/IPS/NSM engine.
Taranov, K., Rothenberger, B., De Sensi, D., Perrig, A., Hoefler, T., 2022. NeVerMore: Exploiting RDMA mistakes in NVMe-oF storage applications. In: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security. pp. 2765–2778.
Taranov, K., Rothenberger, B., Perrig, A., Hoefler, T., 2020. sRDMA–efficient NIC-based authentication and encryption for remote direct memory access. In: 2020 USENIX Annual Technical Conference (USENIX ATC 20). pp. 691–704.
Tasdemir, K., Khan, R., Siddiqui, F., Sezer, S., Kurugollu, F., Bolat, A., 2023. An investigation of machine learning algorithms for high-bandwidth SQL injection detection utilising BlueField-3 DPU technology. In: 2023 IEEE 36th International System-on-Chip Conference. SOCC, IEEE, pp. 1–6.
The Linux Foundation, DPDK, [Online]. Available: https://www.dpdk.org/.
Thinh, T.N., Hieu, T.T., Kittitornkun, S., et al., 2012. A FPGA-based deep packet inspection engine for network intrusion detection system. In: 2012 9th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology. IEEE, pp. 1–4.
Tsai, S.-Y., Payer, M., Zhang, Y., 2019. Pythia: remote oracles for the masses. In: 28th USENIX Security Symposium (USENIX Security 19). pp. 693–710.
Tschofenig, H., Pegourie-Gonnard, M., Vincent, H., 2015. Performance of State-of-the-Art cryptography on ARM-based microprocessors. In: NIST Lightweight Cryptography Workshop. 2015.
Tuaf, T.T.B., Gilboa, T., 2020. kTLS offload performance enhancements for real-life applications. [Online]. Available URL https://netdevconf.info/0x14/pub/papers/29/0x14-paper29-talk-paper.pdf.
Turkovic, B., Oostenbrink, J., Kuipers, F., Keslassy, I., Orda, A., 2020. Sequential zeroing: Online heavy-hitter detection on programmable hardware. In: 2020 IFIP Networking Conference (Networking). IEEE, pp. 422–430.
VenkataKeerthy, S., Andaluri, Y., Dey, S., Shah, R., Tammana, P., Upadrasta, R., 2022. Packet processing algorithm identification using program embeddings. In: Proceedings of the 6th Asia-Pacific Workshop on Networking. pp. 76–82.
Viegas, P., de Castro, A., Lorenzon, A., Rossi, F., Luizelli, M., 2021. The actual cost of programmable SmartNICs: Diving into the existing limits. In: International Conference on Advanced Information Networking and Applications.

Vigna, G., Kemmerer, R., 1998. NetSTAT: A network-based intrusion detection approach. In: Proceedings 14th Annual Computer Security Applications Conference.

VMware, VMware SDDC micro-segmentation white paper, [Online]. Available: https://blogs.vmware.com/networkvirtualization/files/2014/06/VMware-SDDC-Micro-Segmentation-White-Paper.pdf.

VMware, VMware distributed firewalls, [Online]. Available: https://tinyurl.com/mrxn8s56.

Wagner, N., Şahin, C.Ş., Winterrose, M., Riordan, J., Pena, J., Hanson, D., Streilein, W.W., 2016. Towards automated cyber decision support: A case study on network segmentation for security. In: 2016 IEEE Symposium Series on Computational Intelligence. SSCI, IEEE, pp. 1–10.

Wang, H., Soulé, R., Dang, H.T., Lee, K.S., Shrivastav, V., Foster, N., Weatherspoon, H., 2017. P4fpga: A rapid prototyping framework for p4. In: Proceedings of the Symposium on SDN Research. pp. 122–135.

Wu, M., Matsutani, H., Kondo, M., 2022. ONLAD-IDS: ONLAD-based intrusion detection system using SmartNIC. In: 2022 IEEE 24th Int Conf on High Performance Computing & Communications; 8th Int Conf on Data Science & Systems; 20th Int Conf on Smart City; 8th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys). IEEE, pp. 546–553.

Wu, S., Su, Q., Niu, Z., Xu, H., 2024. Tomur: Traffic-aware performance prediction of on-NIC network functions with multi-resource contention. arXiv preprint arXiv:2405.05529.

Xavier, B.M., Guimarães, R.S., Comarela, G., Martinello, M., 2021. Programmable switches for in-networking classification. In: IEEE INFOCOM 2021-IEEE Conference on Computer Communications. IEEE, pp. 1–10.

Xilinx, Xilinx open NIC repository, [Online]. Available: https://github.com/Xilinx/open-nic.

Xin, Y., Jia, C., Li, W., Rottenstreich, O., Xu, Y., Xie, G., Tian, Z., Li, J., 2024. A heterogeneous and adaptive architecture for decision-tree-based ACL Engine on FPGA. IEEE Trans. Comput..

Xing, J., Qiu, Y., Hsu, K.-F., Sui, S., Manaa, K., Shabtai, O., Piasetzky, Y., Kadosh, M., Krishnamurthy, A., Ng, T.E., et al., 2023. Unleashing SmartNIC packet processing performance in P4. In: Proceedings of the ACM SIGCOMM 2023 Conference. pp. 1028–1042.

Xu, Q., Siyamwala, H., Ghosh, M., Suri, T., Awasthi, M., Guz, Z., Shayesteh, A., Balakrishnan, V., 2015. Performance analysis of NVMe SSDs and their implication on real world databases. In: Proceedings of the 8th ACM International Systems and Storage Conference. pp. 1–11.

Yan, J., Tang, L., Li, J., Yang, X., Quan, W., Chen, H., Sun, Z., 2019. UniSec: a unified security framework with SmartNIC acceleration in public cloud. In: Proceedings of the ACM Turing Celebration Conference-China. pp. 1–6.

Yang, W., Chang, C., 2023. SmartGate: Accelerate cloud gateway with SmartNIC. In: Proceedings of the 8th International Conference on Cyber Security and Information Engineering. pp. 8–14.

You, M., Nam, J., Seo, M., Shin, S., 2023. HELIOS: Hardware-assisted high-performance security extension for cloud networking. In: Proceedings of the 2023 ACM Symposium on Cloud Computing. pp. 486–501.

You, M., Seo, M., Kim, J., Shin, S., Nam, J., 2024. Hyperion: Hardware-based high-performance and secure system for container networks. IEEE Trans. Cloud Comput..

Zang, S., Fei, J., Ren, X., Wang, Y., Cao, Z., Wu, J., 2022. A SmartNIC-Based secure aggregation scheme for federated learning. In: Proceedings of the 3rd International Conference on Computer Engineering and Intelligent Control (ICCEIC2022).

Zeek, L., 2020. The zeek network security monitor.

Zhang, B., Davis, P.E., Morales, N., Zhang, Z., Teranishi, K., Parashar, M., 2023. Optimizing data movement for GPU-based in-situ workflow using gpudirect RDMA. In: European Conference on Parallel Processing. Springer, pp. 323–338.

Zhang, K., Samaan, N., Karmouch, A., 2024. A machine learning-based toolbox for P4 programmable data-planes. IEEE Trans. Netw. Serv. Manag..

Zhao, Z., Atre, N., Sadok, H., Sahay, S., Obla, S., Hoe, J.C., Sherry, J., 2022. Pigasus 2.0: making the pigasus IDS robust to attacks and different workloads. In: Proceedings of the SIGCOMM'22 Poster and Demo Sessions. pp. 61–62.

Zhao, J., Neves, M., Haque, I., 2023. On the (dis) advantages of programmable NICs for network security services. In: 2023 IFIP Networking Conference (IFIP Networking). IEEE, pp. 1–9.

Zhao, Z., Sadok, H., Atre, N., Hoe, J.C., Sekar, V., Sherry, J., 2020. Achieving 100gbps intrusion prevention on a single server. In: 14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20). pp. 1083–1100.

Zheng, C., Hong, X., Ding, D., Vargaftik, S., Ben-Itzhak, Y., Zilberman, N., 2023. In-network machine learning using programmable network devices: A survey. IEEE Commun. Surv. Tutor..

Zhou, Y., Wilkening, M., Mickens, J., Yu, M., 2024. SmartNIC security isolation in the cloud with S-NIC. In: Proceedings of the Nineteenth European Conference on Computer Systems. pp. 851–869.

Zou, Y., Pan, T., Lu, L., Li, Z., Yao, K., Huang, T., Liu, Y., 2023. P4RSS: Load-aware intra-server load balancing with programmable switching ASICs. In: ICC 2023-IEEE International Conference on Communications. IEEE, pp. 1893–1898.