SecONet: A Security Framework for a Photonic Network-on-Chip

Janibul Bashir
Computer Science and Engineering
IIT Delhi, India
janibbashir@cse.iitd.ac.in

Chandran Goodchild
Embedded Systems
University of Freigburg
ch.goodchild@gmail.com

Smruti Ranjan Sarangi
Computer Science and Engineering
IIT Delhi, India
srsarangi@cse.iitd.ac.in

***Abstract*—Photonic networks are already commercially available at the board-level, and many fabrication facilities can fabricate optical networks and integrate them with traditional silicon-based SoCs. Almost all the research in on-chip photonics has been in the areas of performance enhancement and static power reduction. However, before the large-scale adoption of such technologies, it is necessary to solve security problems. As opposed to electrical NoCs, optical NoCs are shared to a much larger extent, and are significantly more sensitive to the latencies of cryptographic operations. Hence, it is necessary to design a novel protocol for securing such networks. We propose a novel, secure, and efficient optical network in this paper (*SecONet*) that is immune to eavesdropping, spoofing, replay, and message-removal attacks. Using a combination of speculative execution and pre-computation, we reduce the performance overhead of 39.53% with a conventional implementation to 14.2% for a suite of Splash2 and Parsec benchmarks. The additional area overhead of our proposed hardware is modest: 1.6%.**

***Index Terms*—Silicon nanophotonics, Optical networks, Secure NoCs**

I. Introduction

[1] On-chip photonics is expected to prove to be a disruptive technology in the coming decade [6]. Silicon photonics has already come to the PCB. Commercial products that use Photonics are already available for connecting different components on the motherboard and across racks. The consortium for on-board optics (COBO) has already released specifications for photonic on-board communication and several commercial products implement these specifications.

The next step is to take photonics inside the chip. Several test chips have been fabricated by both academia and industry to illustrate the potential of on-chip photonics [4], [7], [18]. Major semiconductor vendors are actively pursuing research in this area, and there are several fabrication companies that can build small-scale optical networks for customers [23]. As of today, such companies can build bespoke optical networks and connectors; these chips can then be integrated on the motherboard, or incorporated in MCM packages, and as a result the era of bringing photonics into the chip is expected to begin.

[1]This work has been sponsored in part by the Semiconductor Research Corporation (SRC) via grant number 2017-TS-2737.

Till now, most of the research in industry and academia has focused on the performance and power consumption aspects of short-range optical networks (on-chip and on-board). Even though these are very important problems, however we highlight the fact in this paper that unless *security concerns* are addressed, it will not be possible for third-party vendors to sell optical communication solutions at a large scale. In today's complex hardware ecosystem, a single product may contain IP blocks, and chips from many different vendors – all communicating on the same network. Ensuring that there are no malicious entities, no hardware Trojans, and the circuits have not been tampered with is a first-class research problem. Given that, till now the designers of optical networks were primarily concerned with the reliability of their components and static power reduction, research on secure optical networks is still in the nascent stage. Other than a recent work, *Soteria*, by Chittamuru et al. [9], to the best of our knowledge, there are no other research papers in this area till date.

Similar to Soteria, we also observe that if optical networks are not secured it is possible for stations (communicating agents) connected on the network to get unauthorized access to data, and launch spoofing or replay attacks. One agent can masquerade as another agent and maliciously change the state of the system. As compared to electrical networks such as on-chip NoCs (Network-on-Chip), or off-chip PCI-X buses, it is significantly easier to mount such attacks in optical communication systems. The reason for this is that most photonic networks are buses that are connected to all the nodes [6], [22]. Other topologies for optical networks such as meshes do exist, but they are rare because it is undesirable to have an intersection between optical channels, and it is not possible to buffer an optical message without doing an optical-to-electrical conversion [6]. Now, given that all the stations are connected to the same bus, it is very easy for one station to read a message destined for another station, and it is also very easy to insert a fake message and maliciously alter the state of the system.

In this space, there are several trust models: we either trust the optical network but not the connecting stations, or we do not trust anybody. The traditional method used to solve such problems is using cryptography-based encryption techniques. Since these are very short-range links, naive implementations are typically very slow. If we consider an on-chip commu-

nication system, the 25-cycle encryption/decryption latency overshadows the 2 to 3 cycle communication latency by a large margin. Hence, using traditional cryptography has been precluded in most practical on-chip networks.

We observed that by leveraging the idiosyncrasies of optical networks, we can design a secure protocol that need not be constrained by the latency of cryptographic algorithms. The crucial feature that we leverage is that between a sender-receive pair, we have first-in-first-out (FIFO) communication. This allows us to run two synchronized state machines at both ends unbeknownst to the rest of the receivers. As a result, the sender can send a message encrypted with some bits from its current state, and since the receiver has the same state, it can decrypt it. Additionally, this method can be extended to ensure that there are no spoofing attacks (message maliciously modified by others).

The FIFO property also allows us to decrease the latency by enabling us to pre-compute encryption and decryption keys, save them in a cache, and later use them. As long as we have enough keys, there is no delay. We combine all of these ideas to propose a security protocol that records information in private registers in the sender and the receiver, and periodically exchanges this information to verify if there are any security breaches.

The contributions of this paper are as follows:

❶ Designed a full end-to-end security protocol for on-chip optical networks that is immune to replay, eavesdropping, spoofing, and message-removal attacks.

❷ Leveraged the properties of optical networks to propose a scheme to nullify the effect of the long latency of cryptographic operations.

❸ Implemented the system using VHDL and simulated the same using the Cadence RTL simulator.

❹ Demonstrated a meager 14.2% slowdown as compared to a system with no security.

II. Background and Related Work

A. Basic On-chip Optical Communication Framework

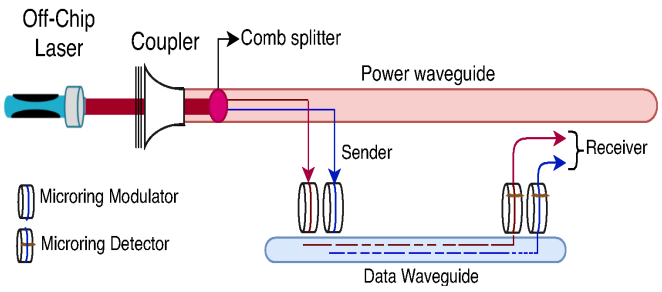

Fig. 1: Photonic on-chip communication framework

Figure 1 shows the essential components required in implementing a photonic on-chip network. An external directly modulated laser source [6] is used to generate an optical signal (at 1550 nm), which is brought on-chip via special tapered couplers. Optical power is routed to the different optical stations (transmitter+receiver) via a set of power waveguides (power in each waveguide is less than the non-linear limit). Each optical station uses a comb splitter [22] to generate 64 equispaced wavelengths. If a station wishes to transmit data it couples these wavelengths to dedicated data waveguides (connected to a subset of the rest of the stations). To modulate optical power based on electrical signals, we use microring resonators (MRs). The 64 wavelengths are multiplexed into the same optical channel (waveguide), and at the receiver's side a set of ring resonators filter out the wavelengths, and then the optical signals are converted to equivalent electrical signals using an array of photodetectors [16].

B. Optical Waveguides

The main advantages of optical communication are as follows: very fast propagation latency (2-3 cycles between any two points on the chip with a 2 GHz clock), low-power communication (roughly independent of the distance), and the feasibility of dense wavelength division multiplexing (64 per waveguide). On the flip side, it is hard to create a complex topology such as a mesh because waveguide crossings dissipate a lot of power. Hence most optical networks are serpentine-shaped buses [6].

We can connect the optical stations in primarily three different configurations: Single-Writer Multiple-Reader(SWMR), Multiple-Writer Single-Reader(MWSR), and Multiple-Writer Multiple-Reader(MWMR). The SWMR configuration has one waveguide per sender. The sender first sends a message to wake up a set of receivers, then it sends the message; each receiver on the way splits a part of the optical power using a tunable splitter, and subsequently reads the message. In comparison, in a MWSR waveguide, each receiver has its dedicated waveguide that is connected to the rest of the senders. The senders need to arbitrate among themselves, and then the winner gets to send data to the receiver. MWMR is a combination of MWSR and SWMR [6].

Whenever an optical network is sold as an IP block or a separate add-on component, it needs to be *generic*, which means that it should support the creation of all kinds of buses. The bus protocol needs to be realized by the electrical components that are connected to it. This is where the security flaws arise because such networks implicitly rely on the trustworthiness of all the senders and receivers, which may not be the case. Let us next look at some attack scenarios.

C. Microring tunning

The resonant wavelength of a microring resonator (MR) is highly sensitive to the thermal variations [11], [14], [20] inside the chip. In addition, the process variations induce the deviations in the dimensions (height, width, and thickness) of the MRs, resulting in variation in the effective refractive index of the ring resonators, thereby deviating them from their nominal resonant wavelength [19], [27]. In order to mitigate the effects of temperature and fabrication induced variations in microring resonators, it is necessary to tune them accordingly [20]. This tuning is usually done by changing the effective refractive index of the MRs that is carried out by applying external

current or by applying heat (thermal tuning) [20] (with the help of integrated microring heaters) [21].

In PNoCs, there is a separate tuning circuit that is responsible for controlling the electrical and thermal tuning of the microring resonators. The same circuit is also required to turn the microring resonators on and off based on the requirement. This tuning circuit is the most vulnerable component inside the optical station, in presence of intelligent Hardware Trojans (HT). The HT can get the control of this tuning circuit and can manipulate the resonant wavelength of ring resonators spuriously. Thus, making the on-chip communication insecure in an otherwise secure system.

D. Network Attacks

Some of the important attacks that may affect the photonic on-chip network are as follows:

1) **Eavesdropping:** Attackers capture the information from the optical waveguides and then easily read the sensitive data.
2) **Impersonation(spoofing):** Attacker acts as some another entity and may easily get access to access-restricted data.
3) **Replay Attack:** The attacker sits in the middle (between the sender and receiver) and replays an earlier message sent between the same pair of stations. The receiver thinks it is a valid message even if it is encrypted.
4) **Integrity Attacks:** The attacker may remove data packets from the bus, maliciously modify them, and then send them to the receiver.

III. Motivation and Threat Model

A. Domain of Trust

In a given SoC, we predominantly have two kinds of actors: the NoC itself, and the components that are connected to it. Different proposals in this domain have considered different trust models for the connected stations and the NoC. In this paper, we consider a generic scenario where we mistrust the NoC as well as the rest of the stations other than the sender and receiver (for a given data transfer). This means that it is possible for the rest of the stations to collude and either eavesdrop on the message (read it clandestinely), maliciously modify it or even create fake messages. Along with defining the attack surface, we also need to define the components that we trust. In our paper, we make the following assumptions (standard in the literature):

1) The foundry is trustworthy ([15]).
2) The silicon integrator is trustworthy, which means that she does not change the design of any of the components that are given to her. Note that the components themselves (inclusive of the NOC) can be malicious unbeknownst to the integrator and the foundry.
3) The clock, power, and ground lines are secure and not tampered with.
4) There is a mechanism to report any breach of security to either software or remote hardware.

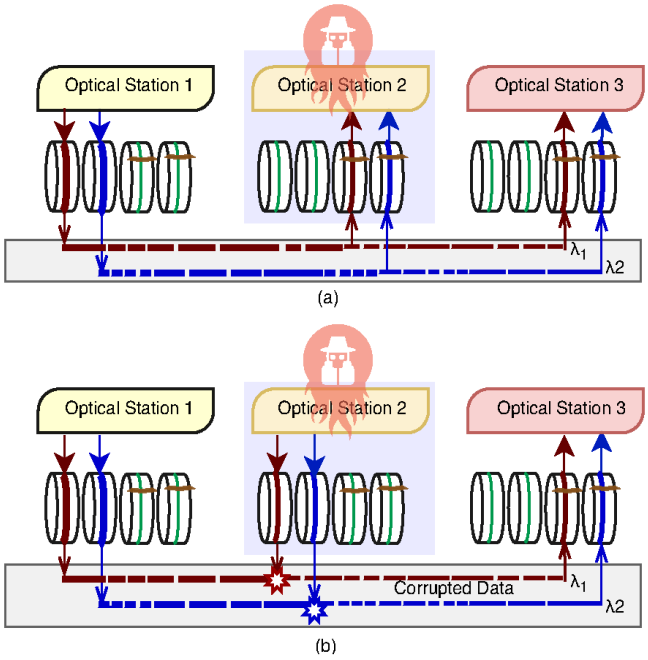

Fig. 2: Attacks on the PNoCs (photonic NoCs)

B. Attacks on an SWMR Network

Consider an SWMR network in Figure 2(a). It shows a scenario where stations 2 and 3 are allowed to read data from the same channel whereas station 1 is allowed to write. In this scenario, we have assumed that station 2 is malicious due to the presence of a hardware Trojan (HT). Now suppose station 1 sends some data to station 3. In normal execution, station 2 should keep its detectors in the off-state. However, due to the presence of the HT in its circuitry, the HT can tune the detectors of station 2 to the wavelengths passing through the waveguide. Thus, it can snoop the data intended for station 3. The malicious station can forward the snooped data to a malicious core attached to it, resulting in the leakage of sensitive information.

C. Attacks on an MWSR Network

In the multiple writers based MWSR configuration, multiple stations are allowed to write on the same data channel. Figure 2(b) shows a scenario where multiple stations (1 and 2) are allowed to write data on the same channel, whereas only station 3 is allowed to read. Let us assume that station 1 wants to send data to station 3, and station 2 is compromised due to the presence of a hardware Trojan (HT) in its circuitry. This HT can easily manipulate the tuning circuit of the MRs attached to station 2 and tune them to the data-carrying wavelengths. Since station 2 is also allowed to write its data, it can change the contents of the message. This compromises the *integrity* of the original message.

Additionally, the hardware Trojan with the help of malicious MRs can easily manipulate the source address of the flits, resulting in a *spoofing* attack. The spoofing attack enables the malicious station to masquerade as some other optical station.

Moreover, spoofing can be employed by a malicious station to respond to some legitimate requests, which are otherwise intended for other stations. These can lead to Byzantine failures.

IV. Design and Implementation

A. Overview

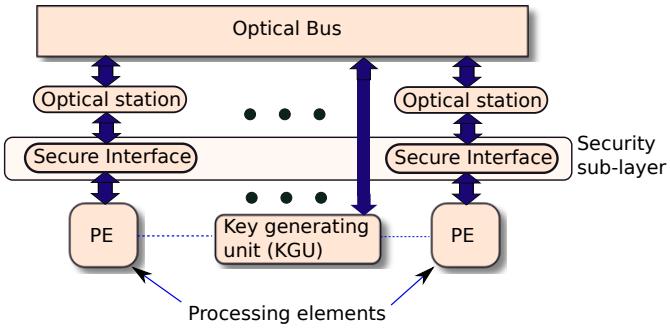

Fig. 3: Overview of the system

Figure 3 shows an overview of the system. We refer to any entity that can send a message on the bus as a processing entity (PE); this can either be a core, a cache, or a memory controller. In state-of-the-art designs the PE is directly connected to the optical station, which can send or receive messages on its behalf [6]. In our design, we add an additional security sub-layer between the PE and the optical station. This layer contains the logic (Secure Interface: **SI**) to ensure the following security properties: confidentiality (not possible for a third party to retrieve the original contents of the message), integrity (not possible to change the contents of the message without getting detected), guaranteed delivery (not possible to remove messages from the network without getting detected), and timeliness (not possible for a third-party to remove a message from the network and reinsert it back with an indefinite delay). Our design, or for that matter any design of a secure network, needs to ensure these four properties.

The operation of our protocol has three phases: ❶ key distribution, ❷ regular execution, and ❸ verification. The latter two phases keep repeating, whereas key distribution is done at boot time. The job of key distribution is done by a dedicated Key Generating Unit (KGU), whose job is to distribute unique cryptographic keys to each pair of optical stations. Each pair of optical stations, then use this key to securely communicate among themselves. However, it is still possible for a third-party station to tamper with the contents of the message, or remove it altogether from the optical network.

We split the work of verifying the contents and the delivery of flits into two parts: integrity checking and flit-count verification. Integrity checking continues in parallel with execution, thus it is a part of the execution phase, whereas verifying that no flits have been dropped, no extra flits have been added, or delayed indefinitely needs to be done periodically (flit-count verification phase).

B. Key Distribution Phase

For a network with n SIs, the aim of this phase is to compute $\binom{n}{2}$ keys so that any two secure interfaces, i and j, can use a dedicated key, K_{ij}, to communicate among themselves. The keys need to be generated and securely distributed without any loss in confidentiality or integrity.

We assume the presence of a specialized hardware block for the key generation unit (KGU) that is assumed to execute correctly. Let us first discuss a trivial solution where we assume a dedicated yet slow set of electrical links connecting each SI to the KGU. In this case each SI simply sends a "hello" message at boot time, and gets a list of $n-1$ keys in the response. We assume that the KGU has a pseudorandom number generator, which is initialized with a *truly random seed* (use techniques in [12] or use the current time) to generate $\binom{n}{2}$ 128-bit encryption keys.

However, if we do not have a secure electrical network that can be used for this purpose, then we need to use the insecure optical network and use public-key cryptography. The standard solution based on the RSA algorithm is not area-efficient because we need to have one such unit with every SI. We instead use the BlueJay asymmetric cipher [24], whose area is limited to 3000 gate equivalents (one 2-input NAND gate is one gate equivalent). In terms of the security level, it is similar to competing cryptographic mechanisms, and the latency is also competitive. Furthermore, note that this process needs to be done only once at boot time; the long encryption/decryption times (typically a million cycles) is not of particular concern.

Let us outline a simple algorithm that can be used to distribute the keys. Let us introduce the terminology used to refer to the keys as given in Figure 4. Each component c (SI and the KGU) has a public key (P_c) and a private key (X_c), which is stored in its ROM. The assumption is that the system integrator or the foundry will not steal them. We can also break this assumption if we embed a key generator in each SI. We assume that the KGU's public key is known to all the stations. The encryption of a message m with key k is denoted by $E(k, \langle m \rangle)$.

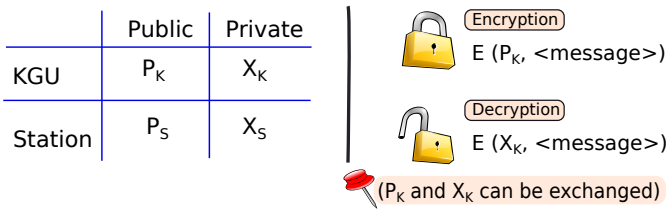

Fig. 4: Encryption and decryption

The protocol is shown in Figure 5. Note that all such messages require double encryption. The first encryption is required to ensure that either the KGU can only decrypt the message or the recipient SI is sure that the message is being sent by the KGU. The second encryption is required such that the KGU is sure that the origin of the message is the SI whose id is there in the message, or only the recipient SI can read the contents of the message. Note that in all such cases there has

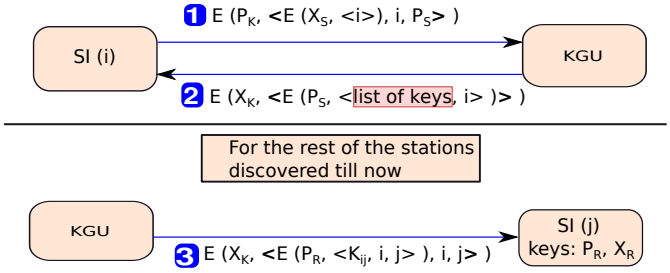

Fig. 5: The key distribution protocol

to be some part of the message that is known to the recipient (eg: i in message ❷) such that it can verify that no tampering has been done. Tampering even a single bit of an encrypted message leads to a large number of bit flips in the decrypted message

At the outset each SI, i, introduces itself to the KGU by sending a message (message ❶). Once the KGU is sure of the origin of the message and its contents, it generates a pairwise communication key of the form K_{ij}, where j is the id of each SI that is known to the KGU (introduced itself before). The list of such keys is sent to the SI, i. Finally, in round ❸, the KGU sends the new key, K_{ij} (called startup key, SK), to each of the SIs (j) that it knows.

C. Execution Phase

Let us start with the main principles. The sending SI needs to encrypt the message, and the receiving SI needs to decrypt the message. Sadly, these operations are rather slow in modern processors and they take upwards of 20-25 cycles each. Given that the latency of the optical link including E/O and O/E conversions does not exceed 4-5 cycles, this is a *prohibitive delay*. Hence, we propose to precompute an encrypted 128-bit number called the one-time pad (OTP) (stored in the OTP buffer). We replicate this number a suitable number of times till the length of the pad is equal to the length of the message. If the message to be sent is $\mathcal{M}$, and the replicated pad is $\mathcal{O}$, then the encrypted message is $\mathcal{M} \oplus \mathcal{O}$ ($\oplus$ stands for XOR). A XOR operation is very fast and can be computed in a single cycle. The receiver will generate the same OTP (mechanism described in Section IV-C1). It will recover the contents of the message by computing a XOR with the OTP again. In this case, if the rate of OTP generation is equal to the rate of message transfer, and we can afford to have a long buffer of precomputed OTPs then we shall have no delays.

Even though we have solved the problem of maintaining confidentiality we need to ensure the following: ❶ the sender and the receiver always have the same value of the pad without actually communicating it (Section IV-C1), ❷ we maintain message integrity, which means that if the messages are tampered with, it can be detected (Section IV-C2).

1) Maintaining Confidentiality: In this section we need to create a mechanism to ensure that the sender and the receiver use the same value of the OTP, and computing it does not *delay* the message transmission process.

Let K_{ij} be the key obtained from the KGU. With no loss of generality assume $i < j$. For $i \rightarrow j$ communication let the *major key* be equal to K_{ij} with its MSB set to 1. For communication in the reverse direction let the major key be K_{ij} with its MSB set to 0. Now, let us consider a 16-bit *minor key* (initialized to 0, stored in the minor counter table: MCT). Both the sender and the receiver are aware of their major and minor keys for a given direction of communication.

Every time we send a message, we increment the minor key by 1. Once it reaches the maximum value ($2^{16} - 1$), we set it to 0, and increment the major key. The *complete key* is a concatenation of the major key and the minor key. This process ensures that the complete key is never repeated in any practical run. Now, to generate the OTP we replicate the complete key and encrypt a piece of dummy data (128 bits) using an encryption engine (EE). The output is only dependent on the complete key; this is the *OTP*. The advantages of this scheme are that several OTPs can be generated in parallel, and because of the FIFO property of the optical channel, the sender and receiver are always synchronized (will always compute the same complete key).

Now to ensure that we always have enough complete keys available we need to have a buffer of keys. We define two thresholds: a *force threshold (α)*, and a *slowdown threshold (β)*. Whenever, the number of keys falls below α, we start generating keys at the maximum possible rate till we have β keys, and then we reduce the rate of generation of keys till the buffer fills up.

2) Integrity Checking: The standard approach for integrity checking is that we compute a hash of the message, and send the hash along with the message. The receiver decrypts the received bytes, recomputes the hash and checks if the hashes match. However, here this cannot be done because the typical latency to compute a hash even with aggressive hardware is more than 10 cycles. This delay is prohibitive.

Hence, we propose an ingenious speculative method. We send a message, and in parallel we start computing the hash at the sender's side. Once it is computed, we **piggyback** the computed hash (only 64 LSBs) with the next message that is sent. This is also done in FIFO order. The receiver maintains a queue of messages whose hashes have not been verified, and verifies them as and when it gets the piggybacked hash values. This allows us to completely *disregard* the latency of hash computation. Meanwhile the receiver can process the message, however it cannot commit any instruction that uses an unverified value. Since the time from load to commit is the order of tens of cycles in modern OOO pipelines, this issue is not of significant concern. If the PE is a cache, it can treat the buffer of unverified entries as a victim buffer. We have a timer with the sender that sends the hash value after waiting for γ cycles (if it does not have a message to send).

Figure 6 summarizes the design and the flow of operations.

D. Verification Phase

In this phase, we verify that no message has been dropped or delayed indefinitely. Most of this can be verified using

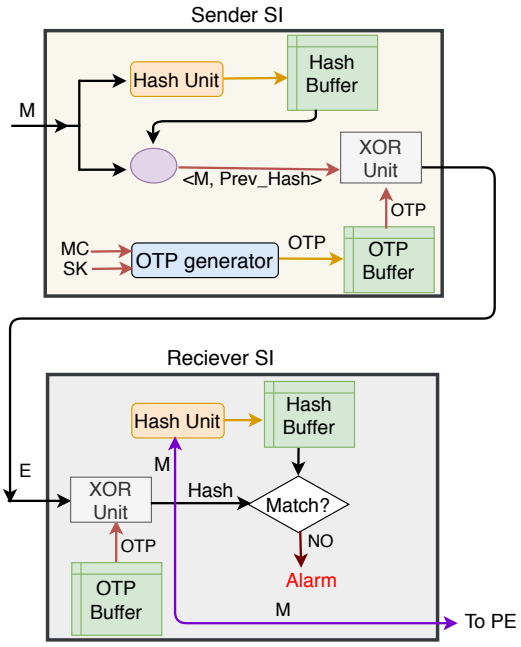

Fig. 6: Complete design of SecONet

the integrity checking mechanism, however if some node is absorbing all the messages then it cannot be done. Here again, the FIFO property is useful. We use a challenge-response mechanism. Every η cycles, i sends a message to j, with its major and minor counters (encrypted as a regular message). If both the counters match, j affixes a known message with the counters, and sends them back (response). If i does not get a response within κ cycles, we can infer a security violation.

The optimal values for $\alpha, \beta, \gamma, \eta$, and κ are determined based on simulation results (see Section V).

V. Evaluation

A. Setup

We integrated our proposed security scheme, *SecONet*, into two state-of-the-art optical networks, *ColdBus* [22] and *Probe* [28], and characterized the impact of our scheme on the performance of these networks. In both these schemes, we assume a 16-station optical network, that comprise 16 optical channels to carry data. We implemented each topology in a cycle-accurate architectural simulator, *Tejas* [25] and evaluated the performance and power overheads associated with our scheme. In addition, we implemented the circuit in VHDL and simulated the same using the Cadence Encounter tool to find out the area and delay overheads.

We simulate benchmarks from the PARSEC [8] and Splash2 [26] suites. For the optical parameters we have assumed standard values used in prior work [5], [6], [22] as given in Table I. They have been derived from real implementations.

B. Parameter Selection

We analyzed the effect of varying the number of one-time pad (OTP) generation units and hash units on the overall performance of the system (see Figure 7). We observe that

Parameter	Value
Component configuration	
Wavelength	1550nm
Waveguide Width	$0.5\mu m$
Waveguide Thickness	$0.2\mu m$
Photodetector power	$36\mu W$
Input Driver Power	$76\mu W$
Propagation Delay	7ps/mm
Optical Loss	
Coupling Loss	50%
Waveguide Loss	0.5dB/cm
Bending Loss	1dB
Splitter Loss	0.36dB
Photodetector Loss	0.1dB
Crossing Loss	0.05dB
Coupler loss	1dB
Off-Ring Loss	0.001dB

TABLE I: Optical Parameters [6], [22]

increasing the number of such units increases the overall performance of the system but at the cost of more area till we start to see *diminishing returns*. For an optimal configuration, we choose 4 hash units and 8 OTP units per station in our final design. Subsequently, we conducted exhaustive simulations by varying the different thresholds and constants used in our design. The best results in terms of performance were obtained for $\alpha = 4$, $\beta = 10$, $\gamma = 15$, $\eta = 5000$, and $\kappa = 50$.

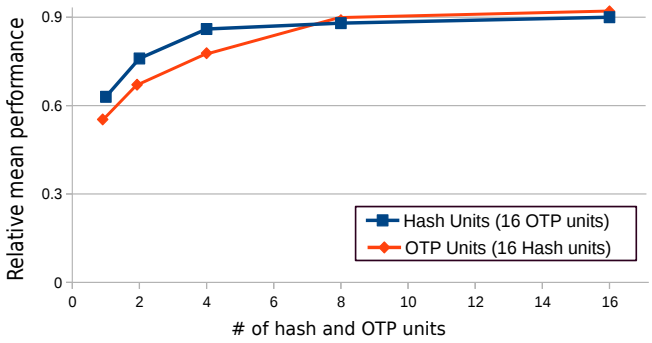

Fig. 7: Effect of hash and OTP units on the overall system performance

Module	Area	Delay (cycles)
BlueJay Unit [24]	$1028\mu m^2$	312000
Whirlpool Unit [2]	$48912\mu m^2$	20
Prince Unit [17]	$10214\mu m^2$	16
XoR Unit	$0.008mm^2$	2
OTP Buffer (per SI)	$0.019mm^2$	-
Minor Counter Table (per SI)	$0.01mm^2$	-
Total area per SI	$0.315mm^2$	-

TABLE II: Area and delay overheads (duly scaled to 14nm [13])

C. Area and Delay Overheads

The main contributors to the area and delay overheads are the *Whirlpool Units* (for hash calculation), *BlueJay units* (for key exchange), *Prince units* (for one-time pad generation),

OTP buffer, and MC table. To calculate the area and delay overheads, we implemented the individual components as well as the complete circuit using VHDL and synthesized the same using the Cadence RTL compiler for a 14nm technology node. In addition, we used the Cacti 6.5 [1] tool to calculate the area associated with the OTP buffer and MC table. Table II shows the area and delay associated with each component required for implementing *SecONet*. The total additional area is $0.315mm^2$ per station. Thus, for a 16-station PNoC (with 16 stations and 1 KGU), the area overhead is 1.6% ($6.32mm^2$) ($400mm^2$ die).

D. Effect of SecONet on Mean Packet Latency, Performance, Power Consumption and Energy-Delay-Square product

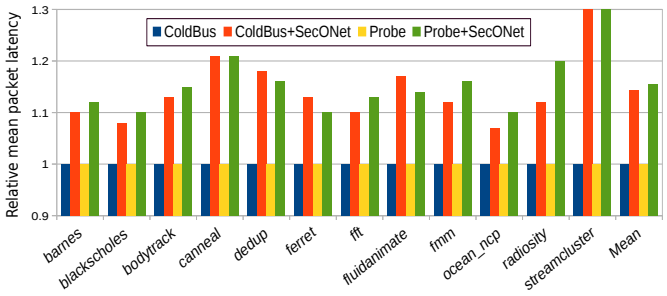

Fig. 8: Effect of SecONet on packet latency

Figure 8 shows the effect of *SecONet* on the mean packet latency. The latency results are normalized to the base designs (no security protocol). It is clear from the graph that *ColdBus+SecONet* has 13.6%, and *Probe+SecONet* has 15.8%, higher mean packet latencies as compared to their baseline designs. It is because of the additional delays associated with our scheme. However, by bringing expensive operations, involved with our scheme, out of the critical path, we were able to maintain the overheads at an acceptable level. Note that without our optimizations, the mean packet latency was 45.12% higher than the baseline design.

A similar trend is observed in the case of the overall performance of the system, as shown in Figure 9. Due to an increase in the mean packet latency, *SecONet* resulted in a 11.54% and 14.2% decrease in the mean performance of *ColdBus* and *Probe* respectively. With a conventional implementation the performance reduction was 39.53%.

E. Analysis

Based on our analysis, these are the major reasons for a reduction in performance: ❶ additional cycles required for the XOR operation at the sender and at the receiver, ❷ messages waiting for an OTP (OTP buffer is empty), and ❸ instructions waiting at the commit stage for the verification of the corresponding hashes. We observed that nearly 72.32% of the slowdown is attributable to reason ❶. The remaining slowdown is mainly because of reasons ❷ and ❸. On an average, we observed that less than 0.09% of memory instructions wait at the commit stage for the hash-verification of their data, thus,

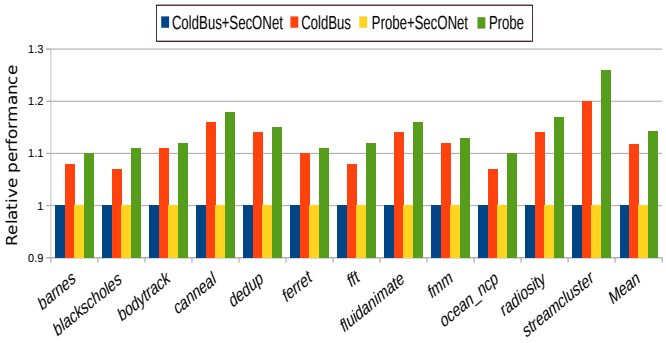

Fig. 9: Effect of SecONet on the performance

accounting for less than 9.7% of the slowdown. Additionally, in the entire chip nearly 0.11% of messages have to wait for their OTP to get generated (OTP buffer empty), resulting in a slowdown of 11.3% (of the net slowdown).

However, in some benchmarks, such as *canneal* and *streamcluster*, reasons ❶ and ❷ are disproportionate contributors. It is because of high traffic generation in these benchmarks (30-70 messages per 1000 cycles). Moreover, we observed that a majority of this traffic is destined to the same receiver, which results in more cases where the OTP buffer is found to be empty.

Next, we analyze the effect of *SecONet* on the overall laser power consumption of the system. From the results for power consumption shown in Figure 10, it is clear that *ColdBus+SecONet* and *Probe+SecONet* resulted in 20.2% and 24.6% increase in overall laser power consumption, respectively. The main reason is the increase in overall message size. In our scheme, we are increasing the size of the messages by including the hash of the previous message, thus requiring more power to send the message, resulting in an increase in overall power consumption.

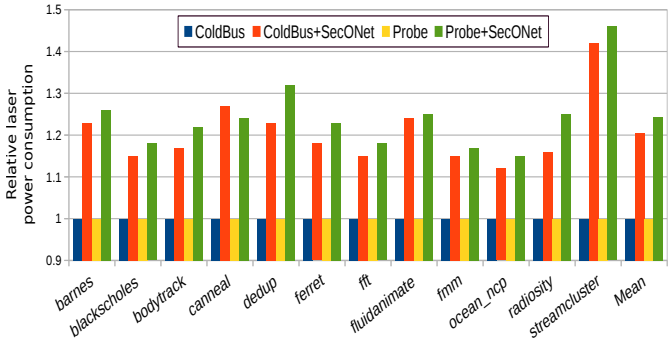

Fig. 10: Effect of SecONet on laser power consumption

Finally, we showed the effect of *SecONet* on the energy-delay (ED) product. It is the standard metric used to depict the efficacy of any scheme. Here energy refers to the energy of the overall system. The results are given in Figure 11. It is clear from the graph that our scheme resulted in 20% and 27% increase in ED values in *ColdBus* and *Probe* respectively. The main reason is the decrease in system performance and

increase in laser power consumption. With a conventional approach there is nearly 82% increase in ED values.

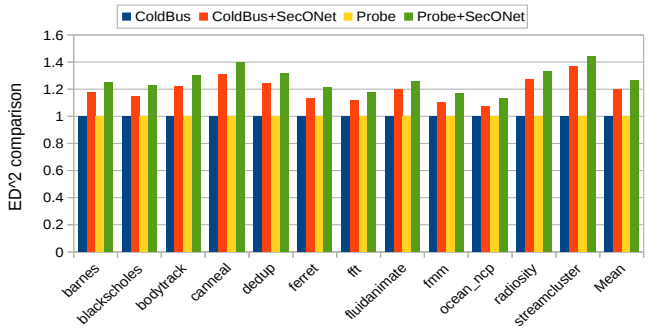

Fig. 11: Effect of SecONet on ED values

VI. Related Work on Photonic NoC (PNoC) Security

1) Electrical Networks: Securing NoCs has gained a lot of research interest in the last few years [10]. The reason is the increasing prevalence of using third-party IPs and even third-party NoCs. Several prior works have tried to provide integrity, authenticity and confidentiality in on-chip networks [3], [10]. These solutions take more than 10-15 cycles, which is not significant in a large electrical NoC, where the end-to-end delay can be as high as 50 cycles. However, such delays are unacceptable in photonic NoCs, and their impact on the message latency is significant. Hence, we need a bespoke protocol where the latency of ensuring confidentiality and integrity is mostly hidden, which is what we have done in this paper.

2) Optical Networks: To the best of our knowledge, the solution proposed by Chittamuru et al. [9] is the only work in the domain of secure PNoCs. However, it has several issues: ❶ it does not guarantee the integrity and authenticity of the messages, ❷ it uses predefined keys to encrypt the messages and even the malicious node posses the correct key, and ❸ it uses a XOR operation to encrypt the data without changing the key for every message (as we do). Our protocol is thus far more secure.

VII. Conclusion

In this paper, we presented a full end-to-end security protocol for on-chip optical networks. We leveraged some properties of optical networks to bring the expensive operations, involved in our scheme, out of the critical path in order to limit the overheads associated with our scheme. By incorporating our proposed protocol in state-of-the-art optical topologies, we demonstrated that our scheme has minimal area (1.6%) and performance (14.2%) overheads. Without any optimizations the performance overheads would have been 39.53%. Thus, our protocol proves to be an attractive solution to provide security in on-chip photonic networks.

References

[1] "Cacti 6.5," https://www.hpl.hp.com/research/cacti/.

[2] Akashi Satoh, "Asic hardware implementations for 512-bit hash function whirlpool," in *2008 IEEE ISCAS*, May.

[3] D. M. Ancajas, K. Chakraborty, and S. Roy, "Fort-nocs: Mitigating the threat of a compromised noc," in *DAC*, 2014.

[4] J. Bashir, E. Peter, and S. R. Sarangi, "Bigbus: A scalable optical interconnect," *J. Emerg. Technol. Comput. Syst.*, 2019.

[5] J. Bashir and S. R. Sarangi, "Nuplet: A photonics based multi-chip nuca architecture," in *ICCD*, 2017.

[6] J. Bashir, E. Peter, and S. R. Sarangi, "A survey of on-chip optical interconnects," *ACM Comput. Surv.*, February 2019.

[7] J. Bashir and S. R. Sarangi, "Predict, share, and recycle your way to low-power nanophotonic networks," *ACM JETC*, 2019.

[8] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The PARSEC benchmark suite: characterization and architectural implications," in *PACT*, 2008.

[9] S. V. R. Chittamuru, I. G. Thakkar, V. Bhat, and S. Pasricha, "Soteria: Exploiting process variations to enhance hardware security with photonic noc architectures," in *IEEE DAC*, 2018.

[10] C. H. Gebotys and R. J. Gebotys, "A framework for security on noc technologies," in *VLSI*, 2003, pp. 113–117.

[11] R. R. Ghosh, J. Bashir, S. R. Sarangi, and A. Dhawan, "Spliesr: Tunable power splitter based on an electro-optic slotted ring resonator," *Optics Communications*, 2019.

[12] J. Holleman, S. Bridges, B. P. Otis, and C. Diorio, "A 3 μw cmos true random number generator with adaptive floating-gate offset cancellation," *IEEE Journal of Solid-State Circuits*.

[13] W. Huang, K. Rajamani, M. Stan, and K. Skadron, "Scaling with design constraints: Predicting the future of big chips," *Micro, IEEE*, 2011.

[14] M. Lipson, "Guiding, modulating, and emitting light on silicon-challenges and opportunities," *Journal of Lightwave Technology*, vol. 23, Dec 2005.

[15] Y. Liu, C. Bao, Y. Xie, and A. Srivastava, "Introducing tfue: The trusted foundry and untrusted employee model in ic supply chain security," in *2017 IEEE ISCAS*, May.

[16] J. Michel, J. Liu, and L. C. Kimerling, "High-performance ge-on-si photodetectors," *Nature photonics*, vol. 4, no. 8, p. 527, 2010.

[17] N. Miura, K. Matsuda, M. Nagata, S. Bhasin, V. Yli-Mayry, N. Homma, Y. Mathieu, T. Graba, and J.-L. Danger, "A 2.5 ns-latency 0.39 pj/b 289μm 2/gb/s ultra-light-weight prince cryptographic processor," in *2017 Symposium on VLSI Circuits*. IEEE.

[18] B. News, "Engineers demo first processor that uses light for ultra-fast communications," *http://news.berkeley.edu/2015/12/23/electronic-photonic-microprocessor-chip/*, 2015.

[19] M. Nikdast, G. Nicolescu, J. Trajkovic, and O. Liboiron-Ladouceur, "Modeling fabrication non-uniformity in chip-scale silicon photonic interconnects," in *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2016.

[20] C. Nitta, M. Farrens, and V. Akella, "Addressing system-level trimming issues in on-chip nanophotonic networks," in *HPCA*, Feb 2011.

[21] K. Padmaraju and K. Bergman, "Resolving the thermal challenges for silicon microring resonator devices," *Nanophotonics*, vol. 3, no. 4-5, pp. 269–281, 2014.

[22] E. Peter, A. Thomas, A. Dhawan, and S. R. Sarangi, "Coldbus: A near-optimal power efficient optical bus," in *HiPC*, 2015.

[23] C. M. Projects, "Silicon photonics," *https://mycmp.fr/datasheet/silicon-photonic-ics-si310-phmp2m*, 2018.

[24] M.-J. O. Saarinen, "The bluejay ultra-lightweight hybrid cryptosystem," in *IEEE Symposium on Security and Privacy Workshops*, 2012.

[25] S. R. Sarangi, R. Kalayappan, P. Kallurkar, S. Goel, and E. Peter, "Tejas: A java based versatile micro-architectural simulator,," in *PATMOS*, 2015.

[26] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, "The splash-2 programs: characterization and methodological considerations," *SIGARCH Comput. Archit. News*, vol. 23, pp. 24–36, May 1995.

[27] Y. Xu, J. Yang, and R. Melhem, "Tolerating process variations in nanophotonic on-chip networks," in *ACM SIGARCH Computer Architecture News*, vol. 40, no. 3. IEEE Computer Society, 2012, pp. 142–152.

[28] L. Zhou and A. K. Kodi, "Probe: Prediction-based optical bandwidth scaling for energy-efficient nocs," in *NOCS*, 2013.