PERFORMANCE ANALYSIS OF CRYPTOGRAPHIC FUNCTIONS ON PROGRAMMABLE NICS

by

Jack Zhao

Submitted in partial fulfillment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
August 2022

Table of Contents

List of Tables

List of Figures

Abstract

The development of programmable network interface cards (also known as SmartNICs) often come with multiple computing cores and multi-hundred Gbps bandwidth that can be used as an enhancement of network computing to extend the server CPU processing capacity. This trend inspired academics and industry to put more roles on the SmartNICs for applications offloading or acceleration that can traditionally only run on the servers (e.g., key-value stores or distributed transactions). However, there are no systematic studies on running network security applications on the SmartNIC, especially those commonly incorporated with heavy-loaded cryptographic operations. This thesis aims to fill the gap by providing the first in-depth analysis of the cryptography capabilities of the current SmartNICs. Our study shows that the SmartNICs' cryptographic performance is highly influenced by cryptographic instructions optimization, crypto-hardware acceleration, and other architectural enhancement. Moreover, data transmissions between SmartNICs and their onboard crypto-hardware accelerator can impact the overall cryptographic performance, especially for small-size short-living tasks. However, SmartNICs can take advantage of their deployment location, i.e., closer to client devices than server CPUs, to speed up crypto-based functions, especially for latency-critical applications. However, the SmartNIC benefits can be easily outweighed if the application is too data-intensive or includes several non-crypto tasks.

List of Abbreviations Used

AEAD	Authenticated Encryption with Associated Data
AES	Advanced Encryption Standard
AI	Artificial intelligence
API	Application Programming Interface
CBC	Cipher Block Chaining
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
DDR	Double Data Rate
DFS	Distributed File System
DMA	Direct Memory Access
DRAM	Dynamic Random Access Memory
DSA	Digital Signature Algorithm
DSP	Digital Signal Processor
ECC	Elliptic-Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
ECMP	Equal-Cost Multi-Path
FHE	Fully Homomorphic Encryption
FPGA	Field-Programmable Gate Array
GCM	Galois/Counter Mode
HPC	High-Performance Computing
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IKE	Internet Key Exchange
IoT	Internet of Things
IPsec	Internet Protocol Security
ISA	Instruction Set Architecture
MAC	Medium Access Control
MAC	Message Authentication Code

MPI	Message Passing Interface
NIC	Network Interface Card
NIST	National Institute of Standards and Technology
NPU	Network Processing Unit
PD	Protection Domain
PKA	Public Key Acceleration
QUIC	Quick UDP Internet Connection
RADIUS	Remote Authentication Dial-In User Service
RAM	Random Access Memory
RDMA	Remote Direct Memory Access
RNG	Random Number Generator
RPC	Remote Procedure Call
RSA	Rivest–Shamir–Adleman
RTT	Round Trip Time
SHA	Secure Hash Algorithm
SIMD	Single Instruction, Multiple Data
SoC	System-on-Chip
SRAM	Static Random Access Memory
TEE	Trusted Execution Environment
TLS	Transport Layer Security
TRNG	True Random Number Generator

Acknowledgements

First, I would like to extend my sincerest gratitude to my supervisor, Dr. Israat Haque, who has watched me tirelessly throughout my Master's degree. She has always been there to keep me on the straight path and nurtured my growth as a researcher. Her motivation kept me going through the hurdles of academia and research. It was an honor to have her as both a supervisor and a guardian throughout my time at Dalhousie.

Next, I would be remiss if I did not acknowledge the contribution of Dr. Miguel Neves, the postdoc fellow at the Programmable and Intelligent Network (PINet) research lab, Dalhousie. He has helped me tremendously throughout my research endeavors and was ever-present to extend a helping hand and ensure I remain motivated enough to stay the course. I also want to thank my colleagues for making our lab like home and my other friends for their friendly advice and support throughout my academic journey.

Lastly, I would love to thank my parents and my partner, without whom I would not be here. Their constant support and love are a debt I cannot repay, and it is one of life's greatest joys to make them proud of my efforts throughout this work and my Master's degree.

Chapter 1

Introduction

With the slowdown of CPU performance advances in the recent years, improving the production process and innovating the hardware architecture are incapable of catching up with what Moore's law predicts for the semiconductor industry, especially for data centers and organizations that require tremendous computing power. People have been continuously looking for other alternatives to help expand the application performance; the adoption of using programmable accelerators (e.g., GPUs, programmable SSDs, SmartNICs) have been stretching in different directions [1].

SmartNICs or smart network interface cards have been one trending solution researched and developed by prominent vendors (e.g., Nvidia, Broadcom, Netronome) to ease this problem. It is beneficial as the supporter of the host CPU and saves them some precious computing power; thus is commonly seen in the production environment, especially cloud environments. Those SmartNICs often pack more powerful and specialized computing resources than the traditional NICs, including multicore processors, onboard SRAM/DRAM, customized hardware accelerators for compression and crypto tasks, and programmable DMA engines [2]. These extensive features have attracted researchers to explore in different directions, especially applications offloading such as load balancing, key-value stores, distributed transactions, etc. [3] [4] [5] [6].

1.1 Motivation

Despite the SmartNICs' wide adoption and studies in different aspects, little is known about SmartNICs' capability to run cryptography-related function. On the one hand, cryptographic hardware accelerators are commonly seen on the SmartNICs; they could produce noticeable results in domain-specific tasks and rapidly process secure network packets. On the other hand, SmartNICs are not comparable with the server in terms of general computing power. Moreover, offloading some of the host tasks onto SmartNICs could increase the data transmission cost; utilizing SmartNIC cores

with other components, like hardware accelerators, could also increase overhead in practice.

State-of-the-art studies have shown some interest in offloading cryptography-related tasks on the SmartNICs; however, those studies only dive into a single problem or task without conducting the big pictures. For instance, Taranov et al. [7] have found that Broadcom Stingray SmartNICs could maintain high processing rates for message authentication ciphers (e.g., AES and SHA). Cui et al. [4] provide solutions to improve the TLS connection performance by offloading the handshake of TLS sessions to NVIDIA BlueField SmartNICs. In contrast, to solve the same problem, Kim et al. [8] chose to partially offload the TLS data encryption/decryption stage to Marvell LiquidIO III boards. Those are all inspiring studies; however, there is a noticeable missing piece in the current study on SmartNICs that needs to be fulfilled by systematically analyzing the SmartNICs' cryptography capabilities. Moreover, we are eager to answer two questions: "when one should offload cryptography tasks to the SmartNIC?" and "what kind of cryptography tasks should be offloaded?"

1.2 Contribution

This thesis provides the first in-depth analysis to fill the missing part of the systematic cryptography capabilities study of SmartNICs. A summary of the contribution and findings of this thesis is listed as follows:

- To meet the performance of cryptographic workloads on the server, present SmartNICs rely laboriously on architecture enhancements, e.g., cryptographic instructions and hardware accelerators. Specifically, thanks to specially dedicated instructions, ARM-based SoC SmartNICs could produce up to 25% better throughput than servers with cryptography hashes.

- The overheads of data transmission between SmartNIC's processor and onboard crypto-hardware accelerator could be the bottleneck of the cryptographic operations; even the cryptographic primitives (e.g., ECC-based digital signatures) have different sources of algorithmic optimizations. Our study shows up to 91% slowdown for ECDSA when using the onboard accelerator.

- Despite the cryptography acceleration mechanisms, current SmartNICs can still struggle with heavy data flow cryptography applications or the workload mixed with additional tasks. For specific applications, for example, client authentication and secure web servers, we can see roughly 56% and 73% worse performance on SmartNICs than the server CPU under high loads, even when the hardware can provide enough bandwidth.

- The deployed location of SmartNICs, i.e., close to client devices than server CPUs has a noticeable impact on the latency and can help accelerate crypto-intensive distributed applications. Our study suggests that there can be up to 50% deductions on latency for packets with smaller data sizes.

This study also comes with an online repository [9] with all the configurations we use to set up the experiments, running logs with results, and automated scripts to reproduce every step on different platforms. Content in the repository also provides further benchmarks, and due to the complexity, we did not show them in the thesis. Overall, we believe our findings positively affect the design and application offloading policies on the SmartNIC and could help make the initial move towards more in-depth studies on how to use SmartNIC to enhance the network security applications:

- For the SmartNIC's design, vendors could use our findings to improve the existing cryptography features and add support for potential features on the SmartNIC. It could also help implement hardware accelerators that could fit the long-term benefits.

- For application's offloading policies, our study could be used as guidance to help build offloading engines with more rational decisions. We also provide some lessons we learned and the future research directions that could extend from our current study.

1.3 Thesis Outline

The remainder of this thesis structure is organized as follows: Chapter 2 is divided into two sections, the first of which, Section 2.1, lays the foundational background required to help interpret the work undertaken in this thesis. Next, Section 2.2 presents some of

the research works relevant to our work. Chapter 3 aims to systematically evaluate the basic cryptography on the SmartNIC and compare it with the server's performance. Here, the overview of why there is a need to do the basic cryptography evaluation is explained in Section 3.1. This context is then followed by how we set up our evaluation testbed and the benchmarking tool in Section 3.2. Following these, Section 3.3 shows the three basic cryptography cipher categories we selected: symmetric ciphers, hash functions, and asymmetric ciphers, with their evaluation and analysis, as well as the takeaways we learned from those evaluations. Chapter 4 provides more detailed studies based on the real-world applications, with the justifications of how the applications are selected in Section 4.1. Following these, we have Sections 4.2, 4.3, and 4.4 for the three applications we selected: VPN tunneling, user authentication, and secure web serving; for each section, we have a comprehensive evaluation comparison and analysis accordingly. Chapter 5 concludes this thesis in Section 5.1 and discusses future research directions in Section 5.2.

Chapter 2

Background and Related Work

This chapter introduces the necessary background and knowledge required to comprehend the work presented in this thesis. Following that, we review related work that could help improve the understanding of the topic.

Network programmability includes various techniques (e.g., software-defined networking (SDN)) and hardware (e.g., SmartNIC, switch, or FPGA). It is possible to solve different networking problems using such techniques and hardware. For example, we can offer reliability, performance, and security in wired [10–20] and wireless [21–26] networks using SDN over programmable switches or NICs. However, in this thesis, our focus is on SmartNICs-based security services evaluation. Thus, we provide necessary background on SmartNICs, their capabilities along with cryptographic functions and their applications.

2.1 Background

2.1.1 SmartNIC Overview

SmartNIC is a programmable accelerator that provides many cryptograph-related hardware accelerations and other networking features. Every server needs a NIC to connect to the network; most of them are currently using 25 GbE and 50 GbE connections and are rapidly moving towards 100 GbE transmission technology. The SmartNIC is the NIC with programmable features and is widely used in the data

Table 2.1: Architectural specifications of popular commodity SmartNICs

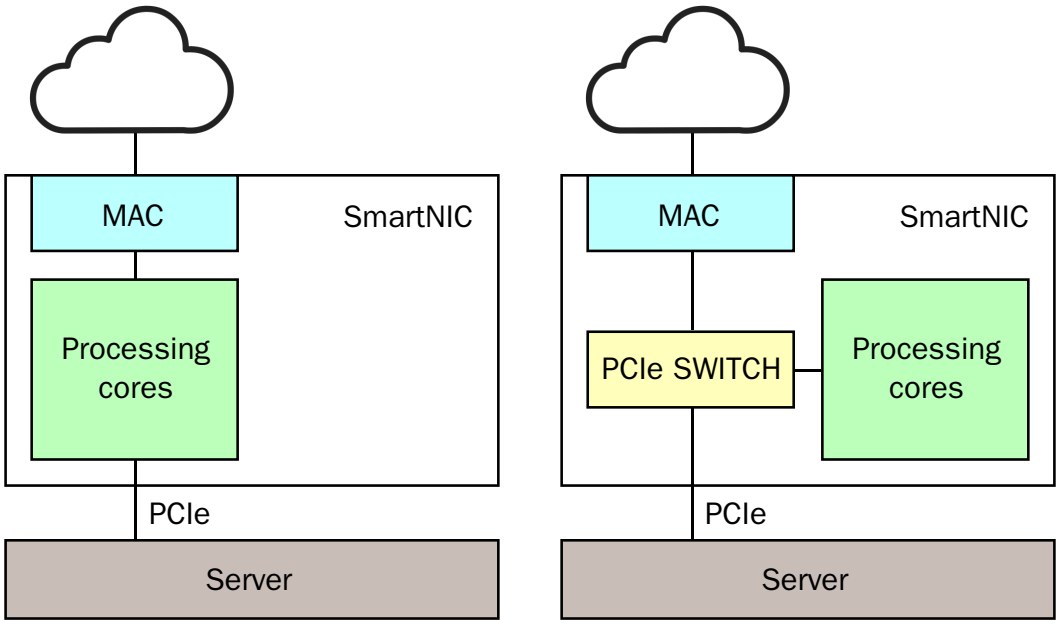
SmartNIC model	Vendor	SoC	CPU	FPGA	NPU	Processor	On/Off path
LiquidIO III CN96XX	Marvell	OCTEON TX2	✓			Arm v8.2, 36 cores, 2.4 GHz	On
Agilio LX	Netronome	NFP-6000			✓	Flow Processing Core (FPC), 120 cores, 1.2 GHz	On
BlueField 1M332A	NVIDIA	BlueField	✓			Arm Cortex-A72, 16 cores, 0.8 GHz	Off
Stingray 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

center nowadays and helps enhance the central server's networking, security, and storage efficiency at a relatively affordable price; many cloud service providers have already adopted it, and it definitely has more growth in the foreseeable future based on the prediction from the industry [27]. Most SmartNIC could provide these additional computing resources for the data center to use, and it is always used the open-source software to utilize, so the usabilities are flexible. Those additional computing resources have always been used to improve network package processing and help offload some lightweight tasks that traditional networking solutions can only run on the host CPU.

To have a more detailed view of the current SmartNIC market, we summarize the specification of six commercial SmartNICs from major vendors in Table 2.1. A typical SmartNIC often comes with onboard memory, accelerators (e.g., cryptography, compression, and regular expression matching), and multiple processing cores. SmartNIC's memory always comes with 32 KB in L1 cache, 1 to 16 MB for L2 cache, and in the range of 4 to 16 GB in the DRAM [28], our study omits the memory capability for simplicity. Those processing cores in the SmartNIC could be general-purpose CPUs, especially the ARM processor, which is the most commonly used due to its power efficiency nature.

There are also other special purpose processors like network processing units (NPUs), digital signal processors (DSPs), field-programmable gate arrays (FPGAs), or even processors specifically designed for artificial intelligence (AI) tasks [2] [29] [30]. Some SmartNICs with domain-specific processors can also let programmers utilize native hardware primitives (e.g., in Micro-C or VHDL) manually to manipulate data and achieve better processing speed. It is also common to see multi-purpose cores packed into one single SmartNIC product.

SmartNIC accelerators are one central selling point of the SmartNIC, and they can work really well for domain-specific processing, especially in cryptography-related domains; for example, a previous study shows that the MD5/AES engine can produce 7.0x/2.5x faster operations than the one on the host server (even using the Intel AES-NI instructions) [28]. Another advantage for the SmartNIC is that the position of the SmartNIC is one step closer to the client or other hosts compared to the host that attached the SmartNIC. In network applications, packet forwarding is not free.



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

Figure 2.1: SmartNIC Architecture Comparison

SmartNIC increases network package processing capabilities and speed; it also reduces network latency by having a shorter physical distance from the target. Moreover, we also know that SmartNICs can process the entire protocol stack if needed, based on one previous study [4].

Based on the network package processing or network traffic on the SmartNIC, the SmartNIC could also categorize based on how the SmartNIC processing cores interact with the traffic; there are two types: on-path SmartNIC and off-path SmartNIC.

Packets on the on-path SmartNIC go through the processing cores (Figure 2.1a). This means the processor on the on-path SmartNIC has to interact with all the packets received or transmitted by the host, which means it is more like the legacy NIC but adds more computing power in specific domains on top of it. However, since on-path SmartNIC needs to interact with all the packets, the hardware design needs to balance the efficiency and functionalities; moreover, in terms of the functionalities, improving the network-related functionalities' performance is its priority. LiquidIO and Agilio are examples of on-path SmartNICs.

For the off-path SmartNIC, the packets could be traveling around the processing cores using the NIC-level switching fabric by following the forwarding rules without

interacting with the processing cores (Figure 2.1b). That helps to let only the packets that need the extra computing power use the resources; furthermore, the hardware can have more room to put in specific types of processors for certain packets without worrying about the influence of the rest of the packets. Mellanox BlueField and Broadcom Stingray are examples of off-path SmartNICs.

It is also possible to find hybrid designs in which the SmartNIC contains both on-path and off-path modules; DSC and Alveo are examples of the hybrid design, where the user could choose the best way to process its packet based on their needs, however, the design and manufacture cost of the hardware is also higher.

2.1.2 Cryptography in Network Security

Cryptography is a broad area; our study only focuses on cryptography functions related to network applications. There are currently three main categories of cryptography functions that are commonly used in major network applications: symmetric ciphers, hash functions, and asymmetric ciphers.

Symmetric ciphers typically represent the cryptography that uses the same or the simple transformation of the same cryptographic keys for both encryption and decryption processes, thus symmetric. Therefore, in a network application, symmetric ciphers are generally used to encrypt the data from plaintext to ciphertext and vice versa. Moreover, modern symmetric ciphers are also designed with the authenticated encryption with associated data (AEAD) feature, which means that the data encryption process can simultaneously assure the confidentiality and authenticity of the data. Commonly used symmetric ciphers include different types of AES-CBC, AES-GCM, and ChaCha20-Poly1305, whereas AES-GCM and ChaCha20-Poly1305 are also AEAD ciphers [31].

The hash function is a cryptography operation that could map data with arbitrary size into fixed-size values using an irreversible operation. Furthermore, a good hash function in the security network applications always needs to satisfy two requirements: first, the hash function operation time should be as short as possible; secondly, the function should be one-way; thirdly, there should be as small duplications (or collisions) from the outputs as possible by using two different input values; finally, the result from the hash function should be able to pass a standard randomness test.

Commonly used hash functions in network security-related applications are SHA-256 and SHA-512 [32].

Asymmetric ciphers also represent the cryptography that uses a pair of related keys. One is used as the public key, which everyone could know, and another uses as the private key, which is only known by the owner that generated this key pair; thus, use asymmetric key pairs to do the data encryption and decryption. Asymmetric ciphers were widely used for data encryption and decryption like the symmetric ciphers; however, asymmetric ciphers are generally used exclusively for data authentication due to some of the following security risks: first, the cryptography risks (including all the state-of-the-art commonly used asymmetric ciphers could be mathematically vulnerable to the quantum computer with the help of Shor's algorithm [33] [34]), second, the risk that developers are often making mistakes when implementing asymmetric ciphers like RSA. Commonly used asymmetric ciphers are RSA, DSA, and ECDSA [31].

2.1.3 Why Offloading Crypto Operations?

Cryptography operations (e.g., hashing, encryption, decryption) are often considered computationally heavy on the server CPUs and use a considerable amount of resources. Semiconductor manufacturers (e.g., Intel, AMD) often design instructions (e.g., AES-NI) or add additional crypto-hardware accelerators based on the need to help improve the performance. However, over time, more computing tasks are added to the server (e.g., deep learning, video processing, and database monitoring) that compete with the already valuable computing power on the server; this has forced the community to explore alternative processing methods for additional workloads.

SmartNICs, on the other hand, have some advantages: First, there are a bunch of cryptography-related processing mechanisms (as we discuss in the next section), and the hardware performance is continuously growing. Second, SmartNICs are considerably cheap to install and upgrade compared to the server in terms of price; moreover, compared to the traditional NIC that requires close source vendor software, it is less complex to migrate to the SmartNIC and deploy existing cryptography functions. Third, most SmartNICs use more power-efficient processor architectures like ARM and FPGA, with colossal software optimization potential to have even better power

Table 2.2: Overview of crypto-acceleration features on commodity SmartNICs

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	○	○	●	●	●	●

● = Crypto-hardware acceleration, ◐ = Processor acceleration, ○ = No acceleration.

efficiency shown in many studies [35] [36] [37], along with the cryptography-related performance and deployment improvements, makes it more appealing to use as a supplement in the data center. Therefore, it is reasonable to offload some cryptography-related tasks on the SmartNIC and use it as a primary or additional source to release the server's pressure.

2.1.4 SmartNIC Cryptography Capabilities

Current SmartNICs support a wide range of cryptography accelerations, including different types of symmetric and asymmetric ciphers, hash functions, and random number generators, all of which play essential roles in network security. We have collected the cryptography acceleration capabilities for the top most popular SmartNICs on the market in Table 2.2. There are three categories of cryptography enhancement on the SmartNIC: the crypto-hardware acceleration, processor acceleration, and software optimizations [2].

Crypto-hardware acceleration. Most of the SmartNICs in our study support some form of crypto-hardware acceleration [2]. Crypto-hardware acceleration is the hardware implementation of certain cryptography operations; the acceleration is primarily achieved by reducing the computing cycles for specific cryptography algorithms on the hardware level or even making certain operations run in parallel. In general, crypto-hardware acceleration could provide an incredible performance increase compared to software optimization; it is also more power-efficient and has a relatively low manufacturing price than optimizing a general-purpose processor [38].

Our investigation demonstrated that most production SmartNIC crypto-hardware accelerators depend predominantly on ASIC design due to its lower price and mature

production line [2]. We also notice that FPGA-based accelerators on the SmartNIC are also trending because the programmable feature could provide more functionalities in the application and has been discussed in many other works [39] [40]. However, we still chose to reveal only the ASICs crypto-hardware accelerator on the SmartNIC because of its dominance in the market. The hardware integration of the ASICs on the SmartNIC is also flexible to produce; it could be designed and embedded on the SmartNIC by the manufacturer (e.g., NITROX V from Marvell or FlexSPARX from Broadcom) or installed separately as a third-party module (e.g., Rambus2 [41] modules in NVIDIA BlueField); furthermore, the communication between the CPU and the crypto-hardware accelerator on the SmartNIC is generally using the PCIe bus [2].

Processor acceleration. The processor acceleration is the collection of build-in cryptography features on the SmartNIC's CPU that come with part of the CPU design. It usually can be accessed by using the CPU's instruction on the ISA level. Unlike the crypto-hardware accelerator on the SmartNIC, which often needs extra driver support from the accelerator vendor, processor acceleration is more generic in software development because it uses the CPU instruction set directly, which should already be supported by the system [2].

Moreover, the program that is already written for the CPU hardware architecture on the SmartNIC can be ported to the SmartNIC smoothly without modifications. CPUs always have commonly used and well-established instructions built-in for those cryptography algorithms; for example, column mixes and hash updates operations in the AES and SHA are built-in instructions on the ARM cores [42]. Moreover, the instruction sets vary in different processor architecture; generally, the newer one could contain more instructions; for example, ARMv8.2 support SHA3 and SHA-512 instructions that are not available in the ARMv8 architecture [2].

Software optimizations. The cryptography performance can also improve by doing the software optimization on the SmartNIC. Software optimization could provide performance improvement even after the hardware is released; however, it takes more thought and effort for the developer and requires them to have a comprehensive knowledge of the hardware architecture and how the cryptography algorithms work. We notice that although many of the software optimizations are not targeted to the

SmartNIC specifically, those software optimization improvements can still provide obvious performance gains on the SmartNIC's cryptography functionality [2].

Those optimization improvements are mainly developed due to the popularity increase of the ARM architecture in recent years [43]; furthermore, as we mentioned previously, most SmartNICs have ARM cores build-in, so the SmartNICs have been blessed with that trend. To be more specific, these optimizations that are cryptography related including design better vectorized implementations [44] and optimizing arithmetic operations (e.g., multi-precision multiplication [45]).

In general, all three categories of cryptography enhancements are important for the SmartNIC. Crypto-hardware accelerator can always create better performance, but it is inflexible in future upgrades, especially since more and more new algorithms have developed and evolved over the years. Moreover, it needs more effort for the vendor to design and test the hardware and often requires specific driver release and maintenance from the vendor. Processor acceleration can be more recognizable, and more developers are working on it to provide meaningful optimizations and feedback. However, it is still part of the hardware design with less flexibility when upgraded. Finally, software optimization always provides the best adaptability when new algorithms and designs exist. Although it might have weaker performance than the hardware accelerator, it still plays a crucial role in enhancing the SmartNIC's cryptography capabilities. Thus, there is no winner in these three categories of cryptography enhancements. An extreme example is Apple's A11 chip which is packed with many custom hardware and processor accelerations; along with their own software optimization, they have created one of the best-in-class chips based on the ARM architecture, which requires a tremendous R&D and financial investment. Therefore, hardware designers and software developers could use one or multiple cryptography acceleration enhancements to achieve the best possible performance depending on the computation demand and the budget.

2.2 Related Works

2.2.1 SmartNIC as a Performance Accelerator

Some of the papers provide systematic evaluations on offloading distributed applications and microservices onto the SmartNIC. iPipe [28] provides a framework for Multicore SoC SmartNICs, which help build distributed applications. The study first analyzes the features of the SmartNICs and concludes with a few design ideas. Their analysis shows that the packet forwarding with different sizes needs different computing cycles; therefore, the offload needs to change the decision for different packet sizes adaptively.

Meanwhile, the hardware packet management can reduce the synchronization overhead efficiently by providing a shared queue abstraction. Also, the offloading framework needs to leverage the domain-specific hardware accelerator. However, it also needs to notice that the NIC core needs to wait for the execution completion, which creates extra latency. Another observation from their study is the performance loss when applications use memory beyond the L2 cache. Moreover, SmartNIC can gain performance from non-blocking direct memory access (DMA). Their iPipe framework design is based on the above findings and provides a programmable and efficient environment that could simultaneously run multiple applications.

E3 [3] focuses on offloading microservices (like the IoT hub or virtual network functions) on the SmartNIC in order to improve the energy efficiency and the latency of those microservices. The study first gives a general architecture analysis of the SmartNIC and finds that network switching is one of the bottlenecks for letting microservices run on the SmartNIC. In order to unleash the power of the SmartNICs in the data center and improve the energy efficiency and the latency of the microservices, the researchers use the equal-cost multipath (ECMP) load balancing to provide a high-performance host and SmartNIC communication. Besides, E3 also uses a communication-aware microservice placement algorithm to provide the SmartNIC overload reduction and migration of the microservice. This study did a great job of demonstrating the potential of using SmartNIC to accelerate the microservices; however, there is no in-depth studies or analysis on the security and cryptography features of the SmartNIC.

Other researchers study the on-path SmartNIC-accelerate solution for distributed transactions and introduce Xenic [6]. Their solution has the following features: first, Xenic supports a stateful offloading by its transaction commit protocol to avoid host Remote Procedure Call (RPC) overhead or limitations of one-sided Remote Direct Memory Access (RDMA). Xenic has a co-designed data structure to allow the data to be stored on both SmartNIC and Server. It also supports function shipping for the transaction logic offloading without worrying about the PCIe crossing. Xenic also implements multi-hop NIC to NIC communication to further improve the overall protocol efficiency. The result is that they double the delivery throughput compared to the state-of-the-art RDMA solutions by optimizing the on-path SmartNIC acceleration. While this work has not touched on any aspect of the security and cryptography acceleration of the SmartNIC, it helps us better understand the on-path SmartNIC acceleration solution.

Researchers also tried to reduce the barrier to offloading the existing applications onto the SmartNIC. NICA [46] creates a hardware-software co-designed framework for the inline application acceleration in packet filtering and packet transformation. The contribution is an inline accelerator that provides the operating system abstractions and virtualization (with state isolation and performance isolation support) support on the SmartNIC.

FLOEM [47] introduces a programming language with the compiler and runtime to provide programming abstraction for the SmartNIC-accelerated applications. Their study has three features: first, it uses the existing data-flow model to provide packet processing. Second, it achieves parallelization of the application by using the logical-to-physical queue mapping plus providing the per-packet state anywhere in the program and the caching between the server and the SmartNIC. Finally, it also supports porting the existing applications with minimum effort. Same as the previous studies mentioned above, these studies did mention the cryptography features on the SmartNIC; however, security and cryptography features on the SmartNIC are not their focus.

Some studies focus on the offloading in specific layers and protocols of the network. The previous study provides solutions and evaluations on the layer-5 protocols for autonomous NIC offloads [48]. Their paper talks about layer-5 protocols over TCP

acceleration in hardware and software. There are on-CPU and off-CPU options in hardware acceleration, and for the off-CPU options, they compare the NIC with other specific accelerators. Furthermore, they compare the existing dependent offloads method with the autonomous offloads they introduce on the NIC acceleration, which shows that the autonomous offloads can increase the performance per dollar. Their study also chose TLS offloading for their evaluation, which is identical to our study. However, their study focuses more on the performance aspect of the TLS offloading on the SmartNIC, not the security aspect, and they only select the AES-GCM cipher for the evaluation in their study.

Some other studies provide a TCP offload engine for the SmartNICs, like FlexTOE [49]. FlexTOE provides a few benefits compared to state-of-the-art solutions. First of all, it has removed all the TCP stack overhead via data-path offload to avoid control logic and cache on the NIC. Compared to the in-kernel and kernel-bypass solution, the *data-path offload* solution could save memory usage because it does not need to have the heavy instruction and instruction cache as well as reduce the overall CPU cycles. Secondly, it separates the TCP data path into modules to make it flexible and easy to modify for data center usage and future upgrade; this is a significant advantage for the data center to adopt the offload design compared to the previous inflexible offload methods. Finally, it uses all possible TCP process parallelization to boost the throughput of the overall design.

A study also focuses on analyzing QUIC protocol and methods to offload it onto the SmartNIC to improve the speed [50]. Unlike previous works, which only focus on the cryptography aspect of the QUIC protocol offload, they evaluate all aspects and separate them into four parts: cryptography, connection setup and tear-down, ACK and packet reordering processing, and packet I/O with header formatting. Their study shows two expensive parts in the overall QUIC protocol, *the kernel data copy action and the cryptography operations*. They then show three suggestions on how to improve the QUIC protocol speed.

The first one is to offload the AEAD cryptography operations onto the SmartNIC; the second is to reorder the decrypted packet to reduce the memory usage and improve the processing speed; Finally, hold the control operations on the server to prevent overstressing the SmartNIC. While this study mentioned the cryptography operations

inside the QUIC protocol, it focuses more on the overall performance of improving the QUIC protocol with the SmartNIC. This study helps us have a better understanding of how SmartNIC can improve the performance of UDP-like protocols.

There is also a study on analyzing the potentiality of offloading load balancers on the SmartNIC [4]. Their design includes the following features: a lightweight networking stack to provide a state machine, connection setup, and packet processing functionalities. A lightweight shared data synchronization supports both L4 (transport layer) and the L7 (application layer) load balancer. Moreover, the study also considers the hardware accelerator for the SmartNIC, including the packet rewrite engines and cryptography accelerator. Their study evaluated the load balancer hash table's performance and compared it with the server x86 architecture; in comparison, SmartNIC's actual performance is always higher than the generic performance in ratio. Sometimes, the performance could even overcome the server CPU. This evaluation result matches the founding of our study, and our study gives a further explanation of why this happens.

Another study LineFS [5] provides an implementation of offloading a distributed file system (DFS) on the SmartNIC. The current solutions for DFS require the DFS service and the actual applications running on the same host, which means they share the same CPU and memory resources. In this situation, the performance for both DFS and applications can be slowed down dramatically when running CPU-intensive tasks. Therefore, offloading DFS on the SmartNIC can solve this problem. However, there are two main challenges: first, the latency to offload tasks on the SmartNIC is relatively higher than the host's direct memory access; second, the SmartNIC has slower performance than the host server. They use two design principles to overcome this shortage. The first solution is to offload only the non-latency-critical task onto the SmartNIC; this could reduce the latency influence of the offloading process. The second solution is to use pipeline parallelism to improve the overall throughput of the DFS service. The result shows that the LineFS solution has a better throughput than the state-of-the-art solutions with lower latency.

Other than ARM-based SmartNIC, there are also studies on FPGA-based SmartNIC, and some of their research methodologies could also help our studies. hXDP [51]

tries to reduce the barrier to implementing software in an FPGA-based NIC by making it easy to use and using as few FPGA resources as possible. Their approach uses the eBPF infrastructure and recreates it onto the FPGA so that people can run Linux's eXpress Data Path (XDP) on FPGA. In order to achieve that, they created the hXDP, which includes the compiler to translate the XDP program, it has a module to extend ISA with some other optimizations, and it has the tool on the FPGA to communicate with XDP programs. They achieve a closer performance on average instructions per cycle than the x86 platform with low hardware resource use and better latency. This research helps us have a better understanding of the FPGA-based SmartNIC though it is not the focus of our research.

Other than utilizing only the SmartNIC to perform network acceleration, other studies try to create an in-network load balancing tool on both the programmable switch and the SmartNIC to improve the overall network application performance. Study [52] introduces an accelerator-aware in-network load balancer called P4Mite, which helps distribute the traffic among the server CPUs and the SmartNICs, or even other types of accelerators. There are two outstanding features P4Mite provides. First, P4Mite can work well in a heterogeneous system; it actively monitors and collects the load statics from all the devices it manages so that the appropriate amount of traffic can be delivered to the desired computing units accordingly based on their resources. Another feature is that P4Mite tries to improve the performance and utilize the switch's memory space by relying heavily on hashing and bit mapping. This study shows us the potential for network application performance improvement by designing the system, combing servers, SmartNICs, and other network devices.

2.2.2 In-network Cryptography

Some researchers are also studying the crypto-related aspect of SmartNIC or network devices. sRDMA [7] stands for secure Remote Direct Memory Access (RDMA). This study tries to build the enhancement of the SmartNIC's Remote Direct Memory Access with efficient authentication and encryption. sRDMA fills out the gap that there is no commonly used security protocol (like IPsec or TLS) support for InfiniBand Architecture (IBA). The sRDMA supports header and packet authentication, payload encryption, and memory protection. It can be backward compatible with the legacy

protocol and can be easily ported to new hardware.

It can prevent eavesdropping and man-in-the-middle attacks and the replay attack that the IPsec protocol cannot do by default. In addition to that, sRDMA integrates with the Protection Domain (PD) level keys so that it does not need to have a key overhead for each connection, thus reducing the memory overhead. It also provides extended memory protection to ensure the security of one-side communication. They also evaluate the latency and bandwidth performance for SHA-256, SHA-512, AES-128, and ChaCha20-Poly1305 in sRDMA.

Their latency evaluation is divided into header authentication only and full packet security. One finding from the experiment is that comparing those two results, they found that payload authentication is more expensive than header authentication. Moreover, AEAD inside the full packet security shows a faster latency increase when payload size increases compared to other choices because AEAD not only provides authentication but also needs more data encryption/decryption when payload size gets larger. They also evaluate the write and read bandwidth for the header authentication, packet authentication, and AEAD. The result shows that all cipher-based algorithms could achieve line rate in bandwidth by using more cores for header and packet authentication (besides SHA-512 for the packet authentication) with the best performance from ChaCha20-Ploy1305, and AES has the best performance for AEAD. This study has a detailed analysis of some of the SmartNIC's cryptography features; however, it can only apply to the RDMA-supported SmartNICs.

Moreover, the previous study already tried to analyze and improve a single security-related protocol, like TLS, on the SmartNIC [8]. Their study focuses on improving the overall throughput of the TCP connection setups by offloading the TLS handshake onto the SmartNIC and the other remaining operations on the Server CPU. The idea is based on the presumption that a modern x86 CPU (for example, with the help of AES-NI) can have better performance for the TLS data packets encryption and decryption than the TLS handshake, which is mostly about public-key cryptography operations.

Moreover, their study reduced the TLS handshake managing complexity by dividing the TCP connection into two stages: the first step is to have a stateless TCP connection setup with an SYN cookie solution. Secondly, provide a client-side API

to handle the key exchange phase packet loss and retransmission. Furthermore, to stop the SmartNIC from overloading, their setup stops TLS handshake offload on SmartNIC with a high load. Their evaluation shows a promising 5.9x throughput improvement compared to a single CPU core. While this study compares some of the AES and RSA ciphers in the TLS, our study further illustrates more general and thorough results in basic cryptography and security-related applications.

There is also a study about cryptographic hashing on the P4 data plane in SmartNIC and FPGA devices [53]. This study proposes the implementation of cryptographic hash functions in P4 platforms to solve the issue that the current P4 platform does not have this type of support. Their prototype runs on three types of hardware, CPU (t4p4s), NPU (from a SmartNIC), and FPGA. Their result shows that the CPU solution is highly extensible with more library support but can create more latency than others; the SmartNIC-based solution can have higher throughput on smaller message sizes, but not for the larger ones. Moreover, the FPGA solution has more potential in terms of low latency, but it lacks library support in the current stage. This study only explores the cryptography hashing performance for the domain-specific language P4 targeted devices.

Besides the in-network cryptography studies on the SmartNIC, other researchers also implement some of the commonly used functions onto the programmable switch and boost up and analyze the performance of those implementations.

One study presents P4-EncKV, which uses the programmable switch to accelerate the encrypted data store [54]. Their study designed an in-network proxy for encrypted key-value stores that improve the query speed and saves bandwidth while maintaining the existing security features. P4-EncKV tries storing different length values into a set of register arrays and reconstructing the value later using the match-action table. Right now, they demonstrate their proof-of-concept design with reduced about 20-25% latency. This study does not directly improve the cryptography algorithms; instead, they tried to improve the network security operation on the network device, which gives us a different way of thinking about improving the secure network application performance.

P4-IPsec [55] implement IPsec in P4-Based Software-defined networks. IPsec is a complex and widely used protocol, the complexity of IPsec protocol can increase

even more in dynamic and multi-peers setup, like most enterprise and extensive organization networks will face. Implementing IPsec on a P4-based SDN can reduce its complexity and improve scalability. Their design tries to be as minimalist as possible, and there are some worth noticing features.

Most IPsec setup uses Internet Key Exchange (IKE) protocol to authenticate both peers and set up a secure channel for key exchange. It also helps negotiates security associations like cryptographic algorithms, encryption keys, and other information necessary for secure communication. In contrast, the first thing P4-IPsec tries to simplify the design is to use an IKE-less implementation, which avoids IKE message exchange and uses the SDN controller as a substitution to set up and renew the unidirectional IPsec tunnel. Secondly, they use the P4 externs to implement the P4-IPsec's cipher suites to reduce the latency created by loading it onto an external process. The result shows that deploying P4-IPsec has a negligible impact on the goodput of the P4 target data plane, and the latency impact on the control plane is also relatively small. This study proves the concept of implementing IPsec on the P4 programmable data plane; however, missing features on the hardware, especially cryptography support, is still the bottleneck for current P4 switch hardware.

One study implemented a pre-computed scrambled lookup table with AES encryption on the programmable switch [56]. This study was inspired by the previous AES optimization studies on embedded devices. Moreover, the researcher noticed that programmable switches and other embedded devices all have similar memory and computing constraints; however, compared to the embedded devices, the programmable switch has a relatively larger memory which is possible to store the lookup table. After using the scrambled lookup table, the operation for AES can be reduced to two stages per round in theory, with one lookup and one XOR operation. In practice, their work only occupied less than 15% SRAM on the Tofino v1 and negligible utilization of other resources. However, the performance and power consumption are not ideal due to the lack of AES hardware accelerator on a programmable switch; it could not even compete with cheaper x86 laptops with AES-NI hardware accelerator.

SipID [57] is an implementation of HalfSipHash on the programmable switch. Hash functions are commonly used in data plane application development, like, indexing, fingerprinting, or sampling; however, developers are forced to use CRC32 hash

in most cases due to hardware limitations. On the other hand, there are secure hash functions, like SHA256, but they are computationally expensive and complicated to implement on a programmable switch. Unlike those mentioned above, HalfSipHash balanced the computation complexity with the security.

SipID improves the performance in two directions, first, reducing the arithmetic operations to only slicing bits and copying the value to a new variable; second, reducing the pipeline stages to only four per SipRound. Their result shows that SipID can handle more than 300 million hashes per second for 8-bytes input, and it is more than enough to replace the current CRC32 hash function. The study also evaluates combined ingress and egress pipelines to reduce overhead; the maximum hash rate for the ingress and egress design can be three times faster than the ingress-only design. The studies of cryptography implementations for programmable switches showed us the probability of doing more secure cryptography operations in-network. However, most of the results are not compatible even with the consumer-level devices, needless to say, the data center level's server. The main reason is the lack of cryptography hardware accelerators on the programmable switch. On the other hand, the crypto-related hardware accelerators are commonly installed on the SmartNIC, making it have better potential than the programmable switch.

2.2.3 Cryptography performance.

We also checked out other studies on cryptography performance evaluation. There are cryptography performance evaluations on different types of hardware.

First, we take a look at cryptography performance analysis on IoT devices. This study has a detailed analysis of the crypto-hardware in low-power and resource-constrained IoT devices [58]. They evaluate the cryptography support in the operating system as well as the cryptography hardware and software performance. To be more specific, their system can split into three levels of cryptography acceleration: full hardware, partial hardware, and purely software acceleration. Furthermore, they have tested their design on cipher hardware and external accelerators.

Their evaluation shows that hardware accelerators can significantly improve the cryptography process. For short inputs, the ciphers with hardware accelerators can gain 4 to 6 times of improvement, and the hashes can gain 2 to 3 times of progress.

However, the hardware control overhead can have a significant influence on short inputs. Whereas long inputs can improve, with 10 to 30 times gain on ciphers and 5 to 10 times earn on hashes; moreover, the external hardware accelerator creates much overhead. The external hardware accelerator consumes more energy than both on-chip hardware and software implementation, which is about 13 to 25 times more.

The other study introduces a secure Message Passing Interface (MPI) library for High-Performance Computing (HPC) applications in data center and cloud services [59]. Their work is based on OpenSSL, BoringSSL, Libsodium, and CryptoPP; they first evaluate these four libraries and finds that all libraries have different encryption overheads.

The study did the benchmarks in the following aspect. The first aspect is encryption-decryption; this is the preliminary evaluation of the cryptography performance of each library with different message sizes. The second is ping-pong, which measures the throughput when two sides communicate with each other back and forth. The third is the OSU benchmark, a micro-benchmark to test multiple network features, including bidirectional bandwidth, multi-threaded latency, and more; their study uses the OSU test suite to measure the throughput when multiple senders and receivers are in uni-direction. Finally, they considered NAS parallel benchmarks, the benchmark for evaluating the performance of highly parallel supercomputers (at least 1000 processors); this study used this benchmark specifically for their network cryptography performance. After that, they use the Hockney and max-rate models to model MPI communication, two commonly used communication performance models to help develop and evaluate parallel algorithms in high-performance computing. This study uses those two models to accurately predict the encrypted model performance so that other people can estimate the performance of the cryptography process without actually testing it.

Those two studies related to the cryptography performance evaluation above help us understand how researchers select and set up experiments on other types of network devices. Moreover, those studies provide background knowledge for us when we select the suitable cryptography library and measurement matrics for our case studies.

To summarize, Table 2.3 lists all related works and their contribution compared to ours. Many works focus on offloading different types of applications on the SmartNIC;

Table 2.3: Related work studies comparison

	Symmetric ciphers analysis	**Hash functions analysis**	**Asymmetric ciphers analysis**	**Cryptography and security related web applications' performance evaluation comparison**	**SmartNIC architecture (on-path / off-path) analysis**	**General web services acceleration or offload analysis**
Our Study [2]	●	●	●	●	●	●
iPipe [28]	○	○	○	◐	●	●
E3 [3]	○	○	○	○	●	●
Xenic [6]	○	○	○	○	●	●
NICA [46]	○	○	○	○	●	●
FLOEM [47]	◐	○	○	○	○	●
Pismenny et al. [48]	○	○	○	◐	●	●
FlexTOE [49]	○	○	○	○	●	●
Yang et al. [50]	○	○	○	◐	○	●
Cui et al. [4]	◐	◐	○	○	○	●
LineFS [5]	○	○	○	○	●	●
hXDP [51]	○	○	○	○	●	●
Tajbakhsh et al. [52]	○	○	○	○	◐	●
sRDMA [7]	◐	◐	○	○	◐	○
Kim et al. [8]	◐	○	◐	◐	○	●
Scholz et al. [53]	○	◐	○	○	○	○
Kuzniar et al. [54]	○	○	○	◐	○	●
P4-IPsec [55]	○	○	○	◐	○	○
Chen [56]	○	○	○	◐	○	○
Yoo et al. [57]	○	◐	○	○	○	○
Kietzmann et al. [58]	●	●	●	○	○	○
Naser et al. [59]	○	○	○	◐	○	○

● = **Well-addressed** in the analysis, ◐ = **Mentioned** in the analysis, ○ = **Not mentioned** in the analysis.

few need cryptography features on the SmartNIC, so they have some analysis; however, there is no state-of-the-art study for a systematic evaluation of the SmartNIC's cryptography capabilities. Other researchers study the cryptography capabilities of the other types of network devices, like the programmable switch and IoT device, and deploy security features on those devices. We believe that SmartNIC also has the potential to embed and enhance the cryptography and security capabilities of the network by leveraging its growing computing resources.

Chapter 3

Basic Cryptography Performance on SmartNIC: Design and Evaluation

3.1 Overview

The state-of-the-art research does not systematically address the SmartNIC's cryptography; moreover, most SmartNIC manufacturers only list all the cryptography accelerator features without their actual performance benchmark. Digging into the essence of the security application can help us have a more precise understanding of the performance of the SmartNIC. Thus, our study focused on evaluating the basic cryptography performance first on the SmartNIC before we start doing more studies on complex applications. Furthermore, we want to compare the performance of cryptography operations on the SmartNIC with the server to see the performance gap between both types of hardware. We also want to know what SmartNIC's computing power and other hardware features could we utilize to help enhance the overall security operations of a network infrastructure.

3.2 Evaluation Setup

Our basic cryptography evaluations work on a single server and single SmartNIC. Our experiments on the single server containing a 10-core Intel Xeon Silver 4210R 2.40 GHz CPU with 32 GB of DDR4 DRAM. The server's operating system runs Ubuntu 20.04.1 LTS with the Linux kernel version 5.4.0; additionally, the hardware supports AES-NI instructions [60]. For the attachment of the server, it is equipped with a two-port 25 GbE Mellanox BlueField SmartNIC containing a 16-core ARMv8 Cortex-A72 0.8 GHz processor and 16 GB of DDR4 DRAM. The SmartNIC also runs a modified version of Ubuntu 20.04 LTS with a custom build Linux kernel provided by Mellanox (version 5.4.44-mlnx.14.gd7fb187) [61].

The tool we are using for the cryptography algorithms' performance benchmarking

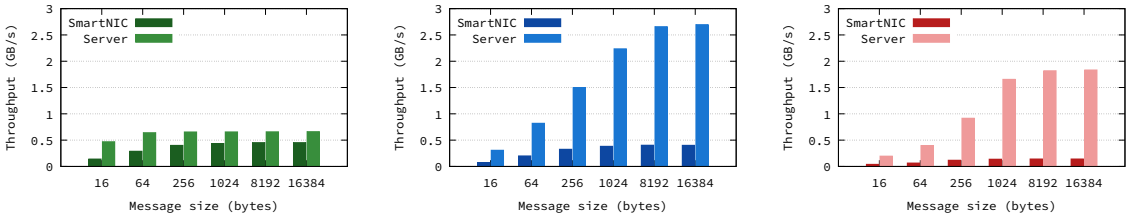
on the SmartNIC is OpenSSL Speed (version 1.1.1f) [62]; it is benchmarking tool integrated into OpenSSL. OpenSSL is one of the most widely used cryptography libraries, and it is open-source. It features a variety of implementations of basic cryptography functions, and it is available on almost all modern operating systems (Linux, macOS, and Windows) with active development. Our experiments test out the performance of different cryptographic algorithms on both the server and the SmartNIC with three classes of commonly used algorithms: symmetric ciphers, hash functions, and asymmetric ciphers. We then select the representative of each class of algorithms to evaluate its performance; thus, we could compare the results of different classes according to the main algorithm's perspective.

To be more specific. For the symmetric ciphers, we evaluate how much the data size that algorithms could perform encryption and decryption per second, and we have done our evaluation with different message sizes. We have similar setups for hash functions as the symmetric ciphers; we evaluate the data size of the hash that could process per second, and we have tested each evaluation with the hash on different sizes of the message. For asymmetric ciphers, our study evaluates the sign and verify speed for different types of asymmetric ciphers with different key lengths. To simplify the problem and eliminate the influence of multi-threading on cryptography operation's performance, each experiment in our study runs on a single thread. We also make sure to reset the operating system environment the benchmark tool is running to eradicate the influence of other factors. Moreover, to make the evaluation more accurate, the results are calculated from an average of 10 repetitions.

3.3 Results and Discussion

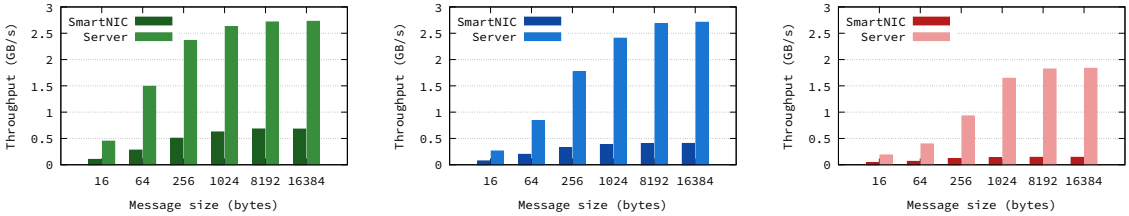
3.3.1 Symmetric ciphers

Our first evaluation is for the symmetric ciphers on different hardware. Our study selected the three most commonly used symmetric ciphers: AES-256-GCM, AES-256-CBC, and Chacha20-Poly1305. AES is commonly seen in the personal computer or more powerful hardware, such as the server. The reason is that most modern hardware nowadays has the hardware accelerator for AES, which could dramatically increase the speed of doing AES operations. To be more specific, AES-GCM is yet



(a) AES-256-CBC (b) AES-256-GCM (c) ChaCha20-Poly1305

Figure 3.1: Symmetric Cipher Throughput Comparison (Encryption)



(a) AES-256-CBC (b) AES-256-GCM (c) ChaCha20-Poly1305

Figure 3.2: Symmetric Cipher Throughput Comparison (Decryption)

the most common cipher for the TLS suite [63] [64], which people use that daily when they surf the web or use the web application. It is widely adopted because of the increasing popularity of AEAD (Authenticated Encryption with Associated Data). On the other hand, the usage of AES-CBC is fading out due to its security concerns and the implementation difficulty with MAC-then-Encrypt mode; however, it still has a notable amount of market share in web applications [65].

ChaCha20-Poly1305 is also an AEAD algorithm, and it is more lightweight than AES-GCM in a fair comparison; however, AES has a much cheaper hardware acceleration solution to boost the performance on most modern servers and personal computers. That being said, ChaCha20-Poly1305 still stands out with its lower battery usage than AES-GCM. Since mobile and IoT devices have limited space to add hardware accelerators and desire to increase the battery life, ChaCha20-Poly1305 could benefit those limited resources hardware [66]. As part of the AEAD's feature, both AES-GCM and ChaCha20-Poly1305 can provide authenticated encryption, and they also both work at the 256-bit security level [2]. Symmetric ciphers are mainly used for data encryption and decryption; therefore, we did our evaluation for both accordingly.

Data encryption. Our study analyzes the symmetric cipher's data encryption

throughput (in GB/s) comparison with different message sizes, and all the results are plotted in Figure 3.1. Our results indicate that for data encryption with symmetric ciphers, the server's performance is always better than the SmartNIC; this behavior is expected because the server's processor has higher performance on paper than the SmartNIC's. Moreover, the lack of hardware acceleration support for symmetric ciphers on the SmartNIC could also be a critical factor influencing the result. One study by Cui et al. [4] uses LiquidIO that support hardware acceleration for symmetric ciphers, and the SmartNIC's performance for AES algorithm is more than 2x faster compared to the server when processing larger packets (1024B or larger). Thus, it indicates that the SmartNIC with hardware accelerator can outperformance the server in some scenarios.

Our result shows that when we compare different cipher types on the server, the throughput for AES-GCM and ChaCha-Poly1305 can significantly (up to 5x) exceed AES-CBC. This result is mainly because AES encryption with CBC mode is based on a serialized process, which means that each block that needs to be encrypted is processed one after another in a sequence. On the processor level, when doing the instruction pipelining, the pipeline needs to stall and wait for the block encryption complete [8]. Our result also indicates that AES-GCM is faster than ChaCha-Poly1305 because the former cipher can utilize the processor-level cryptography extensions (e.g., AES-NI instructions) [2].

On the other hand, SmartNIC has different trends in terms of the encryption performance using AES-CBC cipher, and we believe there are two reasons for that. One reason is that the instruction parallelism works much better on x86 than the SmartNIC's processor (i.e., ARMv8), which could have a noticeable impact on the stream ciphers. Specifically, Intel AVX-512 instructions [67] could provide SIMD (Single Instruction, Multiple Data) support up to 512-bit, which indicates how many multiple data points it could process simultaneously; in contrast, ARMv8 NEON [68] allows at most 128-bit parallelism. The other reason is that AES-CBC does not have integrity protection build-in, which is one of the default features for AEAD algorithms like AES-GCM and ChaCha-Poly1305. Therefore, AES-CBC is a missing separate message authentication step that uses hash on the algorithm's output and thus reduces the algorithm's overall encryption time.

Data decryption. Our measurement of the symmetric cipher's data decryption throughput (in GB/s) comparison with different message sizes is present in Figure 3.2. Our result shows the same trends from most algorithms' performance compared to their encryption counter results; however, the performance of AES-CBC gives different results, with respectively up to 50% and 312% higher throughput on the server and SmartNIC [2]. Compared to AES-CBC encryption, the decryption process could effectively operate parallel, which makes a noticeable improvement with the help of pipelining on both server and SmartNIC, and also further helps the x86 to take advantage of their architecture enhancements [69].

Takeaways

- For symmetric encryption/decryption algorithms with sequential operations (e.g., AES-CBC encryption), the multicore SoC SmartNICs can perform more closely than the server.

- On the other hand, for other types of symmetric algorithms that could benefit from the processor's parallelism, the task offloading is not worth it; the server's CPU could easily outperform the SmartNIC's CPU.

3.3.2 Hash functions

Next, our study focuses on the evaluation of cryptographic hash functions. In this work, we have selected two of the most widely used cryptographic hash functions: SHA-256 and SHA-512 [64] to have their performance comparison. Cryptographic hash functions are commonly seen and fundamental for modern cryptography; some applications include data integrity verification or message authentication, quick and efficient data lookup with the hashtable, and even the proof-of-work in digital currencies (e.g., Bitcoin) [70].

Hash functions throughput (in GB/s) comparison result for SHA-256 is present in Figure 3.3. Our result shows that the throughput increases when the message size gets larger; this behavior is expected since hashing larger messages in a stress test means that there are fewer system interactions. Moreover, our study suggests that hash functions are relatively efficient in modern hardware, and the interaction with

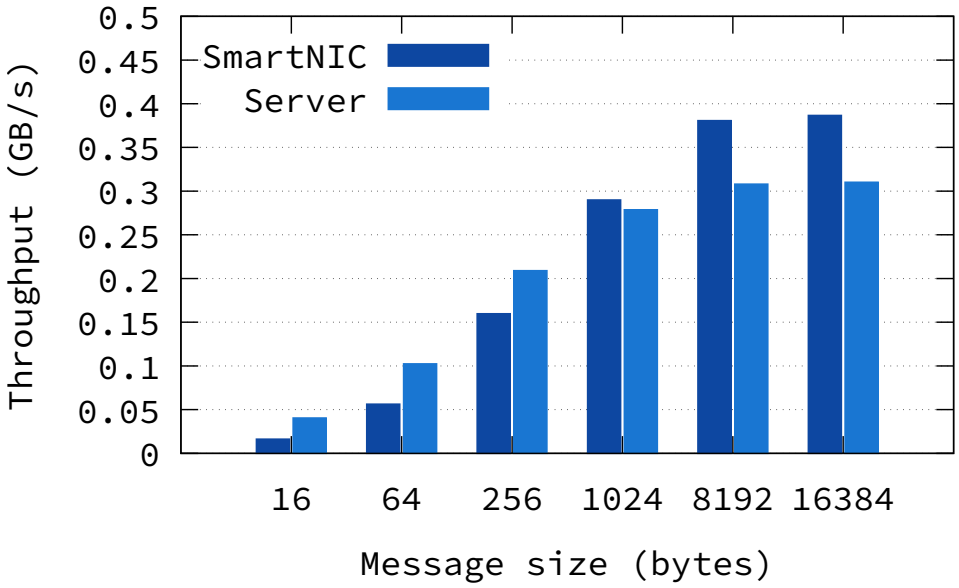


Figure 3.3: SHA-256 Hash Algorithm Throughput Comparison

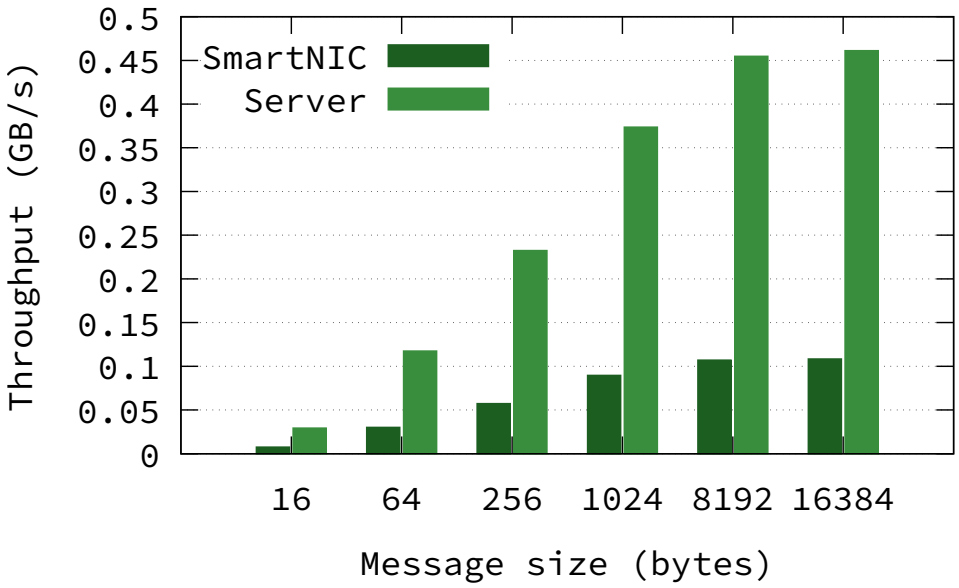


Figure 3.4: SHA-512 Hash Algorithm Throughput Comparison

the system (e.g., memory allocations) can have a noticeable time cost, thus leading to the figure's result [2].

One interesting trend in our result is that for messages greater than 1 KB, the SmartNIC's throughput could surpass the server. The hash computations acceleration on the ARMv8 processors, one of the cryptographic instructions, is the main

contribution to this result [71]. However, this advantage does not appear on the Intel Cascade Lake CPU used in our test. Our study found that the successor versions of Intel Xeon processors (e.g., Ice Lake and Rocket Lake architectures) have also included the support for SHA instructions [72], which in theory could improve the throughput.

Figure 3.4 shows the throughput (in GB/s) comparison result for SHA-512. The result for SHA-512 does not have the same trend as SHA-256; performance on the server is substantially higher than the SmartNIC for all message sizes. This difference is mainly influenced by the lack of dedicated instructions on ARMv8 processors for doing the SHA-512 operations; furthermore, the instruction that supports SHA-512 operations has been introduced to ARM instruction sets starting from the later ARMv8.2 [71].

Additionally, our result shows that the performance of SHA-512 on the server can be up to 49% higher for 16 KB messages compared to SHA-256, which shows a widely noticed trend on the 64-bit x86 machines [73]. The main reason is that on 64-bit arithmetic units, the SHA-512 algorithm could be processed with fewer round operations per byte when accomplishing a continuous input. More specifically, our study addresses that there are 80 rounds over 128-byte blocks for the SHA-512 operations against 64 rounds over 64-byte blocks for the SHA-256 operations when adopting 64-bit arithmetic [2]. Eventually, that summarizes into fewer instructions execution for the same amount of data input; thus, SHA-512 shows better performance on the x86 machines.

Takeaways

- To achieve a better crypto-hashing performance, modern SmartNICs depend extensively on architecture enhancements or processor cryptography accelerations, like cryptography instructions. Therefore, when designing the task offloading for secure hash-related programs, the SmartNIC's architecture needs to be considered, as well as the running state and workload characteristics.

- On the other hand, crypto-hash is a relatively lightweight yet essential cryptography operation; the energy cost per operation should also be an important factor when performing the task offload. SmartNICs are generally designed to

be more power-efficient compared to the server.

3.3.3 Asymmetric ciphers

Another evaluation in our study is to compare the server's and SmartNIC's performance when using various asymmetric ciphers, sometimes known as public-key primitives. The name for asymmetric cipher comes from the asymmetric features in the cryptographic system. Asymmetric ciphers commonly use a public/private key pair with a one-way function, where the data processing with the one-way function by one key can only be retrieved by using the one-way function with another key. Notice that although, in theory, those functions used by asymmetric ciphers should be one-way functions, in practice, we usually use theoretically two-way functions but are extremely hard to compute in one-way. Message authentication (or digital signature) and key exchange tasks are the most common applications for asymmetric ciphers. Our detailed explanation about all ciphers we used with some of their features addressing below, and we first start our investigations with message authentication.

Table 3.1: Asymmetric Cipher (ECDSA) Throughput Comparison (Sign)

Curve	**SmartNIC**	**SmartNIC (PKA)**	**Server**
P256	4884.945	452.61	32165.32
P384	117.775	220.835	717.455
P521	45.185	109.67	2333.695

Table 3.2: Asymmetric Cipher (ECDSA) Throughput Comparison Table (Verify)

Curve	**SmartNIC**	**SmartNIC (PKA)**	**Server**
P256	1517.73	217.845	10499.085
P384	162.695	103.575	942.01
P521	64.885	51.48	1180.115

Message authentication. To perform authentication with the asymmetric cipher, a sender first creates a digital signature by combining the message with a private key, which is then verified by the receiver using the sender's public key. RSA, prime

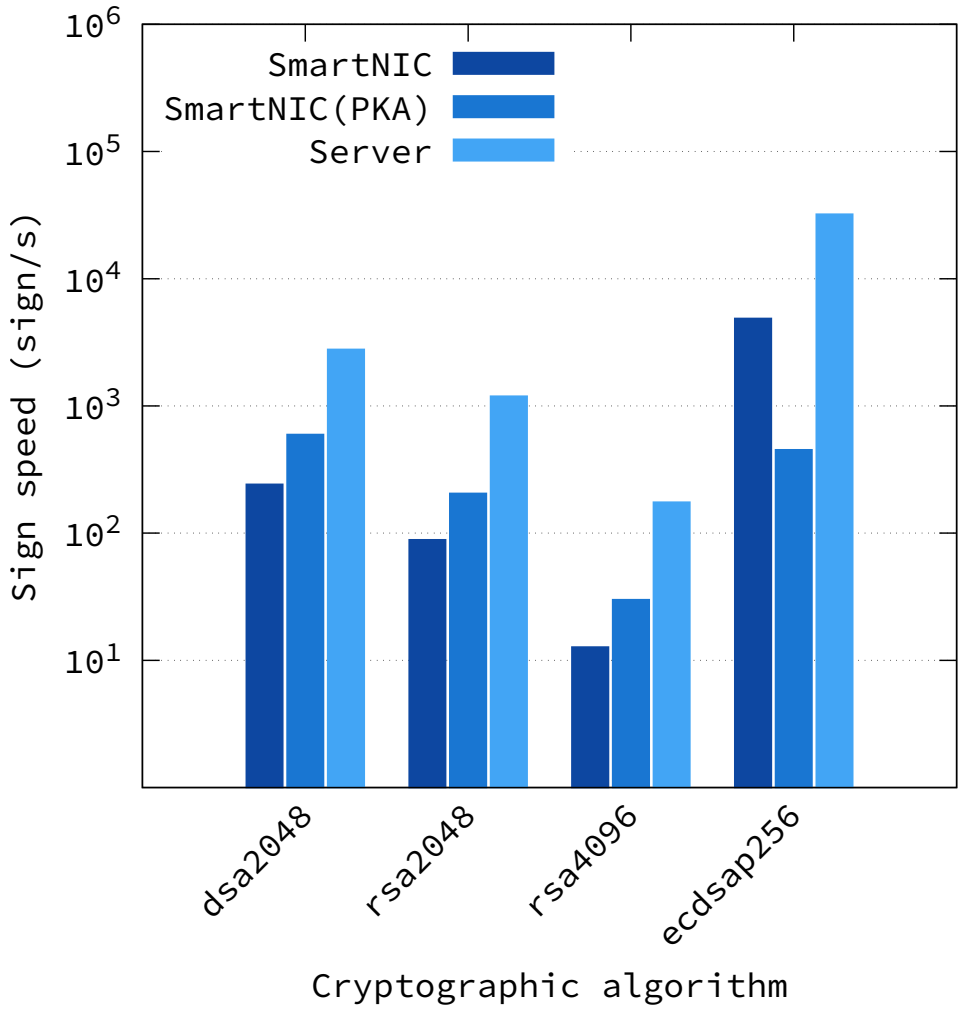


Figure 3.5: Asymmetric Ciphers Throughput Comparison (Sign)

field DSA, and elliptic-curve DSA (ECDSA) are the most extensively used digital signature algorithms today, and those algorithms are all widely embraced in Internet protocols, like TLS and SSH [74] [75]. The security of RSA is based on the factoring problem, where it is practically challenging to factorize the given number into two large prime numbers. While the security of (EC)DSA is based on the discrete logarithm problem, currently, there are no efficient methods to solve those problems. Our experiments evaluate RSA and DSA with key sizes above 2048 bits as those are the minimal sizes presently advocated by the NIST [76]. Notice that since ECC ciphers require smaller key sizes to achieve the same level of security as RSA, our test uses 256-bit keys for a fair comparison for ECC ciphers.

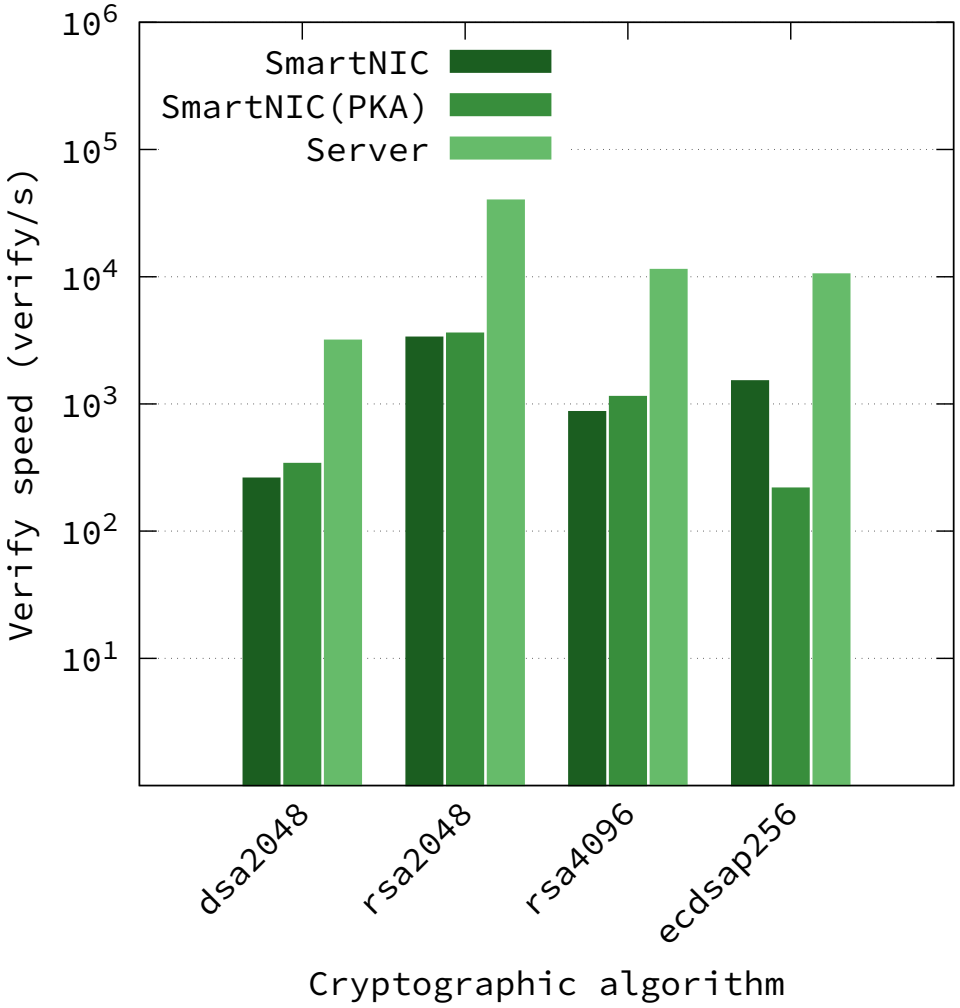


Figure 3.6: Asymmetric Ciphers Throughput Comparison (Verify)

Figure 3.5 shows the signing throughput for three authentication algorithms setup in our analysis for both the server and SmartNIC. Notice that PKA stands for Public Key Acceleration, the software package provided by the manufacturer of our SmartNIC to utilize the hardware public key accelerator; on the figure, PKA indicates that the hardware accelerator is turned on [2]. Unlike past research, e.g., [8], which utilizes a less powerful server in their study, our server result shows a significant performance advantage in all cases we investigated (more than 10x better depending on the scenario). As a result, even in the presence of crypto-hardware accelerators on the SmartNIC, the server layout (e.g., CPU and memory architecture) is still a crucial factor influencing whether or not to offload public-key operations to SmartNICs.

One thing worth noticing is that the performance of using CPU-based processing for ECDSA is better than using a crypto-hardware accelerator on the SmartNIC. Our study found two factors that could influence this result: firstly, there are data transfer costs when processing tasks using the hardware accelerator; this overhead in transferring and synchronizing the data could become the bottleneck of the overall performance; secondly, recent ARM processor core optimizations for supporting elliptic curves (especially NIST P-256 curves) have made their cores considerably faster [77]. Therefore, to better understand this result, we did more tests and created Table 3.1 to compare message signing performance with various elliptic curves on the server and SmartNIC. From the result, we can see that, for curves with bigger key sizes (e.g., up to 2.4x for the NIST P-521 curve), utilizing the crypto-hardware accelerator can still outperform the CPU on the SmartNIC.

Our evaluation also includes the signing performance comparison, and the result is demonstrated in Figure 3.6. Our evaluation illustrates that the server not only demonstrates a better message signing performance but also delivers signature verification throughput. More specifically, for non-ECC-based ciphers, such as 2048-bit DSA, 2048-bit RSA, and 4096-bit RSA, the server can verify up to 9.3x, 11.1x, and 10x more messages per second than the SmartNIC employing crypto-hardware accelerator. Unlike signing, where ECDSA's throughput is generally higher than other asymmetric ciphers we tested with the same setup, ECDSA's message verification performance was comparable to that of non-ECC-based ciphers. However, we notice that for the scenario where the SmartNIC utilizes the crypto-hardware accelerator, ECDSA with curve P-256 gives an 81% worse performance than 4096-bit RSA. Our evaluation also includes the message verification performance comparison with different elliptic curves in Table 3.2. From the table, we can see that the crypto-hardware accelerator (PKA) is losing its advantage in all the cases, suggesting that the software optimizations on the ARM architecture mentioned above significantly impact the SmartNIC CPU utilization and improve the performance.

Key exchange. Key exchanges with the asymmetric ciphers are used to share secrets between two or more communicating parties. Historically RSA and prime field Diffie-Hellman (DH) has been the priority choice for the key exchange. However, due

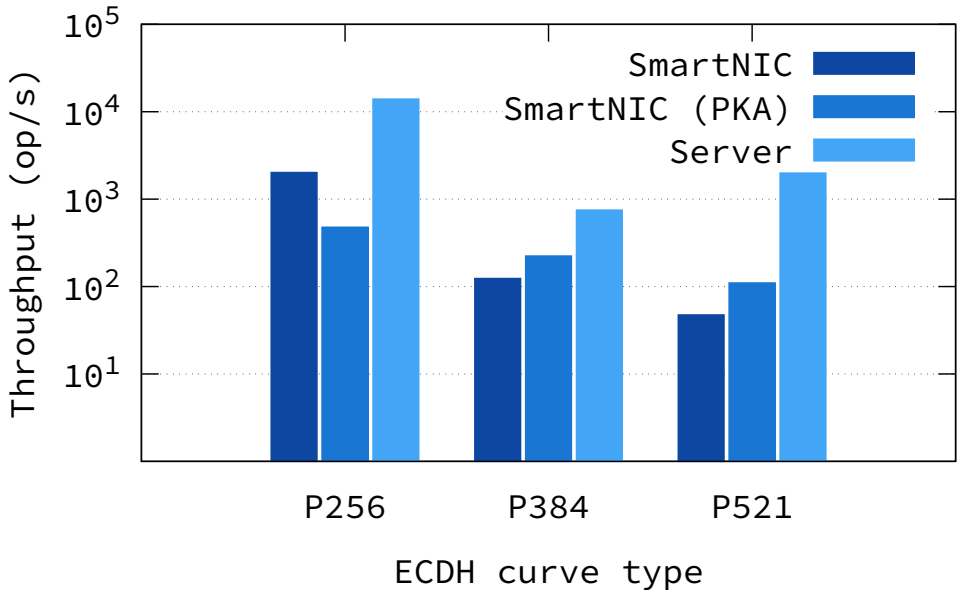


Figure 3.7: ECDH Throughput Comparison

to the complexities of implementing RSA and some other security concerns, elliptic-curve Diffie-Hellman (ECDH) has become increasingly popular, and a predominant option in the state-of-the-art solutions [78]. ECDH is a variant of Diffie–Hellman protocol that uses an elliptic-curve group; specifically, it involves two parties creating an elliptic-curve public/private key pair and distributing it over an insecure channel. The shared secret can then compute on both sides using this key exchange method by combining the received key with their own (private) one; this shared secret can then be used directly as a key or generate another key based on this secret. Other studies suggest that NIST P-256 is still the most commonly used curve so far, but P-384 and P-521 are also becoming popular choices in some of the commonly used modern applications like TLS connections [74] [78].

Figure 3.7 shows throughput compersion for running ECDH with different curve types on both server and SmartNIC. According to our evaluation, the server outperforms SmartNIC in all cases (up to 18x for curve P-521 compared to the SmartNIC with crypto-hardware accelerator setup) [2]. Moreover, as the key size grows, the ECDH's performance decreases as we expected on the SmartNIC since the larger key size requires more operations to process [79]. However, the server does not follow this

trend; instead, the performance relies on the optimizations supported by the processor [2]. For example, one study [80] recently optimized the NIST P-256 and P-521 curves for x86 processors, and those support has been added to OpenSSL. When compared to original OpenSSL implementations, these enhancements yield significant speedups. Thus we could see a better outcome for the P-256 and P-521 curves compared to the P-384 curve on the server.

Takeaways

- The algorithmic improvement and optimization of the cryptographic primitives could, in some cases, outperform the crypto-hardware accelerator solutions. Therefore, it is essential to have a quick evaluation before setup the real-world applications, especially for emerging hardware.

- Moreover, the data transmission overhead between SmartNIC and the hardware accelerator when processing cryptographic operations could also be a non-negligible factor in downgrading the overall performance; thus, it should be considered, especially for tasks with shorter operational time.

Chapter 4

The Case for SmartNIC Cryptography Offload: Design and Evaluation

4.1 Overview

This chapter discusses the advantages of offloading major security applications onto SmartNICs. Moreover, we look at three applications in particular: VPN tunneling, user authentication, and secure web serving. These applications were chosen based on several substantial justifications below: (i) those applications could provide representatives of use cases in different domains where the SmartNICs are widely adopted (e.g., public cloud data centers, university campuses, enterprise networks) (ii) applications that selected are often the primary choices of those use cases in the real-world scenario, which contribute practical value (iii) they use different combinations of the security primitives we studied in Chapter 3, and security plays an essential role in the design and deployment those applications; moreover, (iv) they are open-source, which allowing the community to reproduce the results of this work easily.

In our evaluation, we are especially interested in those real-world cases throughput and latency because those are two of the most noticeable factors to evaluate their performance and could provide a generalization comparison in different scenarios.

4.2 VPN Tunneling

Internet users commonly use virtual private network (VPN) services to protect their privacy, avoid censorship, and access geo-filtered content. It is also utilized by many corporations and organizations to let their workers access internal network resources and add a shell for their intellectual properties from being revealed by third parties. Moreover, more people are accepting VPNs in their everyday and professional life due to the awareness of security and privacy concerns and the need for remote work. According to a recent analysis [81], the global VPN industry is expected to reach

more than USD 100 billion by 2027, with a market value of approximately USD 35.4 billion in 2020.

Although multiple VPN tunneling applications and protocols (e.g., PPTP, SSTP, SSL, WireGuard) exist on the market, OpenVPN and IPsec are still the most popular protocol selections and are utilized by a vast portion of VPN services [82]. OpenVPN, along with other VPNs (e.g., WireGuard), are the route-based VPNs that support dynamic routing information swaps through VPN tunnels. In contrast, our analysis focuses on the IPsec (IKEv2/IPsec, specifically), a policy-based VPN for traffic tunneling. In the policy-based VPNs, virtual network interfaces are replaced by firewall rules to determine which traffic belongs to the VPN so that the permitted traffics are encapsulated and sent through encrypted messages [83]. The secure session establishment process using Internet Key Exchange version 2 (IKEv2) inside the IPsec protocol suite can be breakdown into two phases: The first phase generates a security association (SA), which lists a set of cryptographic parameters such as a shared secret and an encryption/decryption algorithm that could allow IKE messages to transmit securely between two communicating parties. Then in the second phase, it produces a secondary SA (or "child SA") to use as a tunnel for the two parties to set up authentication and begin data exchange [2]. IPSec mainly relies on symmetric encryption to set up private data communication. Notice that, in practice, there is a user authentication step before setting up the VPN tunnel. Our evaluation does not consider this step and only focuses on setup the VPN tunnel directly using IPsec.

4.2.1 Evaluation and analysis

Figure 4.1 shows our experimental setup for VPN tunneling. Our setup includes two endpoints; to better address those two parties, we name them as VPN client and server accordingly, as shown in the figure. The VPN client has the specification with Intel Core i7-9700 @ 3.0 GHz CPU with 8 cores and 16 GB of DDR4 DRAM; the operating system is Ubuntu 18.04.6 LTS (kernel 5.4.0-90-generic). The server and SmartNIC, which are used as the VPN server in this scenario, are the same as described in Section 3.2. Our evaluation compares two VPN tunneling scenarios: client-CPU (named "server") and client-SmartNIC. Both scenarios deploy the VPN tunnels using StrongSwan (version U5.8.2/K5.4.0-74-generic) [84]. For IKEv2

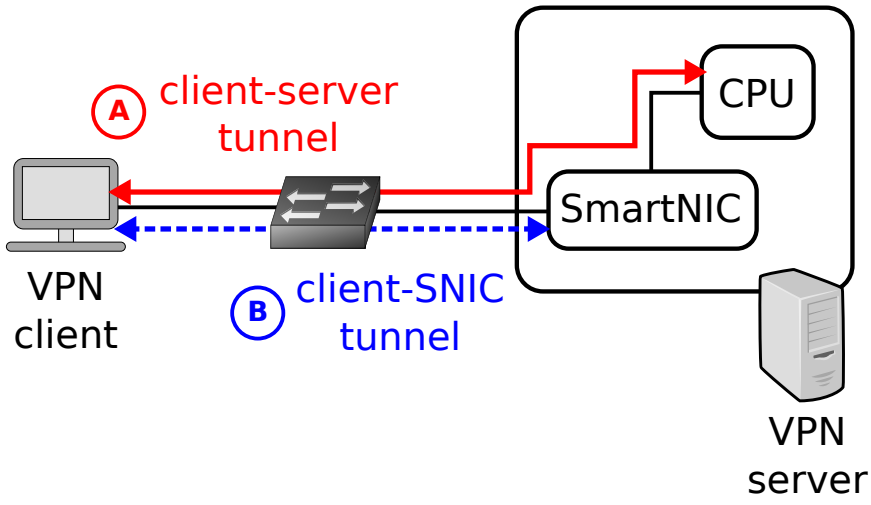


Figure 4.1: VPN tunneling setup

authentication, both server and SmartNIC tunnels employ a pre-shared key (PSK) and SHA1 for hashing. Furthermore, two sets of experiments are executed in our evaluation: i) the VPN client pings 100 times to the VPN server and measures the average round-trip latency, the ping process sequentially to eliminate the influence of other factors. ii) the same VPN client uses *iperf3* (version 3.1.3) [85], a network performance measurement tool, to generate a 1-minute long TCP flow and measures the average throughput. Our study repeated each experiment 10 times and reported the data encryption/decryption results, which used the selected AES-256-GCM and AES-256-CBC ciphers in the operations; and noticed that the Chacha20/Poly1305 cipher is ignored from our evaluation for this case study since we could not run the associated StrongSwan plugin on the SmartNIC. Below, we break down our case study result analysis for VPN latency and throughput.

Latency. The average round-trip latency for the VPN encryption/decryption with various algorithms is shown in the Figure 4.2. In the result, we use the plain-text (ping) latency to represent the "baseline" result. Our evaluation shows that in comparison to the server, the SmartNIC provides much lower latency which could be up to 52% lower in the GCM mode. This result reflects how the SmartNIC's physical position could positively influence the overall latency. To be more specific, the SmartNIC, when it plays the role of the VPN server, is "one hop" closer to the

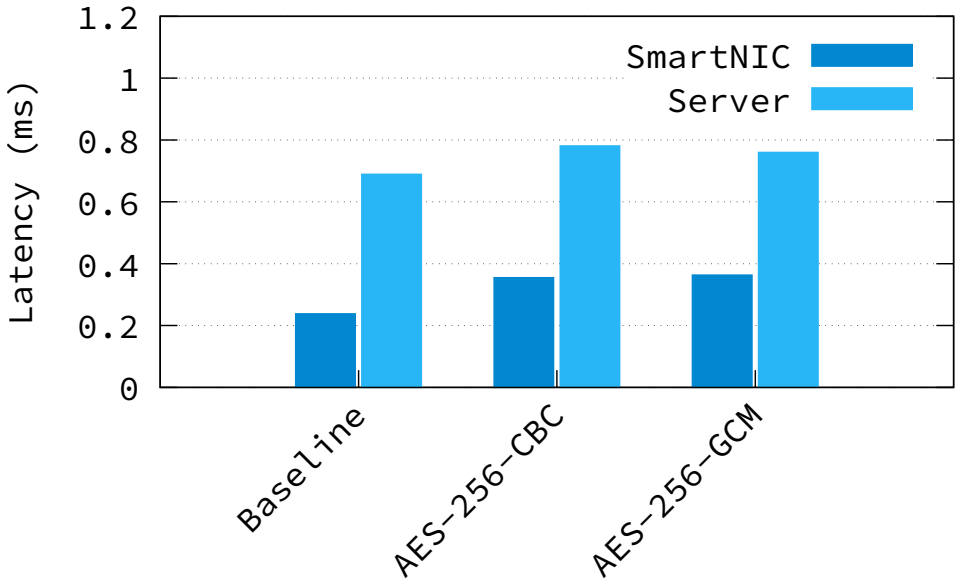


Figure 4.2: Average round-trip latency in a VPN tunnel

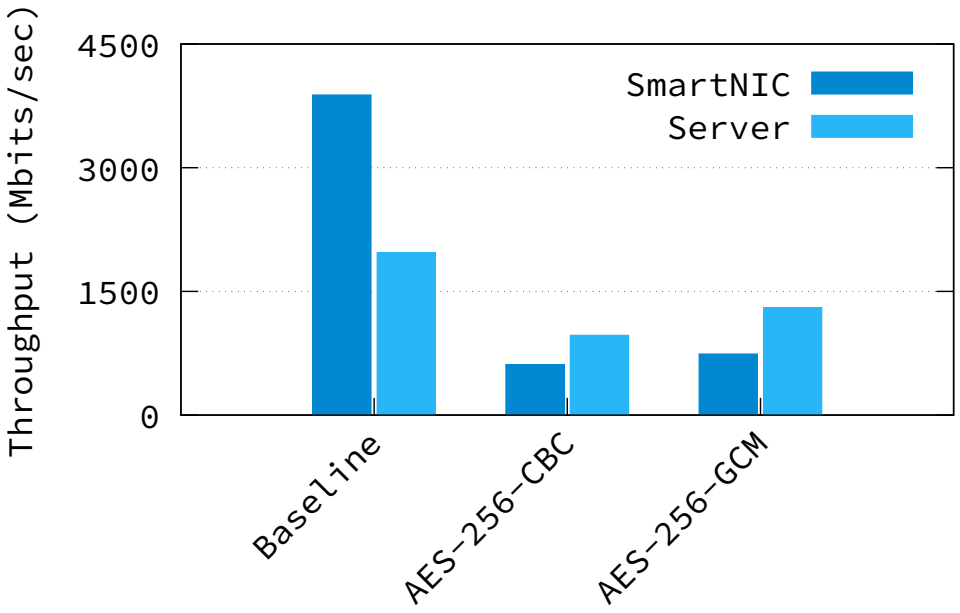


Figure 4.3: Average TCP throughput in a VPN tunnel

VPN client, which is more noticeable in short-path conditions (e.g., an edge data center) [2]. Moreover, compared with the baseline approach, we find that adding the VPN tunnel only contributes a relatively low overhead (less than 35%). This small overhead is due to the message encrypted and decrypted in this case by the VPN endpoints being relatively small, i.e., the ping process only has a tiny request and

reply packet.

Throughput. Although from our previous analysis, the SmartNIC could perform well with the low cryptographic operation load settings, its performance suffers dramatically as the load increase. Our throughput stress test results are present in Figure 4.3. Because the SmartNIC is a high-speed network device and no additional processing is required from its CPU cores when dealing with plaintext TCP (i.e., non-VPN traffic), we notice that in our result, the SmartNIC performs far better than the server. However, for the VPN traffic, which requires the use of crypto-processing, there is an inverted trend demonstrated in our result. For example, the throughput of the SmartNIC and server for an AES-GCM-based VPN tunnel could be approximately 739 and 1300 Mbps, respectively.

When comparing the performance of AES-GCM and AES-CBC on both hardware, we find out that AES-GCM's performance results are generally better than AES-CBC, which is contradictory to our basic cryptography performance evaluation in Section 3.3.1. Our study concluded that the extra cost of processing SHA1 in real-world applications contributes to this worse performance of AES-CBC. More specifically, this SHA1 process is used for message authentication in setting up the VPN tunnel along with the AES-CBC for message authentication; on the other hand, AES-GCM is an AEAD algorithm, as mentioned in the previous Section 3.3.1, which has the message authentication build-in. Thus, when this message authentication becomes a requirement in the real-world application, the overall throughput for the application that uses AES-CBC surpasses the one that uses AES-GCM.

4.2.2 Takeaways

- SmartNICs are "one hop" closer to the client than the server, giving it a position advantage when offloading the task, which could help accelerate the crypto-based networked applications. For network tasks without cryptography, this advantage is more prominent.

- Moreover, the latency-critical applications with lighter cryptography tasks can benefit more from the position advantage of the SmartNIC than bandwidth-intensive applications.

4.3 User Authentication

User authentication widely appears in modern applications; whether it is the user's login on the web services or the identity verifications between IoT devices, the authentication system has already become the gatekeeper of online security. Moreover, many enterprises, ISPs, and educational institutions deploy user authentication systems to protect and control their IT resources' access. As a result, using modern SmartNIC as an assistant for the server CPU in reducing the load on it when operating the authentication servers could be a reasonable choice [2]. Notably, network operators have identified poor resourcing of authentication servers as one of the most common challenges while running this vital infrastructure [86].

In terms of the standard authentication system structure, typically, there is an access server and an authentication server. When the user (e.g., a VPN client or IoT device) attempts to log in to a particular network, it first makes an access request to an access server to grant the access. Next, the access server queries the authentication server to confirm the user credentials. The authentication server then processes the query by matching the user database with the hash of user credentials, and the database could be either local or remote. Finally, the user's access query is granted or denied based on that matching result between user credentials and database information [87].

4.3.1 Evaluation and analysis

Figure 4.4 shows our evaluation setup for the user authentication. Our experiment compares two authentication scenarios: i) sending the user credentials and letting the server CPU (we call it "server") do the authentication (scenario A-B), and ii) sending the user credentials and authenticating by the SmartNIC (scenario A-C). In this case study, we use the same hardware setup as described in Section 4.2 for the network access server. Moreover, the server and SmartNIC that host the authentication server uses hardware specifications we described in Section 3.2.

The authentication service we set up uses the RADIUS (Remote Authentication Dial-In User Service) protocol. We use *radperf* (version 2.0.1) [88] to simulate both the client device and access server, and it controls the desired rate we need to send

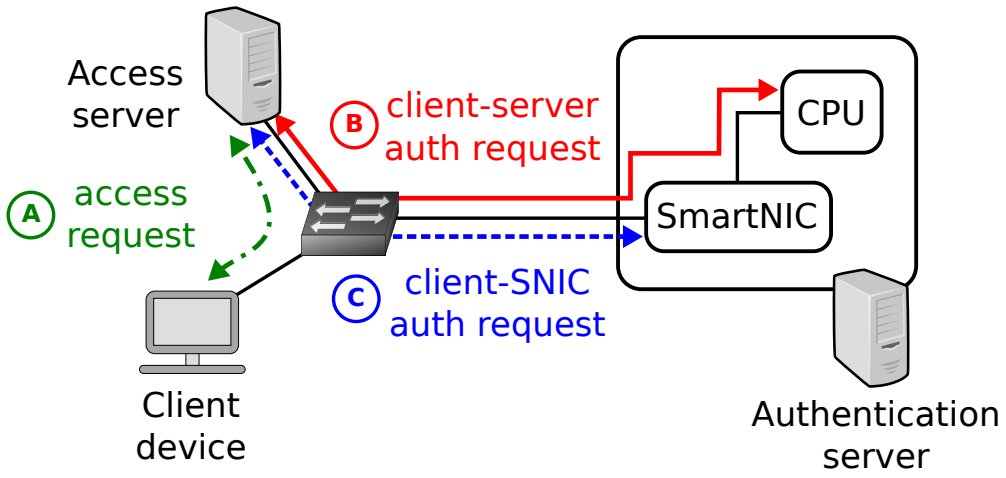


Figure 4.4: User authentication setup

Table 4.1: Average, 95th and 99th percentile of the round trip latency (in miliseconds) for processing a batch of 1K authentication requests

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

the authentication requests. The authentication server is set up to process requests using FreeRADIUS (version 3.0.20), a free and open-source suite of tools for setting up the RADIUS server [89]. For the FreeRADIUS configuration, we keep the default settings, including UDP as transport protocol and SHA-256 as the hashing function for the password. The UDP is selected on the FreeRADIUS as the default setting, mainly because of its faster speed compared to TCP, and RADIUS has a few inherent qualities that are characteristic of UDP (e.g., the stateless nature) [90]. Moreover, to have a fair comparison between the SmartNIC and server CPU and to eliminate the influence of different numbers of cores and clock speed frequency, our experiment only allows the authentication server runs with a single-core and single-thread mode. To make the result more accurate, we repeat each test run 100 times and take the average.

Latency. Table 4.1 compares the latency for each device for processing a thousand authentication requests issued at an unlimited rate. From the result, we found out

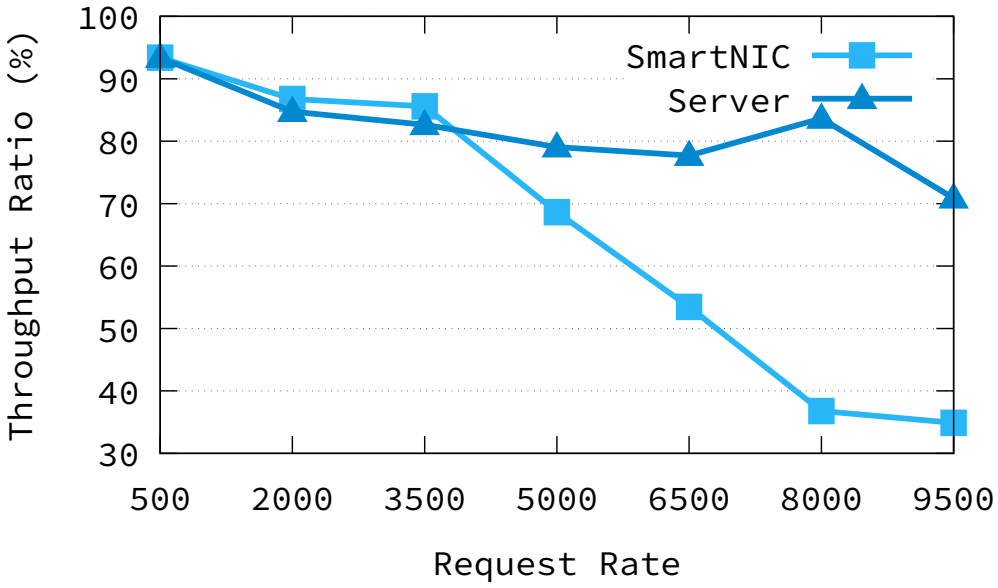


Figure 4.5: Throughput ratio (i.e., the ratio of served authentication requests) as a function of the request rate

that compared to the server, SmartNIC is significantly faster (up to 50% on average). There are two main reasons for this faster result: First, the SmartNIC is "closer" to the client device, i.e., the network access server, which gives it the advantage in terms of network latency, just as we discussed in the VPN study case. Second, as our discussion in Section 3.3.2, the SmartNIC could utilize the cryptographic instructions that provide the hash acceleration features, which could make the checking user credentials process much more efficient. Additionally, the SmartNIC has substantially shorter tail latencies (approximately 52% at the 99th percentile), which can be critical if the workload is deadline-oriented [2].

Throughput. The setup above represents a bursty workload in which we load as much traffic as possible in a short period. In expansion, we focus on a continuous traffic loading scenario for this evaluation. Figure 4.5 shows the result for this scenario, it demonstrates the relationship between request rate and devices' throughput ratio. This result helps us understand the proportion of utilization from the input load; the higher the ratio, the better data it can be served. Our evaluation shows that in compersion to the server, the SmartNIC appears to hit its bottleneck more quickly (at around 3.5 K requests per second. In contrast, the server can maintain

a fairly high throughput ratio (beyond 70% of the input rate) up to 9.5K requests per second, even though its throughput ratio slowly drops when the request rate increases. Our analysis indicates that this slowly dropping in the throughput ratio is due to the background process (e.g., garbage collection) that is an essential part of FreeRADIUS [2].

4.3.2 Takeaways

- SmartNICs have the advantage when offloading tasks requiring only specific cryptography instructions (e.g., tasks that repeatedly compute SHA-256 hashes on the SmartNIC cores).

- On the other hand, this advantage can be readily outweighed in a mixed workload. Therefore, offloading crypto applications' workload is crucial to be considered when deploying the offloading engines.

4.4 Secure Web Serving

A web server is computer software and its underlying hardware that uses HTTP (Hypertext Transfer Protocol) and other protocols to respond to client requests made over the Internet. HTTPS (Hypertext Transfer Protocol Secure) is the secure variant of HTTP and can also use on the web server for secure web serving. The adoption of HTTPS has significantly grown over the last few years [91] and has already become the default way to surfer the websites' content on major browsers. Accordingly, based on the report from Google, more than 95% of its services are currently serving HTTPS by default [92]. As a result of this trend, several preliminary studies (e.g., [8], [48]) are on offloading the TLS operations to SmartNIC to accelerate the secure web serving speed. Despite the fantastic existing outcome, there are still a few important questions that remain unanswered. Our study tries to touch on two of them: i) how is the performance of the web server regarding a full (rather than partial) application offload on the SmartNIC; and ii) how is the change from TLS 1.2 to TLS 1.3 influence the performance when offloading the web server onto the SmartNIC completely.

Two practical observations drive our questions above. First, partial offloads (e.g., only offloading the TLS handshake part or data encryption/decryption part from the

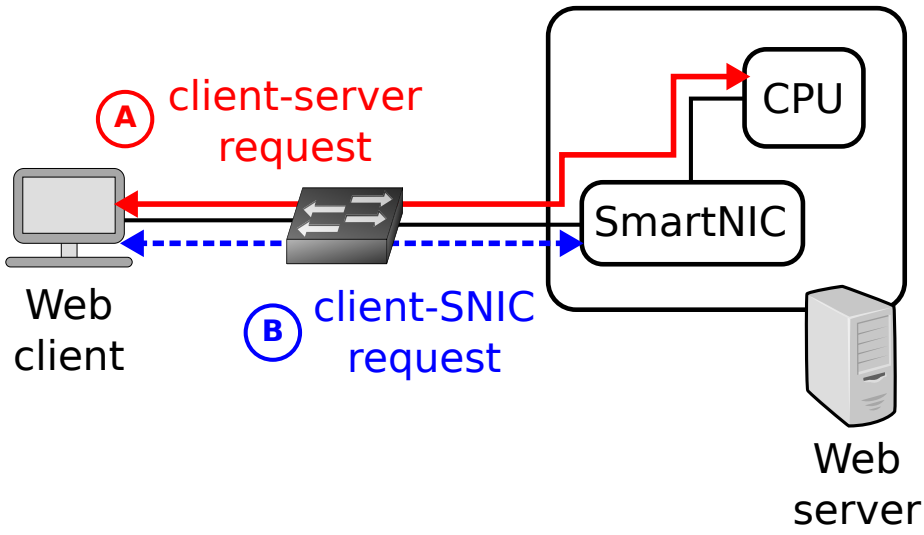


Figure 4.6: HTTPS server setup

entire web serving process) require customization of the protocol. This customization requires extra work on hardware support, code maintenance, debugging, and security patches; moreover, partial offloads may not be economically sustainable compared to full offloads when SmartNIC hardware keeps improving. Second, there is a considerable performance boost in TLS 1.3 compared to its predecessors. To be more specific, the most recent version, in particular, decreases the number of packets that the web server needs to handle by restructuring the client-server handshake process to a single RTT (Round Trip Time) by combing "hello" and key exchange messages [93]. Furthermore, using the "0-RTT" (zero round trip time resumption) mode with TLS 1.3 could make this performance improvement even more noticeable for the initiation of the secure web serving [93].

4.4.1 Evaluation and analysis

Figure 4.6 shows our secure web serving experiment setup. The web server and the web client are running with the same specification we introduced in Sections 3.2 and 4.2. Our experiment compares two scenarios: i) the HTTPS requests to the web server that handles fully on the server, and ii) the HTTPS requests that handle fully on the SmartNIC. To set up our web server, we use NGINX (version 1.18.0) [94] for request serving; it runs on a large fraction of web servers with easy-to-use features,

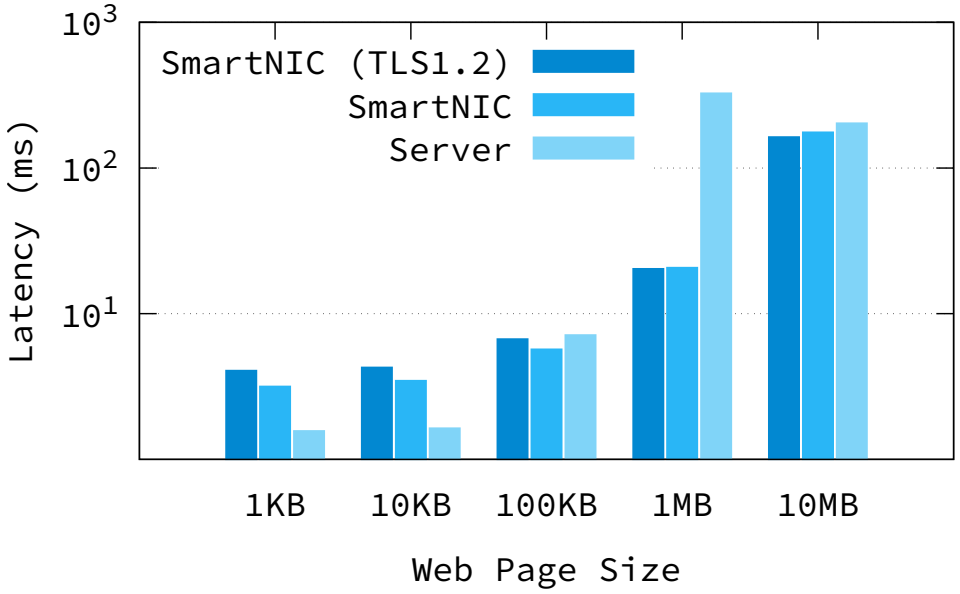


Figure 4.7: Average web server latency

lightweight size, and outstanding performance. We keep all the NGINX configurations as default except the number of workers and custom certificate. For the worker, we only use a single worker to eliminate the influence; the worker is a concept in NGINX where one worker could process multiple requests parallelly, and the request is not necessarily part of the same process. Furthermore, for the custom certificate, we used a self-signed certificate in our setup to reduce the network overhead from fetching the certificate from the Internet. The certificate has a 4096-bit key and uses SHA-256 as its hash function; the security properties on the certificate (e.g., key usage) are compliant with the stand X.509 v3 extensions. To set up our web client, we use the *wrk* [95] benchmarking tool (version 4.2.0). In our experiment, we set the tool to operate on a single thread and a fixed number of 50 parallel connections (i.e., throughout the experiment, the client will keep 50 open connections at any given moment). Each TCP connection in our experiment serves a single web page request, and all the web pages are randomly generated before the experiment with the exact page size we need. Each experiment performs a fixed three minutes, and our result takes an average of 10 runs.

Latency. Figure 4.7 shows the average request latency for web page with different sizes. The result shows that even though the SmartNIC is "one hop" closer to the web

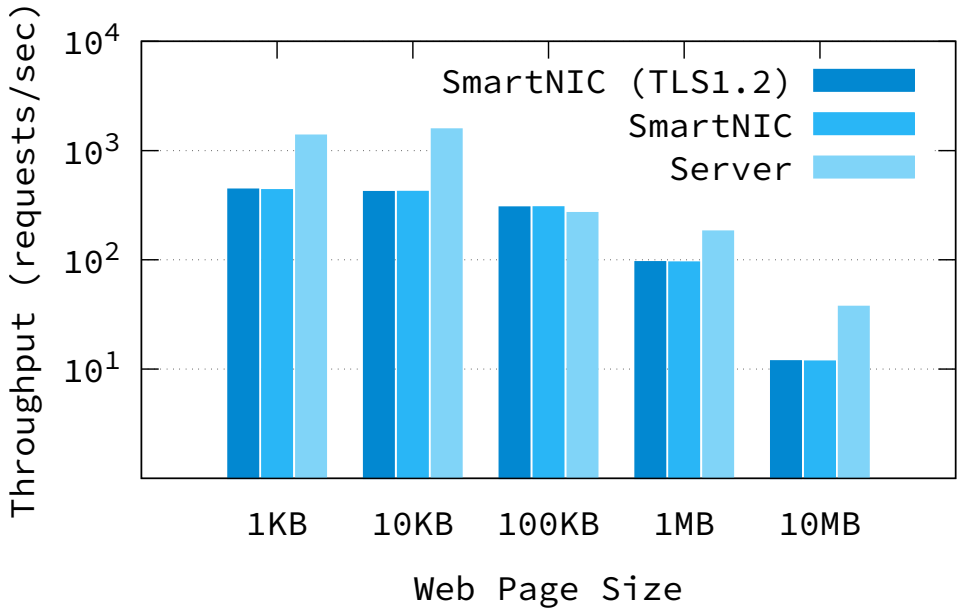


Figure 4.8: Average web server throughput

client than the server, it still performs worse for small requests (up to 10 KB). We believe this worse performance is caused by the SmartNIC's utilization of its crypto-hardware accelerator during the network connection setup, like TLS handshake, which adds extra overhead when transferring data between the crypto-hardware accelerator and the processor, as we saw in Section 3.3.3. When the request size gets larger, server and SmartNIC latencies tend to converge as the request size grows. However, we could spot an interesting discord for 1 MB requests, where the SmartNIC outperformed the server significantly. Our study noticed that as a trade-off between the number/size of segments and the overall request latency, this is contributed by NGINX dividing large web pages into smaller parts before sending them to the client [96]. More specifically, the SmartNIC, which has a weak processor but a closer position advantage than the server CPU, could benefit from a more significant number of smaller segments instead of a large piece. This behavior is not directly linked to the cryptography features of the SmartNIC; therefore, there could be a further investigation of the latency versus fragment size trade-off as future work. Finally, the compersion between TLS 1.2 and TLS 1.3 shows that the newer version has a slightly better output; this is more pronounced for the smaller web pages, where the handshake stage has a more considerable contribution to the entire overhead.

Throughput. Figure 4.8 presents how the throughput (in requests per second) changes on both the server and the SmartNIC as we increase the web page size that the client request. As we expected, larger web page requests could decrease the performance on both devices because they add a higher load on the data transmission and require more data encryption/decryption. Moreover, we can see that the server generally has better performance than SmartNIC. Notice that the server's throughput is remarkably better when there are higher connection requests in the network (i.e., 1 KB web page size) or encryption required by more extensive data (i.e., 10 MB web page size). Our study believes this behavior is because a secure web server requires processing multiple cryptography operations, and various crypto-acceleration mechanisms working together on the server could fulfill this need. Those mechanisms include dedicated server instructions with the public-key cryptography that benefits the handshake phase, symmetric encryption instructions for data encryption/decryption, and data parallelism that are more powerful than SmartNIC [97]. In contrast, current SmartNIC is generally obscure for the mixed workload, and it could perform better on the task with a specific set of cryptography algorithms.

4.4.2 Takeaways

- Applications using a wide variety of cryptographic methods (e.g., the application requires a mixed-use of symmetric and asymmetric ciphers) could face barriers due to the limited hardware crypto-acceleration support on the SmartNICs. This feature leads to a performance loss of up to 73% for a secure web server.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

Offloading applications onto the SmartNICs has been a practical and promising approach for improving the network application performance continuously. While many researchers have studied this trend for "plaintext" applications, the cryptography capabilities of the current SmartNICs still need additional investigation. This thesis describes our contribution to the first in-depth systematic analysis of the current cryptography-based workload support on the SmartNIC.

We have summarized the cryptographic functionalities of six top commercial SmartNICs, then structured the analysis for the ARM-based device with its basic cryptography capabilities and its performance with complex security applications. Our findings demonstrate that current SmartNIC designs can benefit latency-sensitive operations, although caution is advised when dealing with computationally demanding workloads. Mainly, SmartNICs' cryptography capabilities strongly depend on the support of the crypto-hardware accelerators to meet servers' performance; and might not be beneficial for the complex workload that requires more generalized computing capabilities.

5.2 Future Work

This study covers a broad range of cryptography algorithms that are commonly used in modern applications. However, cryptography or network security is a large domain that can be explored from different aspects. We believe our methodology in this paper could be extended to help study and analyze other security features on the same devices.

Random number generator (RNG). Generating random numbers can be

critical to a cryptography application; some algorithms require the RNG for certain randomness to achieve the desired security. Applications like cryptography key generation [98] or smart contracts [99] need certain level of randomness. However, many applications still use the software-based pseudorandom number generators that use pre-determined functions. SmartNIC, on the other hand, often comes with a hardware-based random number generator (known as the true random number generator) that could create random numbers that is practically impossible to model. Furthermore, there is no current study on the performance of the hardware-based RNGs on the SmartNIC, which could be valuable for further analysis.

Trusted execution environment (TEE). TEE is a secure area of the processor; it helps preserve the data inside with confidentiality and integrity. TEE is helpful for cloud providers or companies that need a network infrastructure to provide confidential information, authentication, or proceed with confidential computing. Well-known TEE solutions are ARM TrustZone [100] and RISC-V Keystone [101]; those features are commonly enabled on the SmartNIC with the corresponding processors. Specifically, different SoCs come with different TEE architectures that could create performance impacts of applications offloaded to these environments [102], which could be explored further.

Energy consumption. Cryptography tasks are always considered heavy-loaded and play an essential role in network applications, as described in this study; therefore, they also contribute a noticeable amount of energy consumption. In some cases, even though SmartNICs cannot provide comparable computing power to the server, it could still be valuable to handle some offloading tasks because SmartNICs generally have better performance per watt. Therefore, future studies could address this possible advantage of the SmartNIC. However, it is worth knowing that it is hard to have an accurate power consumption measurement for the processor. It will be even more challenging if we want to measure the hardware accelerator along with it. We can measure energy using hardware or software tool; hardware measurement [103] is often more accurate but less flexiable, whereas software one [104] is less accurate but more flexiable. Some researchers developed more accurate statistical methods for a better energy consumption measurement [105] [106]. However, those studies still need extensive evaluations if they want to apply to different architectures or workloads,

especially if we want to measure precisely for security and cryptography applications.

Furthermore, having a fair comparison between server and SmartNIC with different hardware architectures is even more complicated because the power measurement models are different across different architectures. Therefore, the best way so far, also selected by the previous study [3], is to measure the wall power (measure directly from the power supply) for the comparison. Their study measured the idle and active wall power difference, which is a relatively straightforward way to demonstrate the energy cost comparison for the server and the SmartNIC. Therefore, there should be more justification on which methodology to use for power consumption measurement in future studies.

Modern ciphers. Different security features are not the only things we could extend in future studies; it is also worth noticing that cryptographic algorithms are constantly evolving. Although our study covers the vast majority of cryptography primitives that are supported on current (co)processors, many other experiential "modern" cryptographic algorithms (e.g., post-quantum and fully homomorphic ciphers) are developed to mitigate future threats or used in novel applications. Moreover, since many of those "modern" ciphers do not have the same support on the state-of-the-art hardware, there could be further explorations on utilizing multi-core SmartNICs as an option to offloading post-quantum and FHE-based cryptographic tasks, and the BLAKE (based on the ChaCha cipher) hash functions.

Table 5.1: Impact of Quantum Computing on Common Cryptographic Algorithms

Cryptographic Algorithm	**Type**	**Quantum Computer Impact**
AES	Symmetric cipher	Larger Key Sizes Needed
ChaCha20/Poly1305	Symmetric cipher	Larger Key Sizes Needed
SHA-256/SHA-512	Hash function	Larger Hash Output Sizes Needed
RSA	Asymmetric cipher	No Longer Secure
DSA	Asymmetric cipher	No Longer Secure
ECDSA/ECDH	Asymmetric cipher	No Longer Secure

To further explain how SmartNICs could work with post-quantum cryptography in future studies, we must first understand the quantum computer's impact. Table 5.1 lists the impact of quantum computers on the most commonly used cryptography algorithms nowadays based on existing studies [107] [108] [109]; those ciphers are also what we have chosen in our study.

To be more specific, hash and symmetric ciphers could still be usable under certain conditions in the post-quantum era. Moreover, the existing post-quantum cryptography categories include: lattice-based cryptography, multivariate cryptography, hash-based cryptography, code-based cryptography, supersingular elliptic curve isogeny cryptography, and symmetric key quantum resistance [110]. We observed that offloading hash-based cryptography onto the SmartNIC could be a promising future study direction because hash-based cryptography (e.g., XMSS [111] or SPHINCS+ [112]) can still benefit from the existing acceleration for SHA algorithms, and the SmartNICs could have significant secure hashing performance in some scenarios. We plan to explore such performance analysis in the future.

Bibliography

[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] J. Zhao, M. Neves, and I. Haque, "The case for smartnic crypto offload," IEEE Transactions on Dependable and Secure Computing, 2022, submitted.

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

[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] J. Kim, I. Jang, W. Reda, J. Im, M. Canini, D. Kostić, Y. Kwon, S. Peter, and E. Witchel, "LineFS: Efficient smartnic offload of a distributed file system with pipeline parallelism," in *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles CD-ROM*, 2021, pp. 756–771.

[6] H. N. Schuh, W. Liang, M. Liu, J. Nelson, and A. Krishnamurthy, "Xenic: SmartNIC-accelerated distributed transactions," in *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles CD-ROM*, 2021, pp. 740–755.

[7] 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.

[8] 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.

[9] [Online]. Available: https://github.com/printfer/smartNIC_benchmark

[10] M. Darianian, C. Williamson, and I. Haque, "Experimental evaluation of two openflow controllers," in *2017 IEEE International Conference on Network Protocols (ICNP) Workshop on PVE-SDN*, 2017.

[11] M. Shojaee, M. C. Neves, and I. Haque, "Safeguard: Congestion and memory-aware failure recovery in SD-WAN," in *16th International Conference on Network and Service Management, CNSM 2020, Izmir, Turkey, November 2-6, 2020*. IEEE, 2020, pp. 1–7. [Online]. Available: https://doi.org/10.23919/CNSM50824.2020.9269119

[12] M. A. Moyeen, F. Tang, D. Saha, and I. Haque, "SD-FAST: A packet rerouting architecture in SDN," in *15th International Conference on Network and Service Management, CNSM 2019, Halifax, NS, Canada, October 21-25, 2019*. IEEE, 2019, pp. 1–7. [Online]. Available: https://doi.org/10.23919/CNSM46954.2019.9012703

[13] U. Lekhala and I. Haque, "PIQoS: A programmable and intelligent qos framework," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops, INFOCOM Workshops 2019, Paris, France, April 29 - May 2, 2019*. IEEE, 2019, pp. 234–239. [Online]. Available: https://doi.org/10.1109/INFCOMW.2019.8845158

[14] F. Tang and I. Haque, "ReMon: A resilient flow monitoring framework," in *Network Traffic Measurement and Analysis Conference, TMA 2019, Paris, France, June 19-21, 2019*. IEEE, 2019, pp. 137–144. [Online]. Available: https://doi.org/10.23919/TMA.2019.8784521

[15] I. Haque and M. A. Moyeen, "Revive: A reliable software defined data plane failure recovery scheme," in *14th International Conference on Network and Service Management, CNSM 2018, Rome, Italy, November 5-9, 2018*, S. Salsano, R. Riggio, T. Ahmed, T. Samak, and C. R. P. dos Santos, Eds. IEEE Computer Society, 2018, pp. 268–274. [Online]. Available: https://ieeexplore.ieee.org/document/8584938

[16] H. Siddique, M. Neves, C. Kuzniar, and I. Haque, "Towards network-accelerated ML-based distributed computer vision systems," in *IEEE 27th International Conference on Parallel and Distributed Systems (ICPADS)*, 2021, pp. 122–129.

[17] C. Boeira, M. Neves, T. Ferreto, and H. Israat, "Characterizing network performance on single-node large-scale container deployments," in *IEEE 10th International Conference on Cloud Networking (CloudNet)*, 2021, pp. 97–103.

[18] C. Kuzniar, M. Neves, V. Gurevich, and I. Haque, "IoT device fingerprinting on commodity switches," in *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, 2022, pp. 1–9.

[19] F. Tang, M. Shojaee, and I. Haque, "ACE: an accurate and cost-effective measurement system in SDN," 2022. [Online]. Available: https://arxiv.org/pdf/2108.12849.pdf

[20] H. Tajbakhsh, R. Parizotto, M. Neves, A. Schaeffer-Filho, and I. Haque, "Accelerator-aware in-network load balancing for improved application performance," in *2022 IFIP Networking Conference (IFIP Networking)*, 2022.

[21] M. Kulkarni, M. Baddeley, and I. Haque, "Embedded vs. external controllers in software-defined IoT networks," in *2021 IEEE 7th International Conference on Network Softwarization (NetSoft)*, 2021.

[22] H. Ghannadrezaii, J.-F. Bousquet, and I. Haque, "Cross-layer design for software-defined underwater acoustic networking," in *IEEE OCEANS*, 2019.

[23] I. Haque and D. Saha, "SoftIoT: A resource-aware SDN/NFV-based IoT network," *The Elsevier Journal of Network and Computer Applications*, vol. 193, Nov 2021.

[24] D. Saha, M. Shojaee, M. Baddeley, and I. Haque, "An Energy-Aware SDN/NFV architecture for the internet of things," in *IFIP Networking 2020 Conference (IFIP Networking 2020)*, Paris, France, Jun. 2020.

[25] I. Haque, M. Nurujjaman, J. Harms, and N. Abu-ghazaleh, "SDSense: An agile and flexible SDN-based framework for wireless sensor networks," *The IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1866 – 1876, February 2019.

[26] I. Haque and N. Abu-Ghazaleh, "Wireless software defined networking: a survey and taxonomy," *IEEE Communications Surveys and Tutorials*, vol. 18, no. 4, pp. 2713–2737, May 2016.

[27] K. Srinivasan, "The rise of smartnics," Jun 2021. [Online]. Available: https://semiengineering.com/the-rise-of-smartnics/

[28] M. Liu, T. Cui, H. Schuh, A. Krishnamurthy, S. Peter, and K. Gupta, "ipipe: A framework for building distributed applications on multicore soc smartnics," in *Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM)*, 2019.

[29] T. Döring, H. Stubbe, and K. Holzinger, "Smartnics: Current trends in research and industry," *Network*, vol. 19, 2021.

[30] S. Schweitzer, "Smartnic architectures: A shift to accelerators and why fpgas are ..." Jul 2020. [Online]. Available: https://www.electronicdesign.com/industrial-automation/article/21136402/xilinx-smartnic-architectures-a-shift-to-accelerators-and-why-fpgas-are-poised-to-dominate

[31] S. Cybersecurity, V. Clifton, and R. Hat, "Guide to ipsec vpns." [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-77r1.pdf

[32] I. T. L. Computer Security Division, "Hash functions: Csrc," Jan 2017. [Online]. Available: https://csrc.nist.gov/projects/hash-functions

[33] V. Bhatia and K. Ramkumar, "An efficient quantum computing technique for cracking rsa using shor's algorithm," in *2020 IEEE 5th international conference on computing communication and automation (ICCCA)*. IEEE, 2020, pp. 89–94.

[34] C. Gidney and M. Ekerå, "How to factor 2048 bit rsa integers in 8 hours using 20 million noisy qubits," *Quantum*, vol. 5, p. 433, 2021.

[35] F. Mantovani, M. Garcia-Gasulla, J. Gracia, E. Stafford, F. Banchelli, M. Josep-Fabrego, J. Criado-Ledesma, and M. Nachtmann, "Performance and energy consumption of hpc workloads on a cluster based on arm thunderx2 cpu," *Future generation computer systems*, vol. 112, pp. 800–818, 2020.

[36] K. Keipert, G. Mitra, V. Sunriyal, S. S. Leang, M. Sosonkina, A. P. Rendell, and M. S. Gordon, "Energy-efficient computational chemistry: Comparison of x86 and arm systems," *Journal of chemical theory and computation*, vol. 11, no. 11, pp. 5055–5061, 2015.

[37] S. Biookaghazadeh, M. Zhao, and F. Ren, "Are FPGAs suitable for edge computing?" in *USENIX Workshop on Hot Topics in Edge Computing (HotEdge 18)*, 2018.

[38] W. J. Dally, Y. Turakhia, and S. Han, "Domain-specific hardware accelerators," *Communications of the ACM*, vol. 63, no. 7, pp. 48–57, 2020.

[39] M. Nabeel, M. Ashraf, E. Chielle, N. G. Tsoutsos, and M. Maniatakos, "Cophee: Co-processor for partially homomorphic encrypted execution," in *2019 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 2019, pp. 131–140.

[40] 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.

[41] "High speed public key accelerator," Mar 2021. [Online]. Available: https://www.rambus.com/security/protocol-engines/high-speed-public-key-accelerator/

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

[43] K. Investments, "Qualcomm (qcom): Arm cpu market presents $3.8 bln opportunity," Mar 2022. [Online]. Available: https://seekingalpha.com/article/4491550-qualcomm-arm-cpu-market-presents-3-8-bln-opportunity

[44] P. Longa, "Four{Q}neon: Faster elliptic curve scalar multiplications on arm processors," in *International Conference on Selected Areas in Cryptography*. Springer, 2016, pp. 501–519.

[45] 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.

[46] H. Eran, L. Zeno, M. Tork, G. Malka, and M. Silberstein, "NICA: An infrastructure for inline acceleration of network applications," in *2019 USENIX Annual Technical Conference (USENIXATC 19)*, 2019, pp. 345–362.

[47] P. M. Phothilimthana, M. Liu, A. Kaufmann, S. Peter, R. Bodik, and T. Anderson, "Floem: A programming system for NIC-accelerated network applications," in *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, 2018, pp. 663–679.

[48] 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.

[49] R. Shashidhara, T. Stamler, A. Kaufmann, and S. Peter, "FlexTOE: Flexible tcp offload with fine-grained parallelism," *arXiv preprint arXiv:2110.10919*, 2021.

[50] X. Yang, L. Eggert, J. Ott, S. Uhlig, Z. Sun, and G. Antichi, "Making QUIC quicker with NIC offload," in *Proceedings of the Workshop on the Evolution, Performance, and Interoperability of QUIC*, 2020, pp. 21–27.

[51] M. S. Brunella, G. Belocchi, M. Bonola, S. Pontarelli, G. Siracusano, G. Bianchi, A. Cammarano, A. Palumbo, L. Petrucci, and R. Bifulco, "hXDP: Efficient software packet processing on FPGA NICs," in *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, 2020, pp. 973–990.

[52] H. Tajbakhsh, R. Parizotto, M. Neves, A. Schaeffer-Filho, and I. Haque, "Accelerator-aware in-network load balancing for improved application performance," in *IFIP Networking Conference 2022*, 2022.

[53] 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.

[54] C. Kuzniar, M. Neves, and I. Haque, "Poster: Accelerating encrypted data stores using programmable switches," in *2020 IEEE 28th International Conference on Network Protocols (ICNP)*. IEEE, 2020, pp. 1–2.

[55] 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.

[56] 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.

[57] 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.

[58] P. Kietzmann, L. Boeckmann, L. Lanzieri, T. C. Schmidt, and M. Wählisch, "A performance study of crypto-hardware in the low-end iot." *IACR Cryptol. ePrint Arch.*, vol. 2021, p. 58, 2021.

[59] A. Naser, M. S. Lahijani, C. Wu, M. Gavahi, V. T. Hoang, Z. Wang, and X. Yuan, "Performance evaluation and modeling of cryptographic libraries for mpi communications," *arXiv preprint arXiv:2010.06139*, 2020.

[60] "Why intel® aes-ni matters." [Online]. Available: https://www.intel.com/content/www/us/en/architecture-and-technology/advanced-encryption-standard-aes/data-protection-aes-general-technology.html

[61] Mellanox, "Mellanox/bfb-build: Bfb (bluefield boot stream and os installer) build environment." [Online]. Available: https://github.com/Mellanox/bfb-build

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

[63] "Improving aes-gcm performance," Sep 2017. [Online]. Available: https://blog.mozilla.org/security/2017/09/29/improving-aes-gcm-performance/

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

[65] N. Sullivan, "Padding oracles and the decline of cbc-mode cipher suites," Aug 2021. [Online]. Available: https://blog.cloudflare.com/padding-oracles-and-the-decline-of-cbc-mode-ciphersuites/

[66] ——, "Do the ChaCha: better mobile performance with cryptography," Aug 2018. [Online]. Available: https://blog.cloudflare.com/do-the-chacha-better-mobile-performance-with-cryptography/

[67] "Intel® avx-512 instructions." [Online]. Available: https://www.intel.com/content/www/us/en/developer/articles/technical/intel-avx-512-instructions.html

[68] "Introducing neon for armv8-a." [Online]. Available: https://developer.arm.com/documentation/102474/0100/Fundamentals-of-Armv8-Neon-technology

[69] A. Bogdanov, M. M. Lauridsen, and E. Tischhauser, "Aes-based authenticated encryption modes in parallel high-performance software," *Cryptology ePrint Archive*, 2014.

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

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

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

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

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

[75] J. Bos and M. Stam, *Computational Cryptography: Algorithmic Aspects of Cryptology*. Cambridge University Press, 2021, vol. 469.

[76] E. Barker and Q. Dang, "Recommendation for key management - nist." [Online]. Available: https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-57pt3r1.pdf

[77] H. Tschofenig, M. Pegourie-Gonnard, and H. Vincent, "Performance of state-of-the-art cryptography on arm-based microprocessors." [Online]. Available: https://csrc.nist.gov/csrc/media/events/lightweight-cryptography-workshop-2015/documents/presentations/session7-vincent.pdf

[78] R. Holz, J. Amann, O. Mehani, M. Wachs, and M. A. Kaafar, "Tls in the wild: An internet-wide analysis of tls-based protocols for electronic communication," *arXiv preprint arXiv:1511.00341*, 2015.

[79] J. W. Bos, C. Costello, P. Longa, and M. Naehrig, "Selecting elliptic curves for cryptography: an efficiency and security analysis," *Journal of Cryptographic Engineering*, vol. 6, no. 4, pp. 259–286, 2016.

[80] S. Gueron and V. Krasnov, "Fast prime field elliptic-curve cryptography with 256-bit primes," *Journal of Cryptographic Engineering*, vol. 5, no. 2, pp. 141–151, 2015.

[81] Research and Markets, "Global virtual private network (vpn) market report 2021: Market to reach $107.6 billion by 2027 - covid-19 pandemic provides strong push for vpns," Jul 2021. [Online]. Available: shorturl.at/esBN4

[82] 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.

[83] 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.

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

[85] V. GUEANT, "Iperf - the ultimate speed test tool for tcp, udp and sctptest the limits of your network + internet neutrality test." [Online]. Available: https://iperf.fr/

[86] "8 most common radius mistakes," Jun 2021. [Online]. Available: https://networkradius.com/articles/2021/06/09/8-most-common-RADIUS-mistakes.html

[87] 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.

[88] "Radperf - radius performance testing utility." [Online]. Available: https://networkradius.com/radius-performance-testing/

[89] "Freeradius - fast, feature-rich, modular, and scalable." [Online]. Available: https://freeradius.org/

[90] J. Hassell, "Using udp versus tcp - radius [book]." [Online]. Available: https://www.oreilly.com/library/view/radius/0596003226/ch02s01.html

[91] 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.

[92] "Https encryption on the web." [Online]. Available: https://transparencyreport.google.com/https/overview

[93] H. Lee, D. Kim, and Y. Kwon, "Tls 1.3 in practice: How tls 1.3 contributes to the internet," in *Proceedings of the Web Conference 2021*, 2021, pp. 70–79.

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

[95] [Online]. Available: https://github.com/wg/wrk/releases/tag/4.2.0

[96] 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/

[97] "Crypto acceleration: Enabling a path to the future of computing," Nov 2020. [Online]. Available: https://newsroom.intel.com/articles/crypto-acceleration-enabling-path-future-computing/#gs.xos9e1

[98] N. Heninger, Z. Durumeric, E. Wustrow, and J. A. Halderman, "Mining your ps and qs: Detection of widespread weak keys in network devices," in *21st USENIX Security Symposium (USENIX Security 12)*, 2012, pp. 205–220.

[99] D. He, Z. Deng, Y. Zhang, S. Chan, Y. Cheng, and N. Guizani, "Smart contract vulnerability analysis and security audit," *IEEE Network*, vol. 34, no. 5, pp. 276–282, 2020.

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

[101] K. Team. [Online]. Available: https://keystone-enclave.org/

[102] S. Pinto and N. Santos, "Demystifying arm trustzone: A comprehensive survey," *ACM Computing Surveys (CSUR)*, vol. 51, no. 6, pp. 1–36, 2019.

[103] A. Cabrera, F. Almeida, J. Arteaga, and V. Blanco, "Measuring energy consumption using eml (energy measurement library)," *Computer Science-Research and Development*, vol. 30, no. 2, pp. 135–143, 2015.

[104] S. A. Chowdhury and A. Hindle, "Greenoracle: Estimating software energy consumption with energy measurement corpora," in *2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR)*. IEEE, 2016, pp. 49–60.

[105] M. J. Walker, S. Diestelhorst, A. Hansson, A. K. Das, S. Yang, B. M. Al-Hashimi, and G. V. Merrett, "Accurate and stable run-time power modeling for mobile and embedded cpus," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 1, pp. 106–119, 2016.

[106] M. Chadha, T. Ilsche, M. Bielert, and W. E. Nagel, "A statistical approach to power estimation for x86 processors," in *2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. IEEE, 2017, pp. 1012–1019.

[107] L. Chen, L. Chen, S. Jordan, Y.-K. Liu, D. Moody, R. Peralta, R. Perlner, and D. Smith-Tone, *Report on post-quantum cryptography*. US Department of Commerce, National Institute of Standards and Technology . . . , 2016, vol. 12.

[108] V. Mavroeidis, K. Vishi, M. D. Zych, and A. Jøsang, "The impact of quantum computing on present cryptography," *arXiv preprint arXiv:1804.00200*, 2018.

[109] B. Bathe, R. Anand, and S. Dutta, "Evaluation of grover's algorithm toward quantum cryptanalysis on chacha," *Quantum Information Processing*, vol. 20, no. 12, pp. 1–19, 2021.

[110] D. J. Bernstein and T. Lange, "Post-quantum cryptography," *Nature*, vol. 549, no. 7671, pp. 188–194, 2017.

[111] J. Buchmann, E. Dahmen, and A. Hülsing, "Xmss-a practical forward secure signature scheme based on minimal security assumptions," in *International Workshop on Post-Quantum Cryptography*. Springer, 2011, pp. 117–129.

[112] D. J. Bernstein, D. Hopwood, A. Hülsing, T. Lange, R. Niederhagen, L. Papachristodoulou, M. Schneider, P. Schwabe, and Z. Wilcox-O'Hearn, "Sphincs: practical stateless hash-based signatures," in *Annual international conference on the theory and applications of cryptographic techniques*. Springer, 2015, pp. 368–397.