Securing Network-on-chips Against Fault-injection and Crypto-analysis Attacks via Stochastic Anonymous Routing

AHMAD PATOOGHY and MAHDI HASANZADEH, North Carolina A&T State University, USA
AMIN SARIHI, New Mexico State University, USA
MOSTAFA ABDELREHIM, California State University, USA
ABDEL-HAMEED A. BADAWY, New Mexico State University, USA

Network-on-chip (NoC) is widely used as an efficient communication architecture in multi-core and many-core System-on-chips (SoCs). However, the shared communication resources in an NoC platform, e.g., channels, buffers, and routers, might be used to conduct attacks compromising the security of NoC-based SoCs. Most of the proposed encryption-based protection methods in the literature require leaving some parts of the packet unencrypted to allow the routers to process/forward packets accordingly. This reveals the source/destination information of the packet to malicious routers, which can be exploited in various attacks. For the first time, we propose the idea of secure, anonymous routing with minimal hardware overhead to encrypt the entire packet while exchanging secure information over the network. We have designed and implemented a new NoC architecture that works with encrypted addresses. The proposed method can manage malicious and benign failures at NoC channels and buffers by bypassing failed components with a situation-driven stochastic path diversification approach. Hardware evaluations show that the proposed security solution combats the security threats at the affordable cost of 1.5% area and 20% power overheads chip-wide.

CCS Concepts: • **Security and privacy → Side-channel analysis and countermeasures;**

Additional Key Words and Phrases: Network-on-chips, System-on-chips, security, hardware Trojan, anonymous routing, source routing, stochastic routing

ACM Reference format:
Ahmad Patooghy, Mahdi Hasanzadeh, Amin Sarihi, Mostafa Abdelrehim, and Abdel-Hameed A. Badawy. 2023. Securing Network-on-chips Against Fault-injection and Crypto-analysis Attacks via Stochastic Anonymous Routing. *ACM J. Emerg. Technol. Comput. Syst.* 19, 3, Article 22 (June 2023), 21 pages.
https://doi.org/10.1145/3592798

Authors' addresses: A. Patooghy and M. Hasanzadeh, North Carolina A&T State University, Greensboro, NC 27411, USA; emails: apatooghy@ncat.edu, mhasanzadehhesar@aggies.ncat.edu; A. Sarihi and A.-H. A. Badawy, New Mexico State University, Las Cruces, NM 88001, USA; emails: {sarihi, badawy}@nmsu.edu; M. Abdelrehim, California State University, Bakersfield, CA 93311, USA; email: mabdelrehim@csub.edu.

1550-4832/2023/06-ART22 $15.00
https://doi.org/10.1145/3592798

1 INTRODUCTION

Today's **integrated circuit (IC)** fabrication paradigm allows multiple companies to contribute to the design, integration, fabrication, testing, and packaging of a chip [22]. In this process, **third-party intellectual properties (3PIPs)** are widely used to reduce the design cost and minimize time to market [3, 22]. **Multiprocessor system-on-chip (MPSoC)** is one case in which different providers develop 3PIP processing cores, memory modules, and I/Os, often integrated to construct the final design. Although outsourcing reduces the design/manufacturing time and cost, it is vulnerable to new security risks, such as **Hardware Trojans (HTs)**. An HT is any unwanted modifications or functionalities trying to achieve or facilitate malicious goals in the final product [37].

As most of the modern MPSoCs use an on-chip network at their backbone communication architecture, the industry has already started offering **Network-on-chip (NoC)** IPs (for instance, FlexNoC IP from Arteris company [1, 20]). The main idea of on-chip networks, resource sharing to boost resource utilization, would amplify the security challenges of modern MPSoCs. More specifically, several on-chip routers contribute to forwarding a packet that might not be directly related to them. The benefit gained by resource sharing in NoC fabrics may be counterproductive when data integrity and confidentiality are considered.

As NoC routers demand easy access to the source/destination addresses in the header flits, the application of data encryption methods is only limited to the data flits of packets. Indeed, the encryption of the header information will adversely impact the performance of the MPSoC as each router will have to decrypt, process, and encrypt the packet's header.[1] As a result, a wide range of security attacks that depend on the source/destination pairs of sensitive packets may still be capable of stealing sensitive data and exacerbating security. This emphasizes the need to consider security in the design of the communication fabric of MPSoCs. Crypto-analysis attack is one example in which a malicious router forwards packets to a malicious core to conduct a mathematical analysis aiming to discover the secret data carried by packets [30]. The effectiveness of crypto-analysis attacks can be boosted while accompanied by fault-injection attacks. An adversary makes intentional failure/congestion at some channels/buffers of the NoC to guide their desired traffic toward the malicious router [6, 23].

We propose and evaluate an encryption-based NoC architecture in this article to address the mentioned issue. The architecture features (1) a novel source routing method that works with fully encrypted packets, including source/destination addresses, and (2) a situation-driven stochastic path diversification approach to bypass the failed components. The proposed architecture can effectively deal with many crypto-analysis attacks and malicious faults injected into NoC communication channels or routers' flit buffers. The source-computed paths, which are embedded into the header of a secure packet, do not reveal any information about the source/destination. Nevertheless, they are enough to guide NoC routers to route secure packets. Upon arrival at the destination node, the encrypted destination address embedded in every secure packet will match the router's pre-encrypted destination. The packet is ejected to the local core.

This article is an extension of our previous work [29]. The contributions of this article concerning our previous publication are as follows.

- We have extended the attack model. This article addresses cases where an adversary conducts a combination of crypto-analysis and fault-injection attacks to strengthen the attack scenario.

[1]The encryption process takes tens of cycles to process a block of data as low as 16 bits. Consequently, a typical packet with less than 100 bits of the header will be delayed 100s of cycles at each hop to encrypt the entire packet. Moreover, the encryption key must be exchanged with every single node on the route to make the routing mechanism possible. Due to these huge delays, we believe that entire-packet encryption is not a feasible solution.

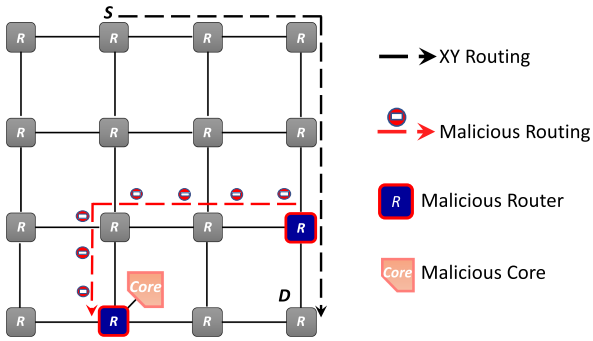


Fig. 1. The used threat model consists of HT-infected routers and malicious tasks.

- The path diversification algorithm, which is the heart of the method in dealing with attacks, is updated to offer paths with a dynamic probability distribution stochastically. This would have a direct impact on the achieved security chip-wide.
- The NI/router architectures are updated to support stochastic routing for fault-injection attack tolerance. Accordingly, the hardware evaluation section is updated to reflect the overheads.
- Fault-injection attacks have been conducted to evaluate the security/reliability gained by the proposed security system.
- A literature review section has been added to the article that shows the gap for this work.

The rest of this article is organized as follows. Section 2 presents the necessary background, including related work and the addressed security threats. Section 3 discusses the proposed security system, including the architecture and the routing method. Simulation experiments and the obtained results are presented and discussed in Section 4, and finally, Section 5 concludes the article.

2 BACKGROUNDS

In this section, we review the threats that may alter data communications security in NoC-enabled MPSoCs and related literature in security threats mitigation.

2.1 Threat Models of NoC Communications

Cores of multi-core SoCs are connected to the on-chip network using their local **network interfaces (NIs)** and routers. Cores and NIs deal with the local data only. However, routers are in charge of forwarding both local and global data. As a result, adversaries prefer routers to host their HTs to wreak havoc on security. Generally, security goals are summarized as confidentiality, integrity, and availability, and the adversary's goal is to compromise at least one of these goals through a plethora of attacks. The most widely addressed attacks applicable to NoCs consist of: denial of service attacks (packet flooding [21], packet dropping [15, 16], packet misrouting [14]), packet tampering [4, 9], packet duplication [4, 10, 11], and eavesdropping [10, 11].

The addressed threat model in this article is depicted in Figure 1. Per this threat, an HT-infected router sniffs the secure exchanged information between a source/destination pair. The malicious router may copy the packet contents and send them to another malicious node that runs rogue software for further processing/analysis to discover sensitive information. Even though the sender might encrypt the sensitive data before injecting it into the network, crypto-analysis attacks can still be launched to infer secret encryption keys. Timing attacks are one example of crypto-analysis in which an adversary makes intentional collisions between its own data and the sensitive data

on a specific path to obtain valuable information about the timing and the volume of the sensitive information [26, 27]. Other studies [10, 11] mention linear and differential crypto-analysis in their threat models to crack keys as well. It is worth mentioning that data corruption attacks are out of the scope of this work.

2.2 Related Work

Existing solutions to protect the confidentiality of NoC packets are mainly based on the application of cryptographic protocols such as Diffie-Hellman protocol [7, 31], **elliptic curve cryptography (ECC)** [7, 34], and MAC [38]. Researchers have also proposed the application of secure zones [2, 25, 33] to create (single/multiple) zones to prevent unauthenticated accesses from attackers outside the zone to the sensitive resources. Although these methods can preserve security against attacks like tampering, NoC packets are still vulnerable to more sophisticated attacks such as crypto-analysis attacks at which some hardware components of the MPSoC are managed to support the malicious activities. Such hardware components may include core(s), router(s), and/or network interface(s). The success rate of crypto-analysis attacks can also be boosted once combined with fault-injection attacks that try to guide the traffic to specific regions of the NoC.

Fernandes et al. [18] have introduced the application of secure zones and three communication scenarios for routing sensitive packets inside and outside the zones. The routing scenarios include cases of (1) the source and destination IP addresses are located within the same zone, (2) source/destination IPs within the same zone with their communication paths partly outside the zone, and (3) source and destination IPs are not located in the same zone. This work attempts to route the packet as much as possible within the zone while packets passing through insecure zones are encrypted. However, the assumed threat model in this work oversimplifies the problem as it ignores having any malicious routers or NI throughout the entire MPSoC. Since the routers are presumed to be trustworthy, this strategy is vulnerable to most attacks launched by malicious routers.

In Reference [31], an architecture to address the implementation of the Diffie-Hellman group key exchange is proposed. They tried to reduce the impact of the Diffie-Hellman protocol on the network performance by controlling the number of IPs involved in the key exchange process. The formed secure zone protects sensitive data by processing them inside the zone. However, the method is still vulnerable to fault-injection attacks. Sharma et al. [35] proposed a group key agreement protocol to establish a session key between group members. The method uses **Diffie Hellman elliptic curve (DHEC)** to address Man-in-the-Middle and key leakage attacks; however, this research does not consider attacks on the exchange protocol, e.g., tampering.

Sepúlveda et al. [32] have proposed a tunnel-based network interface that encrypts all portions of the packet except the destination address. They use AES in counter mode for encryption. SipHash is used for computing the packet hash digest, authentication, and tamper detection. The major shortcomings are (1) the application of the AES encryption, which leads to large area and performance overheads compared to the baseline NoC, and (2) leaving the header information unprotected.

Charles and Mishra [11] assumed a threat model where sniffed packets are sent to a malicious IP for further analysis. The proposed incremental encryption is tailored for specific data types, such as images where chunks of data are fetched from consecutive memory locations. New packets are encrypted with fewer computations, since the whole encryption process is no longer needed. The packet headers transmitted as plaintext are unprotected against stealing and crypto-analysis attacks. Furthermore, various attacks model and countermeasures are widely studied [30].

Charles et al. [10] proposed a key exchange mechanism and an anonymous routing scheme that hides source/destination pairs. They use the MAC-then-Encrypt protocol to secure the data. This

method uses multi-layers of encryption to exchange a shared key securely. After the shared key is successfully established, each node only knows its previous and next neighbor and is unaware of the final source and destination. In addition to significant power and area overheads, attackers can break the route discovery phase by tampering with the route confirmation packets.

Ravikumar et al. [24] have used a test module to perform periodic tests on the physical channels, processing elements, and buffers of the NoC. Alongside the faults that non-intentionally happen in the network, the DoS and power viruses are the threat models in their research. They assumed the result of the test are captured with the global test manager, and these results can be used in the packet retransmit and rerouting. Although the reliability of the NoC is guaranteed, the footprint of the hardware is considerable concerning the test structure and test targets. Furthermore, when tested, an attacker can paralyze the targets, so the DoS attack avoidance cannot be addressed.

Boraten and Kodi [4, 5] have introduced a new attack model using HTs to create a DoS attack in links of NoCs. They have tried to block the link bandwidth using multiple injections. They have proposed a heuristic-based fault detection model to discover HTs in the infected links by continuing to probe and observe the behavior of the links experiencing transient or permanent faults [5]. Also, several techniques, such as data scrambling, packet certification, and node obfuscation, are proposed to avoid Trojan activation. They have proposed a combination of **Algebraic Manipulation Detection (AMD)** and **cyclic redundancy check (CRC)** codes to embed the path information into the packet header. As a result, packet integrity is preserved, and unauthorized duplicated packets can be detected/dropped at the destination router.

Existing solutions in anonymizing NoC communications fail in hiding source and destination information from attackers. This article addresses this deficiency by proposing a combination of lightweight encryption and source routing. Besides, the reliability of our architecture against fault-injection attacks was a secondary target in this study. We study the effect of faults in a network in the presence and absence of our proposed architecture.

3 PROPOSED SECURITY SYSTEM

In the proposed security system, packets are divided into secure and non-secure. Secure packets have high-security requirements, i.e., their information must be encrypted, and neither the packet contents nor their source/destination addresses should be disclosed to an unauthorized party. Conversely, non-secure packets are sent in plaintext, since they do not contain sensitive information. Hence, two different routing mechanisms (one for each packet type) are employed, which will be described later in this section.

3.1 Secure Anonymous Routing

The idea behind secure source routing is to exclude source/destination addresses in the packet. Instead, the approximate route and turns a packet needs are embedded in the secure packets' header. The secure routing stochastically selects a routing scenario for every secure packet, e.g., XY, YX, or XYX. Then, the shortest path following the selected strategy will be computed and embedded in the header flit of the secure packet. The path information is generated not to reveal secure packets' source/destination addresses. Instead, the header of the secure packet only contains the information of two turns that the packet might use. The proposed routing offers the following three routing scenarios for a secure packet: (i) an XY path toward the destination with one real and one misleading turn embedded in the header of the packet, (ii) a YX path toward the destination with one real and one misleading turn, and (iii) an XYX path with two actual turns embedded in the packet header. The goal of having multiple routing scenarios further confuses adversaries about the packet source and destination.

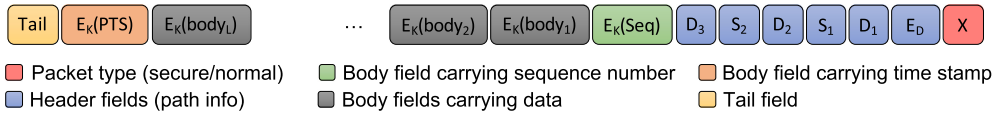


Fig. 2. Format of a secure packet in the proposed anonymous routing. The packet carries its displacement to a point at which the packet should make a turn.

From the adversary's point of view, every local router takes one of the following four actions to process a secure packet: (1) The packet is being forwarded in the direction from which it was received; (2) the packet is redirected from X (horizontal) direction to Y (vertical) direction; (3) the packet is redirected from Y direction to X direction; or (4) the packet is being ejected from the network upon reaching the destination.

The embedded information in the secure packet header (format shown in Figure 2) is as follows. The first field (X_{bit}) is a single bit that indicates the packet type: secure or normal (non-secure). The E_D field carries the encrypted version of the destination address for anonymous routing (details will be explained later in this section). The packet traversal directions can be found in D_1 to $D3$ fields, and they can be one of these four options: X^+, X^-, Y^+, and Y^-. S_1, and S_2 represent the number of strides (hops) the packet has to traverse to reach its first and second turns, respectively. The next field is the packet's encrypted sequence number, $E_K(Seq)$. The fields E_D, $E_K(Seq)$, and $E_K(body_i)$, and $E_K(PTS)$ are encrypted using a pre-shared key K using the symmetric-key encryption scheme. The adopted encryption is **Hummingbird-2 (HB-2)**, which will be discussed later in Section 3.3.1. This key needs to be exchanged in prior between an *initiator* and a *recipient* node. $E_K(PTS)$ is the path time stamp added to the packet by an initiator (at NI) to allow the method to monitor the network travel time for the packet. The initiator will use this information to update its stochastic behavior for possible fault attacks in a path (details are later in this section). Last, the packet is terminated with a tail flit transmitted in plaintext.

Algorithm 1 describes the secure, anonymous routing scheme. The basic idea is that a secure packet will be ejected from the network only if its encrypted destination field is equal to the encrypted address of the router[2] (reflected in lines 1–3). Otherwise, S_1 will be checked, and in case it is non-zero, it will be decremented by 1 (lines 4–6), and the packet will be moved as D_1 commands. After S_1 becomes 0, the same condition will be rechecked for S_2 and decremented by one as well (lines 7–9). At last, if neither of the above conditions is met, then the packet will be forwarded according to D_3 until it reaches its destination (lines 10–12). As for the non-secure packets, we use the well-known dimension order (XY) routing to allow them to share virtual channels with secure packets.

In a typical NoC, header flits are generated by the NI component. To enable both routing configurations, packet headers must support both methods. Hence, a secure header generator is needed in the NI, described in Algorithm 2. It first sets the packet type; then assigns the encrypted destination address to the E_D field. The horizontal and vertical differences between the current and destination nodes are computed in line 3. The algorithm stochastically chooses between $L1$, $L2$, and $L3$ routing scenarios to generate the path. $L1$ and $L2$ generate a path that contains a real and a misleading turn, whereas $L3$ generates a path with two real turns. As can be noticed, in the $L1$ subroutine (lines 6–12), S_1, D_1, and D_2 are computed deterministically based on the source/destination coordinates; however, S_2 and D_3 are chosen randomly to mislead the attacker about the packet's destination. $L2$ (lines 13–20) follows the same idea, except it prioritizes vertical movement over horizontal one. Conversely, in $L3$ (lines 21–27), S_1, S_2, D_1, D_2, D_3 are computed with some levels of

[2]Since the destination address, E_D, is encrypted, every router must also have its encrypted address, E_R, ready for comparison. This way, there is no need to decrypt the field E_D to ensure security.

ALGORITHM 1: Secure Routing

Input: Strides S_1, S_2, Directions D_1, D_2, D_3;
Input: Encrypted Destination Address E_D;
Input: Encrypted Router Address E_R;
Output: Selected Output Channel $Out_{Channel}$;

1: **if** $(E_D == E_R)$ **then**
2: $Out_{Channel} = Local$;
3: **else**
4: Set $Next_{VC} = Current_{V_C}$; # 1 for XY and 2 for YX packets
5: **if** $(S_1 \neq 0)$ **then**
6: $S_1 \leftarrow S_1 - 1$; and $Out_{Channel} = D_1$;
7: **else if** $(S_2 \neq 0)$ **then**
8: $S_2 \leftarrow S_2 - 1$; and $Out_{Channel} = D_2$;
9: **else**
10: $Out_{Channel} = D_3$;
11: Set $Next_{VC} = 2$; # Switch over VC_2 to take the final X
12: **end if**
13: **end if**
14: Return;

adaptivity. This is done by randomly selecting the variable m, which denotes the number of hops to be taken in the first and last X movements. So, if a malicious router is in the path, then there is always a high chance of bypassing it.

Secure, anonymous routing uses a simple stochastic process to select one of the routing scenarios $L1$ to $L3$. This addresses fault-injection attacks that create congestion at some parts of the network to lead the traffic toward specific routers. The detoured traffic can be subject to crypto-analysis attacks conducted by compromised routers. The algorithm picks each routing scenario Li with the probability P_{L_i} such that $P_{L_1} + P_{L_2} + P_{L_3} = 1$. Although the probability values are initially set to be equal, the NI updates them with respect to the turnaround time of the paths. The turnaround time of paths is calculated using the $E_K(PTS)$ fields of secure packets and their ACKs (architectural details of this policy are discussed in Section 3.3.3). For example, suppose the forwarded secure packets under the $L1$ scenario show increasing turnaround time. In that case, the NI reduces P_{L_1} by 0.1 and adds 0.05 to P_{L_2} and P_{L_3} with a lower bound of 0.1 for each of the probability values. This way, (1) the suspicious path altered by a fault-injection attack will have a lower chance of being selected again, and (2) we never give the attacker the option to disable one of our routing scenarios.

To shed light on Algorithms 1 and 2, we provide illustrative examples in Figure 3. In the one-turn routing scheme, S_2 and D_3 fields are randomly assigned to hide the real destination. The packet will be ejected to the local port at the destination node. A misleading path could either make a turn at the destination or either go further in the Y direction and turn. Although the packet will eventually reach its actual destination node, from the malicious router's (may be any of nodes 1, 2, 2′, 3) point of view, numerous nodes can be the ultimate destination of the packet (the area highlighted in blue). Two-turn routing was also leveraged to enhance further the path diversity, in which all the header fields are pre-assigned deterministically. In this scenario, the secure packet can detour the malicious node (given that such information is available) by making the first turn at node 1 even before reaching the destination column. It makes a second turn at node $3^{'}$ and then moves in the

Algorithm 2 : Secure Path Computation

Input: Source & Destination Address Plaintext P_s, P_d;
Output: Encrypted Destination Address E_D;
Output: Strides S_1, S_2, Directions D_1, D_2, D_3;
Output: Packet type X_{bit};

1: $X_{bit} \leftarrow 1$;
2: $E_D \leftarrow HB\text{-}2(P_d)$;
3: $\Delta X = P_d.X - P_s.X$; and $\Delta Y = P_d.Y - P_s.Y$;
4: Set $Current_{V_C} = 1$; # Initial VC for all paths except YX
5: Goto L1, L2, or L3 with a probability distribution of P_{L_1}, P_{L_2} and P_{L_3} s.t. $P_{L_1} + P_{L_2} + P_{L_3} = 1$;
L1: (an XY path) # Path with real and misleading turns
6: $S_1 \leftarrow abs(\Delta X)$;
7: $S_2 \leftarrow Rand(n)$; and $D_3 \leftarrow Rand(X^+, X^-)$; # To mislead the attacker
8: **if** $(\Delta X \geq 0)$ **then**
$D_1 \leftarrow X^-$; **Else** $D_1 \leftarrow X^+$;
9: **end if**
10: **if** $(\Delta Y \geq 0)$ **then**
$D_2 \leftarrow Y^-$; **Else** $D_2 \leftarrow Y^+$;
11: **end if**
12: Return;
L2: (a YX path) # Path with real and misleading turns
13: $S_1 \leftarrow abs(\Delta Y)$;
14: $S_2 \leftarrow Rand(n)$; and $D_3 \leftarrow Rand(Y^+, Y^-)$; # To mislead the attacker
15: Set $Current_{V_C} = 2$; # Initial VC for YX path
16: **if** $(\Delta Y \geq 0)$ **then**
$D_1 \leftarrow Y^-$; **Else** $D_1 \leftarrow Y^+$;
17: **end if**
18: **if** $(\Delta X \geq 0)$ **then**
$D_2 \leftarrow X^-$; **Else** $D_2 \leftarrow X^+$;
19: **end if**
20: Return;
L3: (an XYX path) # Path contains two real turns
21: $m \leftarrow RandBetween(0, \Delta X - 1)$; and $S_1 \leftarrow abs(\Delta X) - m$; # Take m hops once returned to X
22: $S_2 \leftarrow abs(\Delta Y)$;
23: **if** $(\Delta X \geq 0)$ **then**
$D_1, D_3 \leftarrow X^-$; **Else** $D_1, D_3 \leftarrow X^+$;
24: **end if**
25: **if** $(\Delta Y \geq 0)$ **then**
$D_2 \leftarrow Y^-$; **Else** $D_2 \leftarrow Y^+$;
26: **end if**
27: Return;

X direction. The packet will eventually exit the router when the comparison of E_D and E_R is true; however, the dummy path (the red region) will mislead the intermediate node 3′ if it is a malicious one. Also, diversifying the packet path diminishes the chances of being intercepted by malicious nodes. Since the packet's ultimate destination is not revealed to the adversary, crypto-analysis and timing attacks are impossible in this environment.

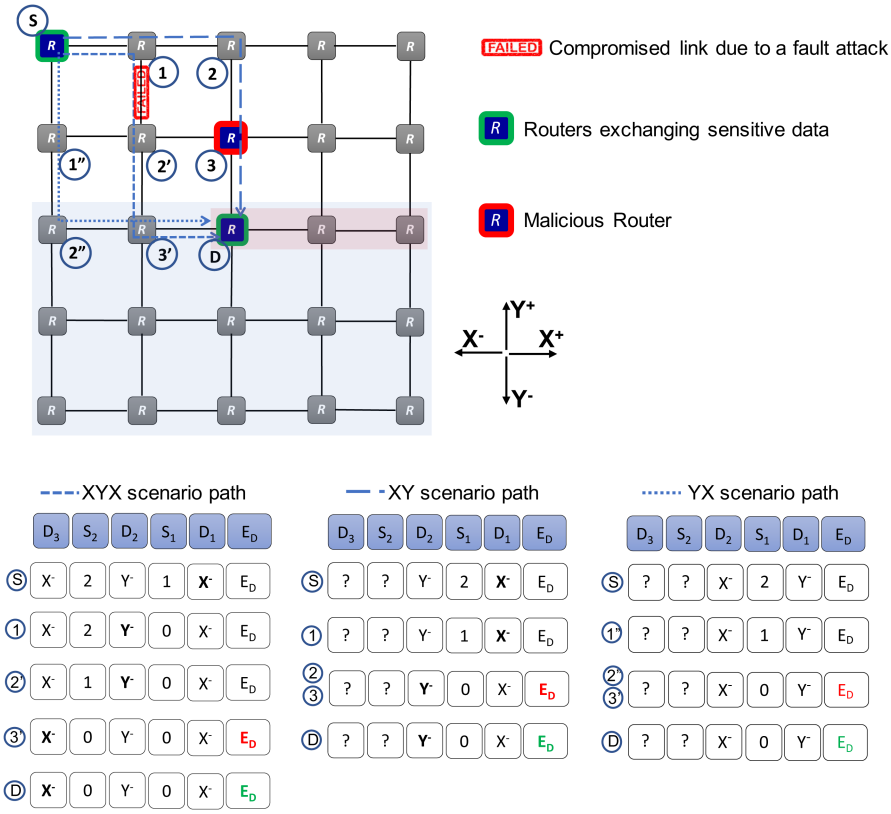


Fig. 3. Example cases of routing a secure packet in the network with one or two legitimate turns embedded in the packet. The secure packet carries information that includes when and where to turn. From the attacker's point of view, all routers in red or blue shaded areas are potential destinations of a two-turn or one-turn packet, respectively.

3.2 Deadlock Freedom

3.2.1 Deadlock Problem. Applying XYX routing for secure packets in its raw form might create a cyclic dependency between NoC buffers. Figure 4(a) shows an example case of four secure packets waiting to access their required buffers. As shown, there are four source nodes, i.e., S_1, S_2, S_3, and S_4, sending secure packets simultaneously to their destination nodes (destination nodes are not specified in the figure). Packets P_1 and P_3 are about to make their second turn and continue their paths in X direction, while packets P_2 and P_4 are making their first turn from X to Y direction. However, each of these four packets has to wait for a buffer already occupied by another packet of the same group. In this example, the clockwise cycle shown in Figure 4(b) depicts the cyclic dependency between packets. Subsequently, the waiting situation will never end. Since the involved packets cannot proceed, they will not release their current buffers. The color code used for illustrating clockwise dependency in Figure 4(b) helps to relate packets to turns. The same situation may happen for counter-clockwise turns, shown in black in Figure 4(b); however, for the sake of simplicity, packets involved in a counter-clockwise cyclic dependency are not shown in Figure 4(a).

3.2.2 Deadlock Prevention. To address the mentioned deadlock situation and ensure deadlock freedom of the proposed system, we have devised two virtual channels, i.e., VC_1 and VC_2, with the

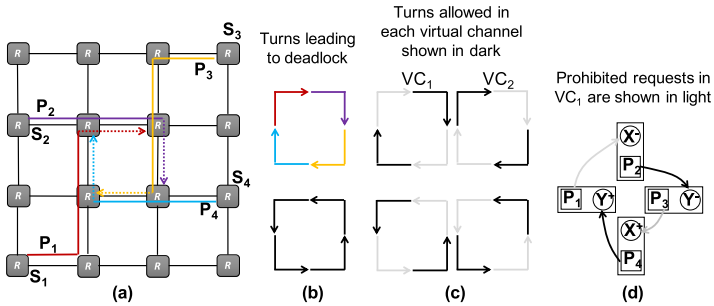


Fig. 4. A group of secure packets waiting to access buffers to continue their path in panel (a) made a cyclic dependency shown in color in panel (b). The cycles can be avoided by using a different virtual channel for the last X movements of secure packets so that two turns of each cycle are assigned to each virtual channel (c).

following allocation policies. Both VCs are available for normal packets at the source; however, normal packets do not have the option to change VC during their journey. In other words, once a normal packet is assigned to a VC, it will use the same VC until it reaches its destination. Normal packets are assigned to VCs based on the traffic condition at the source node. However, secure packets have some restrictions in acquiring VCs, which the routing algorithm enforces based on the routing scenario adopted for each secure packet. Policies used for allocating VCs to secure packets are as follows.

- A normal packet can use either VC_1 or VC_2 without permission to switch its current VC.
- A secure packet with an XY path is only allowed to use VC_1. The source node implements this policy by injecting the packet into the network using VC_1. The assigned VC will be used for the entire path of the secure packet.
- A secure packet with a YX path is only allowed to use VC_2. The packet enters VC_2 and keeps using it until reaching its destination.
- An XYX secure packet starts with VC_1 and continues its path till reaching the second turn, i.e., a turn from Y to X. At this point, the secure packet has to switch over VC_2. In other words, the packet switches over VC_2 when the packet returns to the direction X where the actual destination is located.

Turns allowed in each VC are shown in Figure 4(c). These are the turns allowed in the virtual networks created, respectively, from all buffers of VC_1 and VC_2 over the entire network. The former (VC_1) implements the XY routing without cyclic dependency. Similarly, there is no cyclic dependency in the virtual network created by VC_2 as it implements the YX deterministic routing. Thus, there is no deadlock within each virtual network. From the deadlock point of view, secure packets of type XY and YX that use VC_1 and VC_2, respectively, follow the same routing policies in the corresponding VC, so there is no deadlock between one-turn secure packets and normal packets. The exception is granted to XYX secure packets, which can switch to VC_2 for their second turn. Technically, this cascaded routing allows the packet to make the YX turn (when switching over VC_2) without causing deadlock [12]. The key point is that XYX secure packets cannot ask for VC_1 after switching to VC_2.

The dependency analysis (Figure 4(d)) shows the deadlock freedom for VC_1. A similar analysis applies to VC_2 as well. The two points mentioned prove the deadlock-freedom of the proposed routing. Notably, secure packets do not carry any information that reveals if they have taken an $L1$, $L2$, or $L3$ path. The intermediate routers implement the VC policies by looking at the current VC of a secure packet and if a turn from Y to X has occurred.

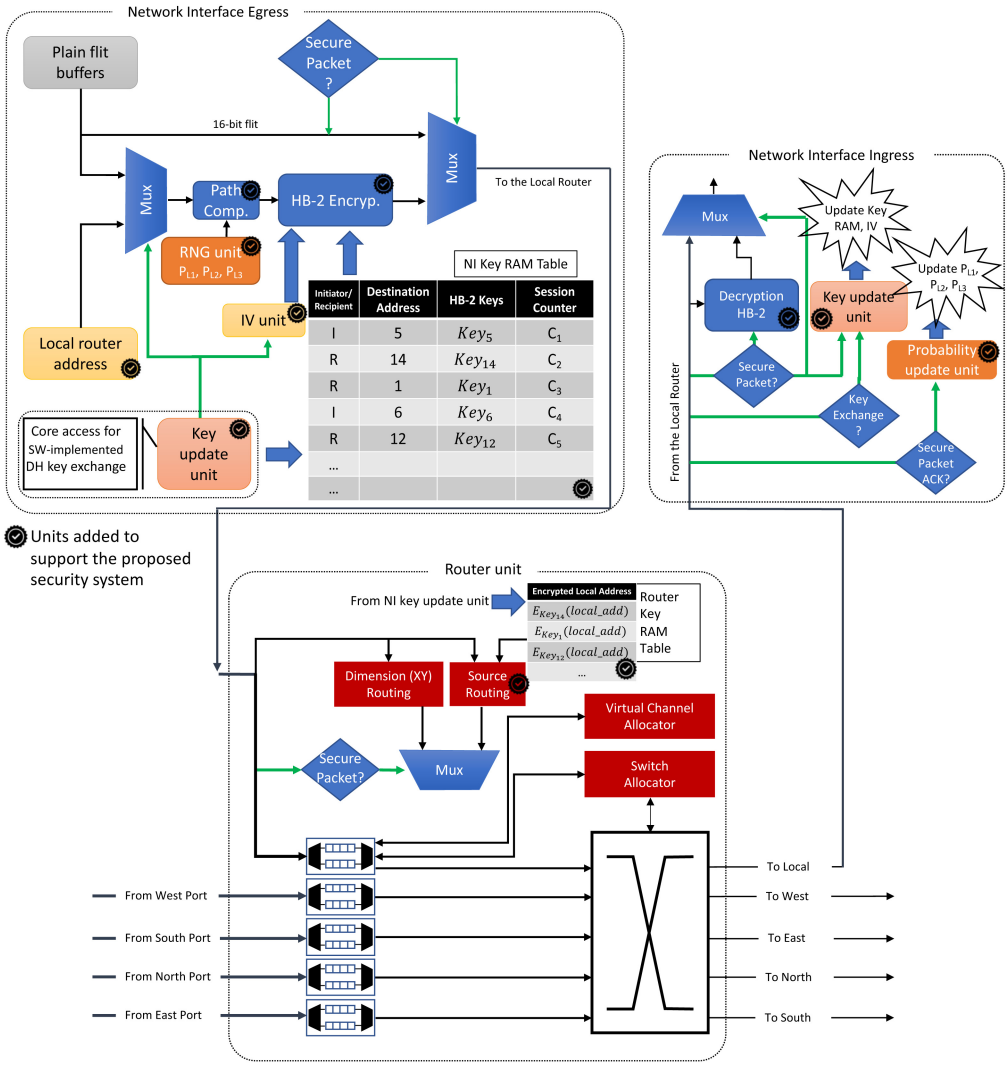


Fig. 5. Proposed architectures for the network interface and the router to support anonymous routing.

3.3 Proposed NoC Architecture

To enable secure routing, the NoC architecture should be modified in both the NI and router to facilitate source routing, encryption, and key exchange. Figure 5 depicts the proposed architecture for the NI and router. The main idea here is to apply the most modifications to the NI's design to avoid increasing the router's delay. In the following, we explain how the mentioned tasks are done.

3.3.1 Encryption. Considering the limited hardware resources of NoC routers, we need to choose a secure, lightweight block cipher with low power consumption, area, and performance penalties. HB-2 has recently emerged as a viable solution with a 128-bit key and 128-bit internal state [17]. According to Reference [17], HB2-ee20c (the version we use in our implementation) only takes 20 clock cycles to encrypt a 16-bit block. Faster versions of HB-2 will impose much more area and power overheads; however, without loss of generality, the architecture designer can adapt any

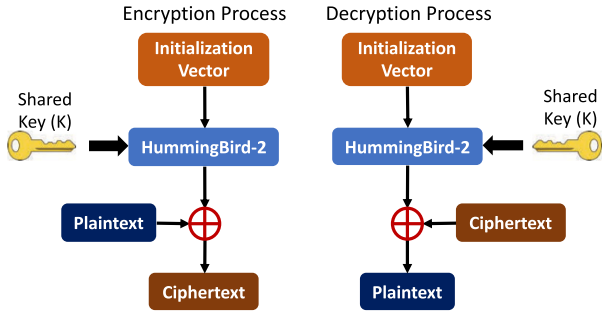


Fig. 6. Encryption and decryption in counter mode.

other version of HB-2 or symmetric-key block ciphers that best suits their needs. We use HB-2 in the counter mode to encrypt both the destination address and body flits. The block size is 16 bits, and the key size is 128 bits. An overview of the encryption scheme is illustrated in Figure 6. An **Initialization Vector (*IV*)** is encrypted with a shared key K and then XORed with a plaintext block to obtain the ciphertext. In each encryption session, IV is incremented by one, and due to the diffusion and confusion properties, the block cipher output will have a completely different outcome. The decryption process is also depicted in Figure 6, where the encrypted IV is XORed with the ciphertext to obtain the plaintext. Figure 5 shows a single encryption module designed in such a way that it can encrypt the destination address and the body flits. Packets will be passed to the encryption module if a secure flit type bit X_{bit} is detected. Moreover, the router's own address is encrypted with the same key K and stored in the router key RAM table. When a secure packet enters one of the input E, W, N, and S buffers, its encrypted destination will be compared with the contents of this table and processed accordingly. After body flits are decrypted, they are ejected to the NI through the local egress port. Last, the encryption modules are power-gated to prevent non-secure packets from unnecessary encryption and dissipating dynamic power dissipation.

3.3.2 Key Exchange. A node involved in a secure connection can be either an *initiator* or *recipient*. An *initiator* will send a key exchange request to a *recipient*. After both parties compute the shared key, a new key will be written in their key RAM tables in the NI. As depicted in Figure 5 (NI key RAM table), *recipients* are shown with R, and *initators* are shown with an I. Since *recipient* is the ultimate destination, the router address will only be encrypted in this node and updated in the router's key RAM table. To securely exchange keys, we use the Diffie-Hellman key exchange mechanism [28]. The session counter defines how long a key should be kept in the key RAM table (shown in the session counter column of the key RAM table in the NI). We set the session expiration to be twice the worst-case delay of the network. This ensures the encrypted packets have enough time to arrive at their destination. When the session expires, new keys should be exchanged; accordingly, the key RAM table will be updated. Via the IV unit in the NI, the *recipient* node will keep track of this value for the incoming packets for proper encryption or decryption.

3.3.3 Routing Algorithms. As previously explained, secure packets are routed stochastically based on XY, YX, or XYX routing with the corresponding probability of P_{L_1}, P_{L_2}, and P_{L_3} such that $P_{L_1} + P_{L_2} + P_{L_3} = 1$. We have added a unit to the NI (shown in Figure 5) that monitors the turnaround time of the ACK packets to update the probabilities. If a unit is compromised due to a fault attack, then the turnaround time of secure packets taking that path will either increase (in partial failure case) or go to infinity (if the path is totally down). The probability update unit (architecture shown in Figure 7) (1) generates a time-stamp for secure packets, (2) calculates the turnaround

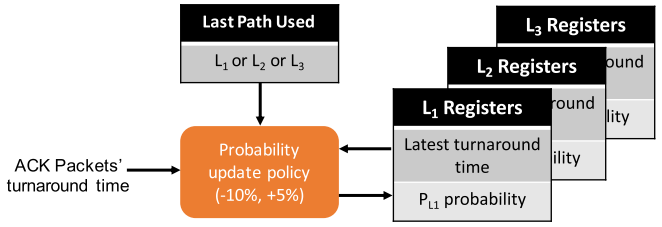


Fig. 7. The architectural design of the probability update module. The chance of a routing scenario with a relatively high turnaround time is decreased not to let future packets take that path very often.

time for each routing scenario with respect to the secure packets' time-stamp and the arrival time of their ACK, and (3) updates the path probabilities if a path shows an abnormal increase in its turnaround time. The unit reduces the chance of selecting an abnormal path by 10% and increases the chance of selecting the other two paths. For example, if $L1$ path experiences an abnormal turnaround time increase, then P_{L_1} will be reduced by 10% and P_{L_2} and P_{L_3} will see a 5% increase. The unit will never kill a path to allow the attacker to disable one of our routing scenarios.

After picking a routing scenario for a secure packet, the path is computed in the path-computation unit at the NI following Algorithm 2. This unit obtains the required encryption key from the key-RAM table and generates fields (as shown in Figure 2) of the secure packet accordingly. Then, the secure packet is ready to start its anonymous journey over the network. We added the necessary hardware to the NI and router logic to generate and forward secure and non-secure packets. When a router receives a secure packet, it prioritizes them over non-secure packets to minimize the period a secure packet lives in the network. The prioritization impacts the virtual channel and switch allocation stages on the router's pipeline by first assigning resources to secure packets.

A key RAM is designed within the NI to store the communication keys. We assume the NI is built in-house, similar to Reference [11], and the memory implementation is secure, i.e., it cannot be compromised by hardware Trojans. A node can either initiate or receive encrypted communication sessions. If a node is a recipient (denoted as R in the key RAM table), then its key RAM table will be updated, and accordingly, the router's local address must be passed to the input of the lightweight block cipher Hummingbird-2 [17] for encryption. Next, the generated ciphertext will be stored in a key table in the router (referred to as RK table hereafter) that holds the encrypted values of the local address using different session keys. Per receiving a secure packet, the encrypted address will be compared with the node's pre-encrypted address in the RK table, and the packet will be injected into the local port. If the counter threshold is passed, then the key RAM and the RK table also contain session counters that can terminate table entries (keys and encrypted addresses, respectively). The threshold value will be decided based on the worst-case delay of the NoC. It is worth mentioning that when a node is an initiator (denoted as I in the key RAM table), the RK table will not be updated. Packets are decrypted after entering the NI through the local egress port. Last, the encryption modules are power-gated to avoid the encryption of non-secure packets and, subsequently, unnecessary dynamic power consumption.

4 EXPERIMENTAL EVALUATIONS

To evaluate the proposed security system, we have conducted two types of experiments. First (Section 4.1), we have used Noxim [8] NoC simulator to simulate the security and performance behavior of an NoC-based MPSoC equipped with our proposed method. Second, we have implemented and synthesized the proposed architecture using Verilog HDL to estimate its hardware and latency overheads with respect to a baseline system. The former results are available in Section 4.2.

4.1 Security Evaluations

Regarding security, the proposed system guarantees adversaries cannot replay packets to the same destination by combining symmetric-key encryption in counter mode and packet sequence number. The sequence number allows the communicating parties to track their exchanged packets. Since the sequence number is encrypted in the counter mode, the man-in-the-middle has no information about the packet content or the number of the exchanged packets. As a result, the packet cannot be replayed with the same sequence number, and any attempt to do so will expose the adversary. The proposed system is also protected against crypto-analysis attacks, as the attacking router/core cannot decrypt any packet. Furthermore, the attacking router/core may not receive all secure packets exchanged between the source and destination pairs due to the stochastic behavior used in the source routing. We have done network-level simulations to evaluate the system's performance against fault injection attacks.

We modified the Noxim simulator to support two types of packets (secure and non-secure packets). Secure packets are routed based on a pre-computed path (see Section 3 for explanation) embedded in the packet header. The non-secure packets are routed using the XY and YX deterministic routing algorithms to allow the routers to share virtual channels between normal and secure packets. Simulations are conducted for 50,000 cycles with 10% warm-up time on 8×8 and 6×6 networks under various rates of traffic generation and secure packets ratios. Average network latency has been measured and reported as a widely used factor in reporting NoC performance [13, 39]. Latency evaluations are done under uniform and hotspot traffic generation patterns. In both traffic models, each core generates packets according to a Poisson process with a rate of λ packets per cycle. As the Poisson process of each node is independent of others, different cores may generate different packets simultaneously. However, the destination selection differs between the two traffic models. In uniform traffic, the destination of each normal packet is uniformly selected out of all network nodes except for the generator node. However, in the hotspot traffic, x% of traffic targets a specific core of the network while the rest of traffic, i.e., $(1 - x)$%, finds its destination uniformly [36]. In our simulations, packets consist of eight flits. Based on our hardware implementation, the HB-2 encryption module requires 20 cycles to encrypt a 16-bit data block. The Network delay of secure packets is computed with respect to their length and the mentioned encryption delay. The key exchange delay is not accounted for in our simulations, since key exchange happens once per session, and only a fraction of the packets need encryption.

To study the resiliency of the proposed system against fault-injection attacks, we have injected faults into the network channels and buffers based on the following threat models.

- **Channel faults:** NoC communication channels are disabled temporarily for a random duration of time to mimic a victim channel that has temporarily stopped working due to a fault attack. Once a channel is attacked, all buffers and VCs related to the channel freeze, and packets cannot move forward.
- **Buffer faults:** NoC virtual channel buffers are disabled temporarily for a random duration of time to mimic a buffer attacked by an injected fault. In this case, the VC buffer loses its data, and packets will be dropped.

We have used three probability distribution functions, i.e., normal, normal, and uniform distribution, to generate the time of the fault-injection attack, the duration of the attack, and the target (either channel or buffer) of the fault attack, respectively. The attack time and duration follow the same normal distribution (Equation (1)) with μ as the mean of the attack time/duration, set to half the simulation time. σ defines the standard deviation of the attack time/duration. After generating a random number according to this probability function, this number will be passed to the fault injector component developed in the Noxim simulator to disable the target component accordingly.

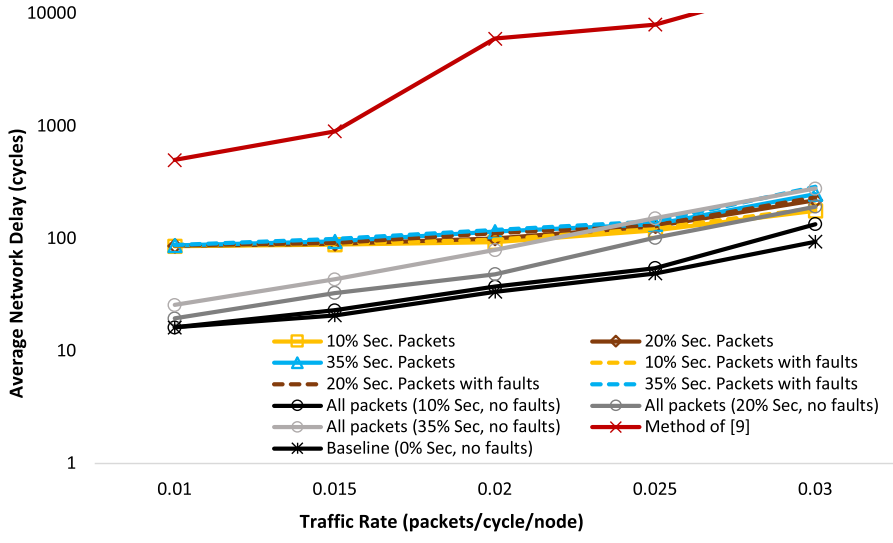
In this equation, x is the input variable determined by the current simulation cycle:

$$p(x|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right). \quad (1)$$

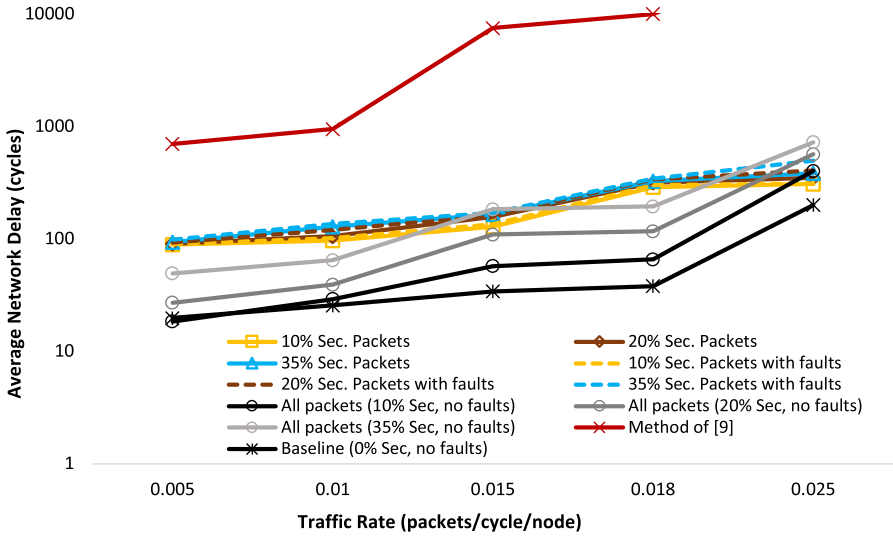
The third distribution function is used to select the target NoC component. For this purpose, we label channels and buffers of the NoC using decimal integer numbers $[1..n]$, and then a uniform random number is generated between 1 to n. This number points to the specific component that should be disabled for the attack. For preventing blind fault-injection attacks, only channels/buffers involved in secure transmission at the time of the attack will participate in this race. In other words, channels/buffers off the secure paths are not considered the targets of fault attacks.

The results of the fault-injection attack experiments are as follows. First, the algorithm could bypass all fault-injection attacks using alternative routing scenarios. A re-transmission option was eventually used to deliver the secure packets that encountered a failed buffer/channel. This is done with the help of the encrypted sequence number embedded in the secure packets. The proposed method could successfully tell when a secure packet was missing. This comes at the cost of performance loss due to the re-transmission of such secure packets. To study such a penalty, we have compared the average network delay of normal and secure packets for 6×6 and 8×8 networks against the baseline NoC. Here, we define 10%, 20%, and 35% of packets as secure packets so that the network routes them using the proposed source routing method. Results of this experiment are shown in Figures 8(a) and 8(b) for 6×6 and 8×8 networks, respectively. As we see in the results, unless the packet generation rate is very high (above 0.02 packets/cycle) or the ratio of the secure packet is high (35%), the performance overhead of the proposed security system is affordable considering its remarkable security improvements. After a closer look at the results, the following observations are made.

- **Performance stability for secure packets.** The results reveal an important property of the proposed security system at the network level. Secure packets have shown an almost constant performance behavior for a given network size. As can be seen, the network delay of secure packets marginally changes by changing the rate of secure packets or conducting fault injections. This is owed to the fact that the secure packets are given the highest priority. So, when the rate of secure packets grows, the extra network delay will be mostly imposed on normal packets to keep the delay increase of secure packets as low as possible. It is a similar case for fault injections, where they lead to re-transmission of the secure packets. We also observe that the network delay of secure packets is less sensitive to the overall packet generation rate than all packets (that consists of secure and normal packets). These performance features show that the proposed security system is reliable and scalable and can be used in various network configurations.
- **Superiority to the literature.** The only method that encrypts the entire packet is presented in Reference [9]. Our comparisons with this method in terms of performance show its unacceptably remarkable performance penalty. The method proposed in Reference [9] uses a broadcast technique to hide secure packets' source and destination addresses (the red curve in the figure). However, broadcasting makes this method impractical. We could not simulate this method for the cases of 20% and 35% secure packet ratio due to the large number of packets being flooded (at these rates, this method puts the network in saturation where no packet can be forwarded). For the 10% secure packet ratio, though, the average network delay is in the order of thousand cycles, which is at least 15× greater than ours, making this method an impractical solution.
- **No major performance variations for secure packets under fault injections.** The performance penalty that secures packets cause to avoid failed components is negligible.



(a)



(b)

Fig. 8. Average network delay versus different packet generation rates under uniform traffic pattern. Secure packets are generated as the ratios of 0% (considered as the baseline), 10%, 20%, and 35% for a 6×6 (a) and an 8×8 (b) networks. The proposed security system is compared with the method proposed in Reference [9] only for a secure packets ratio of 10%.

This fact is owed to the high flexibility path selection of the proposed stochastic routing algorithm.

- **Scalability.** Comparing Figures 8(a) and 8(b) confirms that the proposed security system demonstrates almost similar behavior while the network has expanded by 70%. We can observe nearly all mentioned performance behaviors for both network sizes.

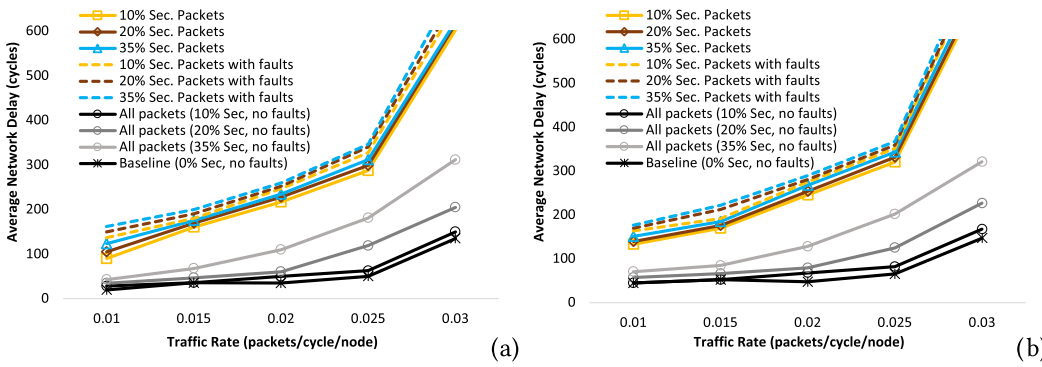


Fig. 9. Average network delay versus different packet generation rates with (a) 5% and (b) 10% hotspot traffic for a 6×6 network.

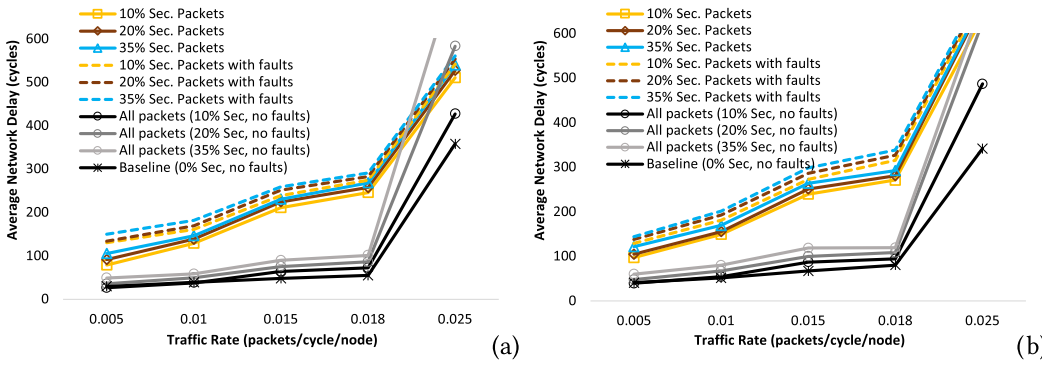


Fig. 10. Average network delay versus different packet generation rates with (a) 5% and (b) 10% hotspot traffic for an 8×8 network.

In the next experiment, we studied the network behavior of the proposed method under the hotspot traffic pattern. We used hotspot rates of 5% and 10% for this study. Results are shown in Figures 9 and 10 for 6×6 and 8×8 networks, respectively. Most performance features under the uniform traffic can be seen here as well. However, the desired features have faded relatively due to having hotspot nodes. Such nodes can impose tens of delay cycles to those secure packets with no other option to bypass the hotspot nodes.

In a different experiment, we counted the number of secure packets that use each routing scenario $L1$ to $L3$ under no, low, mid, and high rates of fault-injection attacks. Results displayed in Figure 11 show that when a higher number of fault-injection attacks are conducted, the stochastic property of the routing algorithm tends to route a higher percentage of secure packets using $L3$ routing scenario. This makes sense as this routing scenario has a higher level of adaptivity and naturally has more capability to bypass failed components. As a result, more secure packets are routed using an $L3$ route.

4.2 Hardware Evaluations

We have implemented the proposed security system in Verilog HDL to verify its functionality and estimate the hardware overhead. We have then synthesized an MPSoC with 16 32-bit MIPS processors interconnected as a 4×4 mesh NoC [19]. The MPSoC was designed in Verilog and synthesized using NanGate 45 nm technology in Synopsys Design Compiler tool for ASIC platforms. The

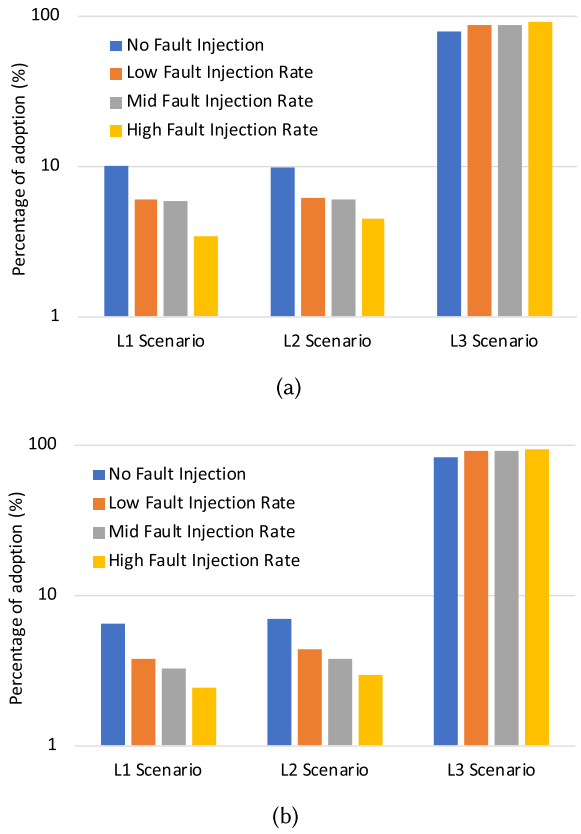


Fig. 11. Distribution of secure packets over $L1$ to $L3$ routing scenarios when the network experiences various rates of fault-injection attacks. Results are shown for a 6×6 (a) and an 8×8 (b) networks.

baseline NoC has four virtual channels per physical channel with a depth of eight buffers and a data width of 32 bits. The NI and the router of the MPSoC have been modified to accommodate the following components for the proposed security system.

- Key RAM table in the router: this module stores the encrypted versions of the router's local address. Each router will use this table to decide whether a new incoming packet should be forwarded or directed to its local core. A new communication channel between the NI and the router is designed to allow the NI to update the router's key RAM table.
- Key RAM table in the NI: This expands the router's table that keeps the exchanged keys, session counters, and other information required to enable secure communications.
- Source routing module: this module, which is added to the router, implements the logic of the proposed source routing method for secure packets. The router should be able to differentiate secure/normal packets and apply appropriate routing algorithms.
- The Hummingbird-2 encryption engine: the HB-2 module is implemented at the NI for data encryption.
- Probability update module: this module is implemented at the NI and keeps track of the turnaround time for secure packets. It updates the selection probability of routing scenarios $L1$ to $L3$ being selected.
- Control unit: this unit is implemented in the NI to keep track of the status of each secure communication in the setup phase. The control unit issues appropriate signals to update the

Table 1. Area and Dynamic Power Consumption Overheads of the Proposed Security System Compared to the Baseline Router and the Baseline 16-core MPSoC

		Silicon Area (μm^2)	Dynamic Power (μW)
Baseline	Baseline (Router + NI)	65161.5	209.9
	Baseline (16-Core MPSoC)	1762560.1	2685.1
Source Routing	Proposed (Router + NI)	81659.3	493.9
	Overhead w.r.t Baseline (Router + NI)	**25.3%**	**135.2%**
	Proposed (16-Core MPSoC)	1780016	2968.6
	Overhead w.r.t Baseline (16-Core MPSoC)	**0.9%**	**10.6%**
Stochastic Source Routing	Proposed (Router + NI)	82337.07	508.01
	Overhead w.r.t Baseline (Router + NI)	**26.3%**	**142.1%**
	Proposed (16-Core MPSoC)	1790732.7	3194.36
	Overhead w.r.t Baseline (16-Core MPSoC)	**1.5%**	**18.9%**

entries of the key RAM tables. Additionally, it controls the data path of both secure and non-secure packets. For the sake of simplicity, the module and its connections are not shown in our architectural figures.

The synthesis results of the 16-core prototypical MPSoC are reported in Table 1. We have implemented two versions of the proposed security system to understand its hardware cost better. The first version contains the entire method with all the previously explained components of Figure 5, referred to as *Stochastic Source Routing*. The second version, which is lighter, only addresses crypto-analysis attacks (referred to as *Source Routing*). The related units for probability updates are not implemented in this version. As Table 1 shows, the chip-wide silicon area of the proposed security system is negligible (less than 2% and 1%, respectively, for the full and light versions). The chip-wide power overhead for the source routing (the lighter version) is reported at 10.6%, assuming that the ratio of the secure packets is 10%. Increasing the ratio of the secure packets will produce more signal activities at the secure modules and consume more dynamic power. This number is 18.9% for the stochastic source routing (full version), which is relatively high due to the activity of the probability update unit. However, further improvements in power consumption can be obtained by applying routine dynamic power reduction methods like clock gating. The area and power consumption overheads of the proposed network interface along with the router are 25.3% and 135.2%, respectively, for the source routing and 26.3% and 142.1%, respectively, for the stochastic source routing. These results show that although the probability update unit does not impose a big area overhead, its power overhead is notable due to its high signal activity. Based on our synthesis results for each separate module, the HB-2 module is the main contributor to the power overhead of the design. The HB-2 module accounts for 80% (of 135.2%) of dynamic power consumption overhead.

5 CONCLUSIONS

This article proposed a novel security solution for NoC-based MPSoCs to safeguard sensitive data from crypto-analysis, man-in-the-middle, replay, and fault-injection attacks. The proposed architecture consists of a novel source-routing method along with encryption modules. The anonymity

employed by the source routing helped 100% hide the routing information of secure packets so that no intermediate router could discover where the packet would eventually eject from the network. This enhances data privacy and prevents various attacks, since the attacker will no longer have ample information to compromise security. The fact that the proposed source routing selects stochastically among multiple routing scenarios helped address fault-injection attacks, i.e., an adversary fails in directing the secure traffic toward a specific compromised router/core. Notably, the proposed system incurs minimal performance overhead, making it a practical solution for MPSoCs.

REFERENCES

[1] Dean Michael Ancajas, Koushik Chakraborty, and Sanghamitra Roy. 2014. Fort-NoCs: Mitigating the threat of a compromised NoC. In *Proceedings of the 51st Annual Design Automation Conference on Design Automation Conference (DAC'14)*. ACM Press, New York, NY, 1–6. https://doi.org/10.1145/2593069.2593144

[2] Siavoosh Payandeh Azad, Gert Jervan, and Johanna Sepulveda. 2019. Dynamic and distributed security management for NoC-based MPSoCs. In *Proceedings of the International Conference on Computational Science (ICCS'19)*, João M. F. Rodrigues, Pedro J. S. Cardoso, Jânio Monteiro, Roberto Lam, Valeria V. Krzhizhanovskaya, Michael H. Lees, Jack J. Dongarra, and Peter M. A. Sloot (Eds.). Springer International Publishing, Cham, 649–662.

[3] Swarup Bhunia, Michael S. Hsiao, Mainak Banga, and Seetharam Narasimhan. 2014. Hardware Trojan attacks: Threat analysis and countermeasures. *Proc. IEEE* 102, 8 (2014), 1229–1247.

[4] Travis Boraten and Avinash Karanth Kodi. 2016. Packet security with path sensitization for NoCs. In *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE'16)*. Research Publishing Services, Singapore, 1136–1139. https://doi.org/10.3850/9783981537079_0180

[5] Travis Boraten and Avinash Karanth Kodi. 2016. Mitigation of denial of service attack with hardware Trojans in NoC architectures. In *Proceedings of the IEEE International Parallel and Distributed Processing Symposium (IPDPS'16)*. IEEE, 1091–1100. https://doi.org/10.1109/IPDPS.2016.59

[6] Travis H. Boraten and Avinash K. Kodi. 2018. Securing NoCs against timing attacks with non-interference-based adaptive routing. In *Proceedings of the 12th IEEE/ACM International Symposium on Networks-on-Chip (NOCS'18)*. 1–8. https://doi.org/10.1109/NOCS.2018.8512166

[7] Luciano Lores Caimi and Fernando Gehm Moraes. 2019. Security in many-core SoCs leveraged by opaque secure zones. In *Proceedings of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI'19)*. IEEE, 471–476. https://doi.org/10.1109/ISVLSI.2019.00091

[8] V. Catania, A. Mineo, S. Monteleone, M. Palesi, and D. Patti. 2015. Noxim: An open, extensible and cycle-accurate network on chip simulator. In *Proceedings of the IEEE 26th International Conference on Application-specific Systems, Architectures and Processors (ASAP'15)*. 162–163.

[9] Subodha Charles, Megan Logan, and Prabhat Mishra. 2020. Lightweight anonymous routing in NoC-based SoCs. In *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE'20)*. IEEE, 334–337.

[10] Subodha Charles and Prabhat Mishra. 2020. Lightweight and trust-aware routing in NoC-based SoCs. In *Proceedings of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI'20)*. IEEE, 160–167.

[11] Subodha Charles and Prabhat Mishra. 2020. Securing network-on-chip using incremental cryptography. In *Proceedings of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI'20)*. IEEE, 168–175.

[12] Kun-Chih Chen, Shu-Yen Lin, Hui-Shun Hung, and An-Yeu Andy Wu. 2013. Topology-aware adaptive routing for nonstationary irregular mesh in throttled 3D NoC systems. *IEEE Trans. Parallel Distrib. Syst.* 24, 10 (2013), 2109–2120. https://doi.org/10.1109/TPDS.2012.291

[13] Nizar Dahir, Terrence Mak, Alex Yakovlev, et al. 2013. Highly adaptive and deadlock-free routing for three-dimensional networks-on-chip. *IET Comput. Dig. Tech.* 7, 6 (2013), 255–263.

[14] Luka Daoud and Nader Rafla. 2018. Routing aware and runtime detection for infected network-on-chip routers. In *Proceedings of the IEEE 61st International Midwest Symposium on Circuits and Systems (MWSCAS'18)*. IEEE, 775–778.

[15] Luka Daoud and Nader Rafla. 2019. Analysis of black hole router attack in network-on-chip. In *Proceedings of the IEEE 62nd International Midwest Symposium on Circuits and Systems (MWSCAS'19)*. IEEE, 69–72.

[16] Luka Daoud and Nader Rafla. 2019. Detection and prevention protocol for black hole attack in network-on-chip. In *Proceedings of the 13th IEEE/ACM International Symposium on Networks-on-Chip*. 1–2.

[17] Daniel Engels, Markku-Juhani O. Saarinen, Peter Schweitzer, and Eric M. Smith. 2011. The Hummingbird-2 lightweight authenticated encryption algorithm. In *Proceedings of the International Workshop on Radio Frequency Identification: Security and Privacy Issues*. Springer, 19–31.

[18] Ramon Fernandes, César Marcon, Rodrigo Cataldo, Jarbas Silveira, Georg Sigl, and Johanna Sepúlveda. 2016. A security aware routing approach for NoC-based MPSoCs. In *Proceedings of the 29th Symposium on Integrated Circuits and Systems Design (SBCCI'16)*. IEEE, 1–6.

[19] Michel A. Kinsy, Michael Pellauer, and Srinivas Devadas. 2011. Heracles: Fully synthesizable parameterized MIPS-based multicore system. In *Proceedings of the 21st International Conference on Field Programmable Logic and Applications*. IEEE, 356–362. https://doi.org/10.1109/FPL.2011.70

[20] Jean-Jacques Lecler and Gilles Baillieu. 2011. Application driven network-on-chip architecture exploration & refinement for a complex SoC. *Design Autom. Embed. Syst.* 15, 2 (2011), 133–158.

[21] Kyle Madden, Jim Harkin, Liam McDaid, and Chris Nugent. 2018. Adding security to networks-on-chip using neural networks. In *Proceedings of the IEEE Symposium Series on Computational Intelligence (SSCI'18)*. IEEE, 1299–1306.

[22] Debdeep Mukhopadhyay and Rajat Subhra Chakraborty. 2014. *Hardware Security: Design, Threats, and Safeguards*. CRC Press, UK.

[23] Ahmad Patooghy, Maral Filvan Torkaman, and Mehdi Elahi. 2019. Your hardware is all wired up! attacking network-on-chips via crosstalk channel. In *Proceedings of the 12th International Workshop on Network on Chip Architectures*. 1–6.

[24] C. P. Ravikumar, S. Kendaganna Swamy, and B. V. Uma. 2019. A hierarchical approach to self-test, fault-tolerance and routing security in a Network-on-Chip. In *Proceedings of the IEEE International Test Conference India (ITC India'19)*. IEEE, 1–6. https://doi.org/10.1109/ITCIndia46717.2019.8979997

[25] Cezar Reinbrecht, Bruno Forlin, Andreas Zankl, and Johanna Sepulveda. 2018. Earthquake—A NoC-based optimized differential cache-collision attack for MPSoCs. In *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE'18)*. IEEE, 648–653. https://doi.org/10.23919/DATE.2018.8342090

[26] Cezar Reinbrecht, Altamiro Susin, Lilian Bossuet, and Johanna Sepúlveda. 2016. Gossip NoC—Avoiding timing side-channel attacks through traffic management. In *Proceedings of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI'16)*. IEEE, 601–606.

[27] Cezar Reinbrecht, Altamiro Susin, Lilian Bossuet, Georg Sigl, and Johanna Sepúlveda. 2017. Timing attack on NoC-based systems: Prime+ Probe attack and NoC-based protection. *Microprocess. Microsyst.* 52 (2017), 556–565.

[28] Eric Rescorla et al. 1999. *Diffie-hellman Key Agreement Method*. Technical Report. RFC 2631.

[29] Amin Sarihi, Ahmad Patooghy, Mahdi Hasanzadeh, Mostafa Abdelrehim, and Abdel-Hameed A. Badawy. 2021. Securing network-on-chips via novel anonymous routing. In *Proceedings of the 15th IEEE/ACM International Symposium on Networks-on-Chip*. 29–34.

[30] Amin Sarihi, Ahmad Patooghy, Ahmed Khalid, Mahdi Hasanzadeh, Mostafa Said, and Abdel-Hameed A. Badawy. 2021. A survey on the security of wired, wireless, and 3D network-on-chips. *IEEE Access* 9 (2021), 107625–107656. https://doi.org/10.1109/ACCESS.2021.3100540

[31] Johanna Sepúlveda, Daniel Flórez, and Guy Gogniat. 2015. Efficient and flexible NoC-based group communication for secure MPSoCs. In *Proceedings of the International Conference on ReConFigurable Computing and FPGAs (ReConFig'15)*. IEEE, 1–6. https://doi.org/10.1109/ReConFig.2015.7393301

[32] Johanna Sepúlveda, Andreas Zankl, Daniel Flórez, and Georg Sigl. 2017. Towards protected MPSoC communication for information protection against a malicious NoC. *Procedia Comput. Sci.* 108 (2017), 1103–1112.

[33] Gaurav Sharma, Soultana Ellinidou, Veronika Kuchta, Rajeev Anand Sahu, Olivier Markowitch, and Jean-Michel Dricot. 2018. Secure communication on NoC-based MPSoC. In *Security and Privacy in Communication Networks*, Raheem Beyah, Bing Chang, Yingjiu Li, and Sencun Zhu (Eds.). Springer International Publishing, Cham, 417–428.

[34] Gaurav Sharma, Veronika Kuchta, Rajeev Anand Sahu, Soultana Ellinidou, Suman Bala, Olivier Markowitch, and Jean-Michel Dricot. 2019. A twofold group key agreement protocol for NoC-based MPSoCs. *Trans. Emerg. Telecommun. Technol.* 30, 6 (June 2019), 1–18. https://doi.org/10.1002/ett.3633

[35] Gaurav Sharma, Veronika Kuchta, Rajeev Anand Sahu, Soultana Ellinidou, Suman Bala, Olivier Markowitch, and Jean Michel Dricot. 2019. A twofold group key agreement protocol for NoC-based MPSoCs. *Trans. Emerg. Telecommun. Technol.* 30, 6 (2019), 1–18. https://doi.org/10.1002/ett.3633

[36] Ebadollah Taheri, Mihailo Isakov, Ahmad Patooghy, and Michel A. Kinsy. 2020. Addressing a new class of reliability threats in 3D network-on-chips. *IEEE Trans. Comput.-Aided Design Integr. Circ. Syst.* 39, 7 (2020), 1358–1371. https://doi.org/10.1109/TCAD.2019.2917846

[37] Jie Zhang, Feng Yuan, and Qiang Xu. 2014. DeTrust: Defeating hardware trust verification with stealthy implicitly-triggered hardware Trojans. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*. ACM, New York, NY, 153–166. https://doi.org/10.1145/2660267.2660289

[38] Gang Zhou, Yafeng Wu, Ting Yan, Tian He, Chengdu Huang, John A. Stankovic, and Tarek F. Abdelzaher. 2010. A multifrequency MAC specially designed for wireless sensor network applications. *ACM Trans. Embed. Comput. Syst.* 9, 4, Article 39 (Apr. 2010), 41 pages. https://doi.org/10.1145/1721695.1721705

[39] Jun Zhou, Huawei Li, Tiancheng Wang, and Xiaowei Li. 2016. LOFT: A low-overhead fault-tolerant routing scheme for 3D NoCs. *Integr. VLSI J.* 52 (2016), 41–50.

Received 1 March 2022; revised 30 December 2022; accepted 13 March 2023