On the (dis)Advantages of Programmable NICs for Network Security Services

Jack Zhao, Miguel Neves, Israat Haque
Dalhousie University

***Abstract*—Emerging programmable network interface cards (a.k.a. SmartNICs) are a viable alternative to reduce the gap between network bandwidths, currently at the scale of multi-hundred Gbps, and the server CPU processing capacity. This has rapidly led to many efforts exploring SmartNICs for offloading or accelerating applications that traditionally run solely on servers (e.g., key-value stores, data analytics). Despite the success of this paradigm, the suitability of SmartNICs for running security applications, specially those that heavily rely on cryptographic operations, still remains largely unstudied. In this paper, we aim at filling this gap and provide the first in-depth analysis of current SmartNICs' crypto capabilities. Our experiments with an ARM-based multi-core SmartNIC show that the device depends heavily on architecture enhancements (e.g., cryptographic instructions and hardware accelerators) to meet server performance on crypto-workloads. Moreover, data movements between the SmartNIC and crypto-hardware accelerator cores can introduce significant overhead and make the latter ineffective, particularly for short living tasks. From a service perspective, SmartNICs can take advantage of their privileged position (i.e., closer to client devices than server CPUs) to speed up crypto-based functions. However, the SmartNIC benefits can be easily outweighed if the application is too much data-intensive or includes several non-crypto tasks.**

***Index Terms*—SmartNIC, cryptography, measurements, network security**

I. Introduction

The gradual slowdown in CPU performance improvements on recent years has transformed programmable accelerators (e.g., GPUs, programmable SSDs, SmartNICs) into the focus for continued scaling of application performance [1]. In particular, emerging SmartNICs have demonstrated to be very helpful to ease the host CPU from expensive computation tasks and thus are becoming commonplace specially in cloud environments. Current SmartNICs from major vendors (e.g., NVIDIA, Broadcom, Netronome) comprise powerful computing resources, including multicore processors, onboard SRAM/DRAM, customized hardware accelerators for compression and crypto tasks, and programmable DMA engines. This has led to multiple research efforts exploring SmartNICs for (partially) offloading various applications such as load balancing, key-value stores, distributed transactions, and more [2].

Despite the success of this paradigm, little is known about the capabilities of SmartNICs to run crypto-based applications. On one hand, SmartNICs are usually equipped with cryptographic hardware accelerators that can process data at high rates. On the other hand, they have wimpy cores compared to host CPUs, and the need to interleave computations on different SmartNIC components (e.g., a processor core and a hardware accelerator) requires data movements that can be costly in practice.

High-level insights on offloading crypto tasks to SmartNICs are often scattered across the literature. For example, Taranov et al. [3] have found Broadcom Stingray SmartNICs can sustain high processing rates when running message authentication ciphers (e.g, AES and SHA). Cui et al. [4], on their turn, offload the handshake of TLS sessions to NVIDIA BlueField NICs with high performance. Kim et al. [5] adopt a "flipped" model and rather offload the TLS data encryption/decryption to Marvell LiquidIO III boards. None of these efforts provide a systematic analysis of the crypto capabilities from current SmartNICs though, and the answers to questions like "when one should offload a crypto task to a SmartNIC" and "which kind of crypto tasks should be offloaded" remain to be answered.

In this paper, we aim to start filling this gap and provide the *first in-depth analysis of current SmartNIC's crypto capabilities*. We start by characterizing the cryptographic functionalities of six commodity SmartNICs based on vendor information and exploring the most common crypto-acceleration approaches found on current designs (§III). Next, we extensively benchmark the performance of a multicore system-on-chip (SoC) SmartNIC for basic cryptographic primitives (e.g., symmetric and asymmetric ciphers). Based on our findings, we discuss the most important factors that drive the adoption of SmartNICs for accelerating crypto tasks (§IV). Finally, we assess to which extent current SoC-based SmartNICs can support offloading several crypto-intensive applications including virtual private network (VPN) tunneling and secure web serving (§V).

Our main findings are as follows:

- Current SoC-based SmartNICs depend heavily on architecture enhancements (e.g., cryptographic instructions and hardware accelerators) to meet server performance while running crypto-workloads. In particular, our ARM-based SoC NIC can offer up to 25% better throughput than servers when computing cryptography hashes thanks to dedicated instructions.
- Algorithmic optimizations to cryptographic primitives (e.g., ECC-based digital signatures) combined with data movement overheads between SmartNIC and accelerator cores can make on-board crypto-hardware accelerators ineffective. Specifically, we observed slow-downs of up

ISBN 978-3-903176-57-7

TABLE I: Architectural specifications of main commodity SmartNICs.

SmartNIC model	Vendor	SoC	CPU	FPGA	NPU	Processor	On/Off path
LiquidIO III CN96XX	Marvell	OCTEON TX2	✓			Arm v8.2, 36 cores, 2.4 GHz	On
Agilio LX	Netronome	NFP-6000			✓	Flow Processing Core (FPC), 120 cores, 1.2 GHz	On
BlueField 1M332A	NVIDIA	BlueField-1	✓			Arm Cortex-A72, 16 cores, 0.8 GHz	Off
Stingray PS225	Broadcom	BCM58802H	✓			Arm Cortex-A72, 8 cores, 3.0 GHz	Off
DSC-100	Pensando	Capri	✓		✓	NPU: Match Processing Unit, 112 cores, 0.83 GHz CPU: Arm Cortex-A72, 4 cores, 3.0 GHz	On: NPU Off: CPU
Alveo SN1000	Xilinx	Alveo	✓	✓		FPGA: XCU26 CPU: NXP Layerscape LX2162A	On: FPGA Off: CPU

to 91% for ECDSA when using an on-board accelerator on our testing NIC.

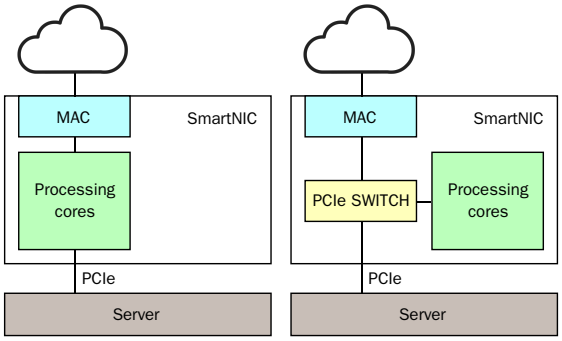
- SoC-based SmartNICs can take advantage of their privileged position (i.e., closer to client devices than server CPUs) to speed up crypto-intensive distributed applications. We found latency reductions of up to 50% for short-living flows on a VPN setup, even though our testing SmartNIC has no hardware support for IPSec acceleration.
- The benefits of crypto-acceleration mechanisms on SoC-based SmartNIC designs can be easily outweighed if a crypto-application is too much data-intensive or includes several non-crypto tasks. Throughput stress tests revealed our testing SmartNIC is approximately 56% worse under high loads than a server CPU for a client authentication application, despite its high nominal traffic speeds (two 25 GbE ports).

We do *not* claim to be either exhaustive or generic in our analyses and rather acknowledge performance may vary greatly among SmartNICs depending on their architecture. Our findings still have broad implications to SmartNIC design and application offloading policies though. From a design perspective, SmartNIC vendors can use our findings to architect better solutions to support crypto-based workloads. For offloading policies, our results are particularly important to help offloading engines make more informed decisions. Overall, we hope our work can make the initial move towards a deeper examination of how to leverage SmartNICs for accelerating network security services. We provide an online repository [6] along with this paper containing all configurations we used in our tests, running logs and automated scripts for reproducibility. The repository also includes further benchmarks we did not add to the paper due to space limitations.

II. Background and Motivation

A. Overview on SmartNICs

SmartNICs are NICs that can perform custom processing of network traffic. Ultimately, this programmability enables them to offload tasks from the CPU and thus free the latter to work on more important tasks. A typical SmartNIC is equipped with on-board memory, DMA engines, accelerators (e.g., for crypto, compression and regular expression matching) and multiple processing cores. Table I summarizes the specification of six commercial SmartNICs from major vendors. As



(a) On-Path SmartNIC (b) Off-Path SmartNIC

Fig. 1: SmartNIC Architecture Comparison

we can observe, current SmartNICs reflect different design trade-offs with respect to performance and programmability. Some SmartNICs (e.g., Agilio, DSC and Alveo) are based on domain-specific processors and can achieve extremely fast processing speeds at the cost of requiring programmers to use native hardware primitives (e.g., in Micro-C or VHDL) to manipulate data. Others, such as BlueField and Stingray cards, can run a Linux-like operating system on a general-purpose CPU, which lowers the barrier for application development but slows down data processing compared to their counterparts (e.g., due to context switching).

SmartNICs can be further categorized based on how their processing cores interact with traffic. *On-path* SmartNICs have their processing cores on the data path between the network port and the host processor (Figure 1(a)). As a result, the SmartNIC processor interacts with every packet received or transmitted by the host. Usually, on-path designs require beefier processors to not harm the flow's performance. LiquidIO and Agilio are examples of on-path SmartNICs. *Off-path* SmartNICs, on their turn, can let packets bypass the SmartNIC processor based on forwarding rules installed on a NIC-level switching fabric, usually a PCIe switch (Figure 1(b)). Mellanox BlueField and Broadcom Stingray are off-path SmartNICs. Finally, it is also possible to find hybrid designs (e.g., DSC, Alveo) in which the SmartNIC has both on-path and off-path modules. In this case, users can choose the best place to run each packet processing job based on its main characteristics.

TABLE II: Overview of crypto-acceleration features on commodity SmartNICs. ● = Crypto-hardware acceleration, ◐ = Processor acceleration, ○ = No acceleration.

Feature \ SmartNIC/ Accelerator	**LiquidIO III** NITROX V	**Agilio LX** Custom	**BlueField** Rambus EIP-154	**Stingray PS225** FlexSPARX 4	**DSC-100** PenTrust + PenAccel	**Alveo SN1000** Custom
AES	●	●	◐	●	●	●
ChaCha20/Poly1305	○	○	○	○	○	●
RSA	●	○	●	●	●	●
DSA	○	○	●	●	●	●
ECDSA/ECDH	○	○	●	●	●	●
SHA-256	●	○	◐	●	●	●
SHA-512	◐	○	○	○	●	●
TRNG	○	○	●	●	●	●

B. Why offloading crypto operations?

Crypto operations (e.g., hashing, encryption, decryption) are often costly to perform on server CPUs, to the point various manufacturers (e.g., Intel, AMD) deploy dedicated instructions for certain crypto primitives such as the AES and SHA algorithms. Increasing computational demands from other applications like deep learning and video processing have put a tremendous pressure on the already scarce CPU resources though, leading the community to look for alternative processing means for additional workloads. SmartNICs have plenty of mechanisms for crypto-related processing (as we discuss in the next section), which come at a relatively affordable price. Therefore, it is intuitive to consider them as a primary source for running crypto tasks.

III. SmartNIC crypto capabilities

Current multicore SoC SmartNICs exhibit varying crypto-acceleration capabilities (Table II). That includes support for multiple types of symmetric and asymmetric ciphers, hash functions and random number generators. We can further classify crypto-support from SmartNICs into three types: crypto-hardware acceleration, processor acceleration and software optimizations.

Crypto-hardware acceleration. Most of the SmartNICs we surveyed support some form of crypto-hardware acceleration. These are hardware implementations of cryptographic algorithms that can offer significantly higher performance and power-efficiency compared to their software counterparts [7]. Even though FPGA-based accelerators are widely discussed in the literature [8], our survey revealed production SmartNIC crypto-hardware accelerators rely predominantly on ASICs. Moreover, these ASICs can be either designed by the SmartNIC manufacturer itself (e.g., NITROX V from Marvell or FlexSPARX from Broadcom) or embedded into the SmartNIC as a third-party module (e.g., Rambus[1] modules in NVIDIA BlueField). Communication between the SmartNIC CPU and the crypto-hardware accelerator usually happens through the PCIe bus.

Processor acceleration. SmartNIC CPUs can have embedded support for cryptographic primitives in the form of dedicated instructions (i.e., at the ISA level). For example, production ARM cores have built-in instructions for computing heavy AES and SHA operations such as column mixes and hash updates, respectively [9]. The presence of these instructions varies depending on the processor architecture (e.g., ARMv8.2 has support for SHA3 and SHA-512 instructions, which are not present in the ARMv8 architecture). Moreover, the performance of cryptography instructions is also dependent on other processor aspects such as the level of data parallelism.

Software optimizations. Finally, current SmartNICs can also benefit from deployment optimizations of cryptographic software. These optimizations range from vectorized implementations [10] to optimized arithmetic operations (e.g., multi-precision multiplication [11]). Even though software optimizations are not targeted to SmartNICs, they have the advantage of lowering the cost for speeding up crypto applications on the latter as they do not require any hardware modification.

IV. Basic crypto performance

We start by analyzing the performance of a commodity SmartNIC when running basic crypto operations.

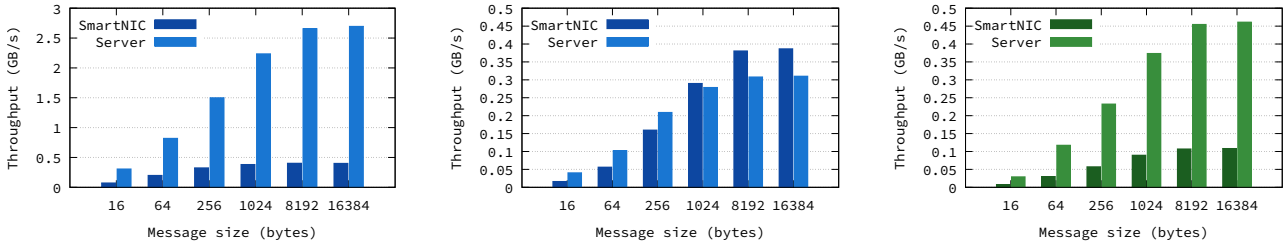
A. Measurement setup

We conduct our experiments on a single server containing a 10-core Intel Xeon Silver 4210R 2.40 GHz CPU with 32 GB of DDR4 DRAM. The server runs Ubuntu 20.04.1 LTS with kernel version 5.4.0 and has AES-NI instructions [12] enabled. It is equipped with a two-port 25GbE Mellanox BlueField-1 SmartNIC containing a 16-core ARMv8 Cortex-A72 0.8 GHz processor and 16 GB of DDR4 DRAM (same SmartNIC described in Table I). The SmartNIC runs a custom Linux kernel provided by Mellanox (version 5.4.44-mlnx.14.gd7fb187)[2].

We use the OpenSSL Speed benchmarking tool (version 1.1.1f) [13] in our experiments and test the performance of different cryptographic algorithms on both the server and the SmartNIC. More specifically, our study analyzes three classes of algorithms: symmetric and asymmetric ciphers as well as hash functions. For each class, we select a set of representative algorithms to compare and measure different resources according to the main algorithm's objective. In particular, we evaluate encryption and decryption performance for symmetric

[1] https://www.rambus.com/security/protocol-engines/high-speed-public-key-accelerator/

[2] https://github.com/Mellanox/bfb-build



(a) (b) (c)

Fig. 2: Throughput comparison for the *a)* AES-GCM; *b)* SHA-256; and *c)* SHA-512 cryptographic algorithms.

ciphers, sign and verify performance for asymmetric ones, and hashing speed for cryptographic hash functions. We consider throughput as the main metric in all cases. Each algorithm runs on a single thread and the results report an average of 10 repetitions.

B. Results

1) Symmetric ciphers: We first study the performance of the devices under test for different symmetric ciphers. In particular, we consider three ciphers: AES-256-GCM, AES-256-CBC and Chacha20-poly1305. Due to space constraints, we only report the results for AES-GCM in the paper. AES-CBC and Chacha-poly follow similar trends, and we refer the interested reader to our public repo [6] for more details. AES-GCM is currently the most common cipher in the TLS suite [14]. It provides authenticated encryption and can handle inputs (i.e., messages) of arbitrary length.

Figure 2(a) shows the encryption throughput (in GB/s) for AES-GCM as we vary the message size. The server throughput is consistently higher than the SmartNIC, which is expected as the SmartNIC does not support hardware acceleration for symmetric ciphers and the server has a beefier processor compared to it. That may not be the case when the SmartNIC supports hardware acceleration though, as Cui et. al [4] observe in a LiquidIO device. In this case, the SmartNIC can be up to 2x faster than the server thanks to its hardware accelerator, as the authors report for 1024-byte messages encrypted using the AES algorithm. Data decryption results follow a similar pattern.

Takeaways. Multicore SoC SmartNICs can perform comparably to servers when running encryption/decryption algorithms (e.g., AES-GCM encryption) thanks to hardware acceleration. Otherwise, the higher parallelism of server CPUs can easily lead to better performance and therefore task offloading is not recommended.

2) Hash functions: Cryptographic hash functions are a basic tool of modern cryptography with applications ranging from message integrity verification to Proof-of-Work in digital currencies (e.g., Bitcoin) [15]. In this work, we compare the performance of both the server and the SmartNIC when running two widely used cryptographic hash functions: SHA-256 and SHA-512 [14]. Figure 2(b) shows the throughput results for the former (in GB/s). As expected, the throughput increases with the message size since larger messages require less interactions with the system (e.g., less memory allocations). Interestingly, the SmartNIC outperforms the server for messages bigger than 1KB. The main reason stems from the fact ARMv8 processors can rely on cryptographic instructions to accelerate hash computations [16] which is not the case for Intel Cascade Lake ones (i.e., the server CPU architecture). Later versions of Intel Xeon processors such as the ones based on the Ice Lake and Rocket Lake architectures have incorporated support for SHA instructions though [17].

Figure 2(c) shows the throughput of both testing devices for SHA-512. Unlike SHA-256, the server significantly outperforms the SmartNIC for all message sizes. That is primarily due to the lack of support from ARMv8 processors to dedicated instructions for computing SHA-512, which were added later as part of the ARMv8.2 instruction set [16]. Moreover, as widely noticed [18], SHA-512 shows better performance than SHA-256 on 64-bit x86 machines (up to 49% better for 16KB messages on our tests). The main reason is the lower amount of round operations per byte required by the SHA-512 algorithm on 64-bit arithmetic units. For instance, SHA-512 requires 80 rounds over 128-byte blocks against 64 rounds over 64-byte blocks from SHA-256 when adopting 64-bit arithmetic. Ultimately, that translates into less instructions to be executed for the same amount of input data.

Takeaways. Modern SmartNICs depend heavily on architecture enhancements (e.g., cryptography instructions) to meet server performance on crypto-hashing workloads. Therefore task offloading engines should be aware of the underlying SmartNIC CPU architecture in addition to its running state and the characteristics of the workload.

3) Asymmetric ciphers: Next, we compare the performance of the server and SmartNIC while running different asymmetric ciphers, a.k.a. public-key primitives. Asymmetric ciphers are mostly used on message authentication (or digital signature) and key exchange tasks. Due to space constraints, we only report results for message authentication in this paper, and refer the interested reader to our public repo for more details on the SmartNIC performance for key exchange tasks.

To authenticate a message, a sender combines it with a private key to create a digital signature, which is then verified

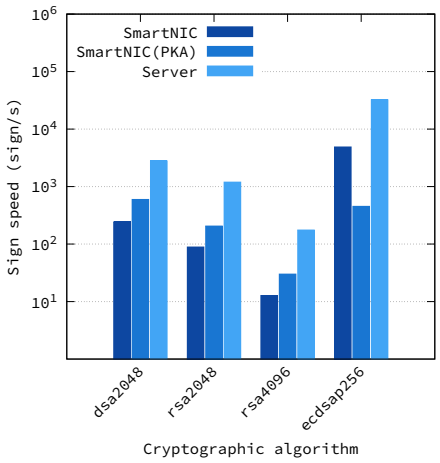


Fig. 3: Asymmetric Ciphers Throughput Comparison (Sign)

by the receiver using the sender's public key. Currently, the most popular digital signature algorithms are RSA, prime field DSA, and elliptic-curve DSA (ECDSA), all widely adopted in Internet protocols such as TLS and SSH [19]. RSA relies on integer factorizarion (i.e., finding the prime factors of a number) to provide the desired security strength while (EC)DSA is based on the discrete logarithm problem. We test key sizes above 2048 bits for RSA and DSA in our experiments as those are the minimum sizes currently recommended by NIST[3]. Note that ECC ciphers require smaller key sizes to provide the same security strength as RSA, so we test the former using 256-bit keys for a fair comparison.

Figure 3 shows the signing throughput for the three authentication algorithms under analysis on both the server and the SmartNIC. PKA (Public Key Acceleration engine) depicts the SmartNIC performance when its crypto-hardware accelerator is turned on. Unlike previous work, e.g. [5], in which the authors considered a less capable server, our server showed better performance for all scenarios we tested (more than 10x better depending on the scenario). Therefore, we find the server set up (e.g., CPU and memory architecture) plays an important role to determining whether offloading public-key operations to SmartNICs or not, even in the presence of crypto-hardware accelerators.

Interestingly, the SmartNIC performs worse when using its crypto-hardware accelerator compared to CPU-based processing for ECDSA. We believe that is due to two reasons: i) dispatching tasks to the crypto-hardware accelerator demands transferring data to (and synchronizing with) the latter, which introduces overhead; and ii) recent optimizations for supporting elliptic curves (particularly NIST P-256 curves) on ARM processors have made their cores significantly faster [20]. We observe similar trends for the signature verification throughput.

[3]https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57Pt3r1.pdf

Takeaways. Algorithmic (i.e., software) optimizations of cryptographic primitives can make SmartNIC crypto-hardware accelerators outdated. Further, data movements between SmartNIC and accelerator cores introduce non-negligible overhead, particularly for short-living tasks.

V. Case Studies

In this section, we analyze the benefits of offloading major security applications onto SmartNICs. More specifically, we study three applications: VPN tunneling, user authentication, and secure web serving. We chose these particular applications for several important reasons: (i) they represent cases from distinct domains where SmartNICs are widely adopted (e.g., public cloud data centers, university campuses, enterprise networks), (ii) they use different combinations of the security primitives we studied in Section IV, and (iii) they are available under an open-source license, enabling the community to easily reproduce the results from this work.

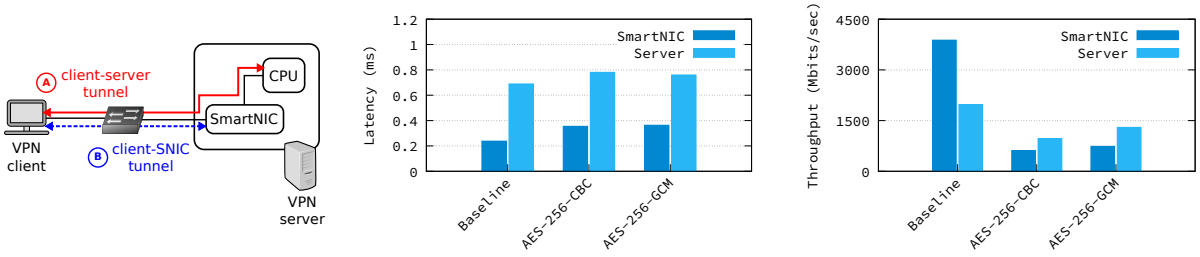
A. VPN tunneling

Internet users rely widely on virtual private network (VPN) services to preserve their privacy, circumvent censorship, and access geo-filtered content. While multiple VPN tunneling protocols exist (e.g., PPTP, SSTP, SSL, WireGuard), OpenVPN and IPSec still account for a significant portion of the available VPN services [21]. In this paper, we focus on the latter. IPSec is a policy-based VPN suite[4] that uses the Internet Key Exchange version 2 (IKEv2) protocol for traffic tunneling. IKEv2 works in two phases: first, the protocol establishes a security association (SA), i.e., a set of cryptographic attributes such as a shared secret and an encryption/decryption algorithm to securely carry IKE messages between the two communicating parties. After that, it creates an additional SA (or "child SA") for the two parties to authenticate and start exchanging data [23]. IPSec relies heavily on symmetric encryption for its private data exchange.

Figure 4(a) shows our experimental setup. The VPN client is equipped with an Intel Core i7-9700 @ 3.0 GHz CPU with 8 cores and 16 GB of DDR4 DRAM. It runs Ubuntu 18.04.6 LTS (kernel 5.4.0-90-generic). The server and SmartNIC are the same as described in Section IV-A. We compare two VPN tunneling scenarios: client-CPU (named "server") and client-SmartNIC. In both cases, we use StrongSwan (version U5.8.2/K5.4.0-74-generic) to deploy the VPN tunnels [24]. Both server and SmartNIC tunnels use a pre-shared key (PSK) and SHA1 for IKEv2 authentication. Moreover, we run two sets of experiments: i) the client sends 100 pings and measures the average round-trip latency; and ii) the same client generates a 1-minute long TCP flow using *iperf3* (version 3.1.3) and measures the average throughput. We repeat each experiment 10 times and report the results for AES-256-GCM and AES-256-CBC as data encryption/decryption algorithms[5].

[4]Policy-based VPNs adopt firewall rules rather than virtual network interfaces to determine which traffic belongs to the VPN [22].

[5]We omit results for Chacha20/Poly1305 as we could not run the associated StrongSwan plugin on the SmartNIC.



(a) (b) (c)

Fig. 4: *a)* VPN tunneling setup; *b)* Average round-trip latency in a VPN tunnel; *c)* Average TCP throughput in a VPN tunnel.

Latency. Figure 4(b) shows the average round-trip latency for the different VPN encryption/decryption algorithms. The "baseline" scenario depicts plaintext (ping) latency. We observe the SmartNIC can provide substantially lower latency (up to 52% lower in the GCM mode) compared to the server. This difference stems from the fact that the SmartNIC is "one hop" closer to the VPN client, i.e. packets do not need to cross the PCIe channel to reach out the NIC cores, which becomes more prominent in short-path scenarios (e.g., an edge data center) [25]. Moreover, the overhead compared to the baseline solution is relatively low (less than 35%), mainly due to the small amounts of data, i.e., a single ping packet, to be encrypted/decrypted by the VPN endpoints.

Throughput. Although the SmartNIC can perform well in low crypto load scenarios, its performance degrades significantly under high loads. Figure 4(c) shows the results of our throughput stress test. As expected, the SmartNIC exhibits markedly better performance compared to the server for plaintext TCP (i.e., non-VPN traffic), as it is a high-speed network device and no further processing is required from its CPU cores. However, we found an inverted trend when crypto-processing is in place. For instance, the SmartNIC and server throughput were around 739 and 1300 Mbps, respectively, for an AES-GCM-based VPN tunnel.

Takeaways. Current SoC-based SmartNICs can take advantage of their privileged position (i.e., closer to client devices) to speed up crypto-based network services. However, offloading crypto-tasks is only indicated for low load scenarios if the SmartNIC does not have a crypto-hardware accelerator. In particular, latency-critical applications can benefit more than bandwidth-intensive ones.

B. User authentication

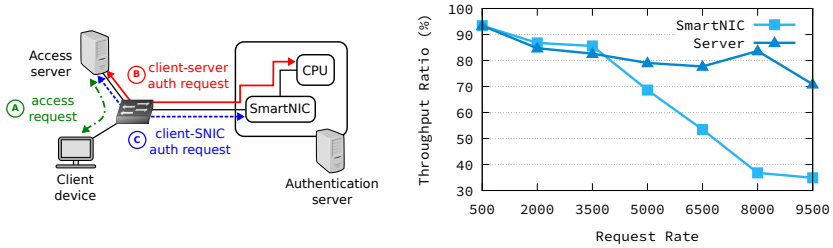
Several enterprises, ISPs and educational institutions deploy user authentication systems to control access to their IT resources. As such, it makes sense to consider taking advantage of recent SmartNICs to help alleviate the burden on server CPUs when running authentication servers. In a typical authentication system, a user (e.g., a VPN client or IoT device) issues an access request to a network access server. The access server then queries an authentication server to confirm the user credentials and grant/deny the user access to a particular network resource [26]. In order to process a query, the authentication server hashes the user credentials and checks them on either a local or remote data base.

Figure 5(a) depicts our evaluation setup. We compare two authentication scenarios: sending the user credentials to be authenticated on either the server CPU (scenario A-B, which we name "server") or the SmartNIC (scenario A-C). The network access server runs on the same machine described in Section V-A while the authentication server and SmartNIC are as presented in Section IV-A. We use *radperf* (version 2.0.1)[6] to simulate both the client device and access server, and make it issue requests to the authentication server following a desired rate. The authentication server, on its turn, processes requests using FreeRADIUS (version 3.0.20)[7]. We adopt the default FreeRADIUS configurations, which includes UDP as transport protocol and SHA-256 as the password hashing function. Moreover, the authentication server runs in a single core and in single-thread mode to enable a fair comparison between the SmartNIC and host CPU, as they have different numbers of cores and clock speeds. Our results show the average of 100 runs.

Latency. Table III compares each device's latency for processing a batch of 1K authentication requests issued at an unlimited rate. We find that the SmartNIC is significantly faster (up to 50% in the average) compared to the server. This is a combination of two factors: first, similar to the VPN scenario the SmartNIC is "closer" to the client device, i.e., the network access server, which reduces the overall communication latency. Second, the SmartNIC cores rely on specialized cryptographic instructions for accelerating hash computations and therefore can hash user credentials more efficiently as we saw in Section IV-B2. Interestingly, the SmartNIC also presents significantly shorter tail latencies (around 52% lower at the 99th percentile), which can be important if a workload is deadline-oriented.

Throughput. In addition to the bursty workload from the above setup, we also evaluated the authentication server in a sustained load scenario. In this case, Figure 5(b) shows how the request rate affects each device's throughput ratio, i.e., the proportion of the input load that can be served. The higher the ratio the better. We observe the SmartNIC hits its bottleneck

[6]https://networkradius.com/radius-performance-testing/
[7]https://freeradius.org/



(a) (b)

Fig. 5: *a)* User authentication setup; *b)* Throughput ratio (i.e., the ratio of served requests) as a function of the authentication request rate.

TABLE III: Average, 95th and 99th percentile of the round trip latency (in miliseconds) for processing a batch of 1K authentication requests.

Device	Average(ms)	95th	99th
SmartNIC	0.36	0.46	0.47
Server	0.71	0.90	0.91

faster (at around 3.5K requests per second - RPS). The server, on the other hand, can sustain a relatively high performance (beyond 70% of the input rate) up to 9.5K RPS, though its throughput ratio gradually decreases. We attribute this decrease mainly to the background processes (e.g., garbage collection) run by FreeRADIUS.

Takeaways. The speed-ups from specialized cryptography instructions (e.g., for computing SHA-256 hashes) on SmartNIC cores can be easily outweighed if the workload includes additional tasks. Therefore, it is important for offloading engines to assess the whole workload structure when offloading crypto applications.

C. Secure web serving

HTTPS adoption has grown substantially over the last few years [27]. This trend has sparked preliminary investigations (e.g., [5], [28]) on offloading TLS operations into the SmartNIC as an alternative to speed up secure web serving. Despite the existing efforts, a few important questions still remain open. We explore two of them in this paper: i) how would a web server perform considering a full (rather than partial) application offload; and ii) how would a fully offloaded server perform when using the newer TLS 1.3 version rather than the standard TLS 1.2.

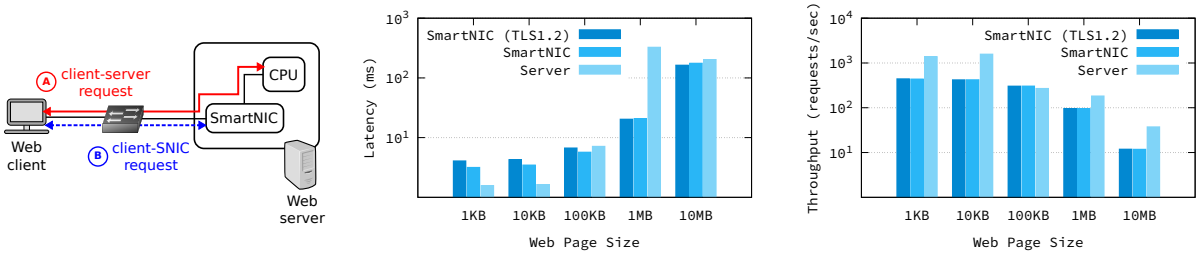
Figure 6(a) shows our experimental setup. The web server and the client run on the same machines as described in Sections IV-A and V-A, respectively. Our tests compare two scenarios, namely using either the server or the SmartNIC to handle HTTPS requests. We use NGINX (version 1.18.0) [29] for request serving and set it to run on a single worker[8] over a self-signed certificate. The certificate has a 4096-bit key and uses SHA-256 as its hash function. We use standard X.509 v3 extensions to specify the certificate security properties (e.g., key usage) and keep the remaining NGINX parameters as default. The web client runs the *wrk* benchmarking tool (version 4.2.0)[9]. We fix the number of parallel connections to 50 (i.e., the client will maintain 50 open connections at any given time throughout the experiment) and set *wrk* to run on a single-thread. Each TCP connection serves a single web page request and we run each experiment for three minutes. Our results report the average of 10 runs.

Latency. Figure 6(b) shows the average request latency for different web page sizes. Despite closer to the client, the SmartNIC performs worse than the server CPU for small requests (up to 10 KB). We believe that stems from the fact the SmartNIC uses its crypto-hardware accelerator for the connection setup (i.e., TLS handshake), which incurs in latency overhead as we saw in Section IV-B3. As the request size increases, however, server and SmartNIC latencies tend to become more similar. Interestingly, the SmartNIC significantly outperformed the server for 1 MB requests. Since NGINX fragments large web pages before sending them to the client [30], there is a trade-off between the number/size of segments and the overall request latency. More specifically, a larger number of smaller segments can benefit the SmartNIC, which has a wimpy processor but is closer to the user compared to the server CPU. We leave further investigation about the latency versus fragment size trade-off as future work. As expected, TLS 1.2 is slightly worse than its successor, specially for small pages, where the connection handshake cost is more prominent.

Throughput. Figure 6(c) illustrates how the throughput (in requests per second) evolves on both the server and the SmartNIC as we increase the web page size. As expected, the performance degrades on both devices for larger web pages because there is more data to encrypt and transmit. Additionally, the server consistently outperforms the SmartNIC. In particular, it shows higher throughput on scenarios involving either extensive connection setups or heavy data encryption (1 KB and 10 MB in our experiments, respectively). That is likely the result of a more diverse set of crypto-acceleration mechanisms (e.g., dedicated instructions, extensive data parallelism) on the server, which can efficiently address the needs of different security primitives. Current SmartNICs, on the other hand, usually excel on a restricted set of crypto algorithms that not always fit the requirements of heterogeneous workloads.

[8]One NGINX worker can process multiple requests in parallel, not necessarily as part of the same process.

[9]ttps://github.com/wg/wrk/releases/tag/4.2.0



(a) (b) (c)

Fig. 6: *a)* HTTPS server setup; *b)* Average web server latency; *c)* Average web server throughput.

Takeaways. Limited support of crypto-acceleration features on current SmartNICs are particularly harmful for applications deploying multiple cryptographic algorithms (e.g., both symmetric and asymmetric ciphers). For a secure web server, that meant up to 73% throughput drop in our case.

VI. Discussion

Modern ciphers. This study covers a broad range of crypto primitives that are widely supported on current (co)processors. However, novel and updated applications are increasingly making use of "modern" cryptographic algorithms (e.g., post-quantum and fully homomorphic ciphers), which do not find the same support on current hardware. Future work could explore to which extent commodity multi-core SmartNICs could be used for offloading post-quantum and FHE-based tasks.

Other security features. While this study is a first step towards understanding the design and capabilities of cryptographic modules on current SmartNICs, we believe the methods in this paper could be extended to assess other security features available on the same devices. For instance, future research could explore the raw performance of hardware-based random number generators present on SmartNICs and their impact on related applications (e.g., crypto key generation, smart contracts). Trusted execution environments (TEEs) such as ARM TrustZone [31] are another intriguing feature commonly supported on SmartNICs and that deserve further analysis. In particular, different SoCs support different TEE architectures, which can impact the performance of applications offloaded to these environments.

VII. Related work

Offloading applications to SmartNICs. There is a growing body of research on offloading applications to SmartNICs. As we mentioned in the introduction, some of these works provide ad hoc conclusions on the challenges and opportunities of offloading certain crypto operations to SmartNICs. However, none of them performs a systematic and in-depth analysis of the crypto-offloading space. iPipe [32] and E3 [2] propose frameworks for accelerating generic applications using SmartNICs. These frameworks are orthogonal to our work and can take advantage from our insights to make more informed decisions.

SmartNIC performance evaluation. A few recent efforts have studied the performance of commodity SmartNICs while running different network functions. Katsikas et al. [33] analyze the performance of basic packet classification tasks (e.g., header matching and state updates) on four SmartNICs from a major vendor. Liu et al. [34] test the capabilities of a BlueField-2 SmartNIC for different computational tasks, including memory and a few basic crypto operations (e.g., SHA-256, AES-XTS). Qiu et al. [35] and Krude et al. [36] propose automated frameworks to predict the performance of a network function when the latter runs on a SmartNIC. Xing et al. [37] characterize the performance of serverless applications on two SmartNICs from different vendors. Unlike these efforts, our paper concentrates on the crypto capabilities of current SmartNICs and for the first time provides detailed insights on this matter.

In-network cryptography. Efficient techniques have been proposed to speed up certain crypto tasks by offloading them to network devices. P4-IPsec [38] uses P4 switches to run specific IPSec components (e.g., ESP header manipulation) directly in the network data plane. The author in [39], on its turn, deploys AES encryption on P4-enabled devices while SipID [40] implements a keyed hash function (HalfSipHash) on programmable switches. Despite their high processing speeds, network switches have limited support for crypto operations, which hinders their adoption as crypto-accelerators. SmartNICs offer better support for crypto offloads compared to network switches and therefore have also been explored as target devices. SmartTLS [5] offloads the handshake of TLS connections to SmartNICs, which reduces the burden on server CPUs when dealing with high-amounts of short-living flows. sRDMA [3] uses a SmartNIC to perform authentication and encryption of RDMA packets. Even though these proposals provide relevant insights about the SmartNIC performance when running crypto workloads, their results are restricted to particular use cases. Scholz et al. [41] evaluate the performance of NPU and FPGA-based NICs while running various cryptographic hash functions. Our study complements their work by considering multiple cryptographic operations and applications.

VIII. Conclusion

This paper provides the first in-depth analysis of current SmartNIC support for crypto-based workloads. We characterize the cryptographic functionalities of six commodity SmartNICs and extensively assess the performance of an ARM-based device for both basic crypto operations and higher-level security applications. Our results show that current SoC-based SmartNIC designs can be beneficial for latency-sensitive tasks, but requires caution with computationally heavy ones.

Acknowledgements. We would like to thank the anonymous reviewers for their valuable feedback. This research is supported in part by NSERC under a Discovery Grant and CFI under a JELF grant.

References

[1] J. Nider and A. Fedorova, "The last cpu," in *Proceedings of the Workshop on Hot Topics in Operating Systems*, 2021, pp. 1–8.

[2] M. Liu, S. Peter, A. Krishnamurthy, and P. M. Phothilimthana, "E3: Energy-efficient microservices on smartnic-accelerated servers," in *2019 USENIX Annual Technical Conference (USENIX ATC'19)*, 2019, pp. 363–378.

[3] K. Taranov, B. Rothenberger, A. Perrig, and T. Hoefler, "srdma–efficient nic-based authentication and encryption for remote direct memory access," in *2020 USENIX Annual Technical Conference (USENIX ATC'20)*, 2020, pp. 691–704.

[4] T. Cui, W. Zhang, K. Zhang, and A. Krishnamurthy, "Offloading load balancers onto smartnics," in *Proceedings of the 12th ACM SIGOPS Asia-Pacific Workshop on Systems*, 2021, pp. 56–62.

[5] D. Kim, S. Lee, and K. Park, "A case for smartnic-accelerated private communication," in *4th Asia-Pacific Workshop on Networking*, 2020, pp. 30–35.

[6] Smartnic crypto analysis. [Online]. Available: https://github.com/PINetDalhousie/smarNIC-crypto

[7] Z. Jiang, H. Jin, G. E. Suh, and Z. Zhang, "Designing secure cryptographic accelerators with information flow enforcement: A case study on aes," in *Proceedings of the 56th Annual Design Automation Conference 2019*, 2019, pp. 1–6.

[8] M. S. Riazi, K. Laine, B. Pelton, and W. Dai, "Heax: An architecture for computing on encrypted data," in *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2020, pp. 1295–1309.

[9] [Online]. Available: https://developer.arm.com/documentation/ddi0596/2020-12/SIMD-FP-Instructions

[10] P. Longa, "Fourqneon: Faster elliptic curve scalar multiplications on arm processors," in *International Conference on Selected Areas in Cryptography*. Springer, 2016, pp. 501–519.

[11] Z. Liu, K. Järvinen, W. Liu, and H. Seo, "Multiprecision multiplication on armv8," in *2017 IEEE 24th Symposium on Computer Arithmetic (ARITH)*. IEEE, 2017, pp. 10–17.

[12] "Intel® advanced encryption standard (aes) new instructions set." [Online]. Available: https://www.intel.com/content/dam/doc/white-paper/advanced-encryption-standard-new-instructions-set-paper.pdf

[13] I. OpenSSL Foundation, "openssl-speed." [Online]. Available: https://www.openssl.org/docs/man1.1.1/man1/openssl-speed.html

[14] V. Krasnov, "How "expensive" is crypto anyway?" Aug 2018. [Online]. Available: https://blog.cloudflare.com/how-expensive-is-crypto-anyway/

[15] M. Dubrovsky, M. Ball, and B. Penkovsky, "Optical proof of work," *arXiv preprint arXiv:1911.05193*, 2019.

[16] "A64 cryptographic instructions." [Online]. Available: https://developer.arm.com/documentation/100076/0100/a64-instruction-set-reference/a64-cryptographic-algorithms/a64-cryptographic-instructions

[17] "Intel® sha extensions." [Online]. Available: https://www.intel.com/content/www/us/en/developer/articles/technical/intel-sha-extensions.html

[18] S. Gueron, S. Johnson, and J. Walker, "Sha-512/256," in *2011 Eighth International Conference on Information Technology: New Generations*, 2011, pp. 354–358.

[19] N. Heninger, "Rsa, dh, and dsa in the wild," *Cryptology ePrint Archive*, 2022.

[20] H. Tschofenig, M. Pegourie-Gonnard, and H. Vincent, "Performance of state-of-the-art cryptography on arm-based microprocessors," *NIST Lightweight Cryptography Workshop 2015*.

[21] M. T. Khan, J. DeBlasio, G. M. Voelker, A. C. Snoeren, C. Kanich, and N. Vallina-Rodriguez, "An empirical analysis of the commercial vpn ecosystem," in *Proceedings of the Internet Measurement Conference 2018*, 2018, pp. 443–456.

[22] W. J. Tolley, B. Kujath, M. T. Khan, N. Vallina-Rodriguez, and J. R. Crandall, "Blind in/on-path attacks and applications to vpns," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 3129–3146.

[23] "Overview of negotiating ikev2 security associations." [Online]. Available: https://www.ibm.com/docs/en/zos/2.2.0?topic=protocol-overview-negotiating-ikev2-security-associations

[24] "strongswan - about." [Online]. Available: https://www.strongswan.org/about.html

[25] R. Neugebauer, G. Antichi, J. F. Zazo, Y. Audzevich, S. López-Buedo, and A. W. Moore, "Understanding pcie performance for end host networking," in *Proceedings of the ACM SIGCOMM'18*. New York, NY, USA: Association for Computing Machinery, 2018, p. 327–341.

[26] A. Feraudo, P. Yadav, R. Mortier, P. Bellavista, and J. Crowcroft, "Sok: Beyond iot mud deployments–challenges and future directions," *arXiv preprint arXiv:2004.08003*, 2020.

[27] A. P. Felt, R. Barnes, A. King, C. Palmer, C. Bentzel, and P. Tabriz, "Measuring https adoption on the web," in *26th USENIX security symposium (USENIX security 17)*, 2017, pp. 1323–1338.

[28] B. Pismenny, H. Eran, A. Yehezkel, L. Liss, A. Morrison, and D. Tsafrir, "Autonomous nic offloads," in *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 2021, pp. 18–35.

[29] [Online]. Available: https://hg.nginx.org/nginx/rev/release-1.18.0

[30] A. Rawdat, "Testing the performance of nginx and nginx plus web servers," Aug 2021. [Online]. Available: https://www.nginx.com/blog/testing-the-performance-of-nginx-and-nginx-plus-web-servers/

[31] A. Ltd., "Trustzone for cortex-a – arm®." [Online]. Available: https://www.arm.com/technologies/trustzone-for-cortex-a

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

[33] G. P. Katsikas, T. Barbette, M. Chiesa, D. Kostić, and G. Q. Maguire, "What you need to know about (smart) network interface cards," in *International Conference on Passive and Active Network Measurement*. Springer, 2021, pp. 319–336.

[34] J. Liu, C. Maltzahn, C. Ulmer, and M. L. Curry, "Performance characteristics of the bluefield-2 smartnic," *arXiv preprint arXiv:2105.06619*, 2021.

[35] Y. Qiu, J. Xing, K.-F. Hsu, Q. Kang, M. Liu, S. Narayana, and A. Chen, "Automated smartnic offloading insights for network functions," in *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles*, 2021, pp. 772–787.

[36] J. Krude, J. Rüth, D. Schemmel, F. Rath, I.-H. Folbort, and K. Wehrle, "Determination of throughput guarantees for processor-based smartnics," in *Proceedings of the 17th International Conference on emerging Networking EXperiments and Technologies*, 2021, pp. 267–281.

[37] T. Xing, H. Tajbakhsh, I. Haque, M. Honda, and A. Barbalace, "Towards portable end-to-end network performance characterization of smartnics," in *Proceedings of the 13th ACM SIGOPS Asia-Pacific Workshop on Systems*, ser. APSys '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 46–52.

[38] F. Hauser, M. Häberle, M. Schmidt, and M. Menth, "P4-ipsec: Site-to-site and host-to-site vpn with ipsec in p4-based sdn," *IEEE Access*, vol. 8, pp. 139 567–139 586, 2020.

[39] X. Chen, "Implementing aes encryption on programmable switches via scrambled lookup tables," in *Proceedings of the Workshop on Secure Programmable Network Infrastructure*, 2020, pp. 8–14.

[40] S. Yoo and X. Chen, "Secure keyed hashing on programmable switches," in *Proceedings of the ACM SIGCOMM 2021 Workshop on Secure Programmable network INfrastructure*, 2021, pp. 16–22.

[41] D. Scholz, A. Oeldemann, F. Geyer, S. Gallenmüller, H. Stubbe, T. Wild, A. Herkersdorf, and G. Carle, "Cryptographic hashing in p4 data planes," in *2019 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*. IEEE, 2019, pp. 1–6.