Enhancing Security Features of Network-on-Chip Using Lightweight Cryptosystem, Trust-Aware Routing, and Anonymous Communication

Syam Sankar

Supervisor: Dr. John Jose

Thesis submitted in partial fulfilment
of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science and Engineering
INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

January 2025

I would like to dedicate this thesis to my parents, my brothers, and my wife. . . .

Declaration

I hereby declare that, except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this or any other university. This dissertation is my own work and contains nothing that is the outcome of work done in collaboration with others, except as specified in the text and acknowledgements.

January 2025 Syam Sankar

Certificate

This is to certify that the thesis titled **Enhancing Security Features of Network-on-Chip Using Lightweight Cryptosystem, Trust-Aware Routing, and Anonymous Communication**, submitted by **Syam Sankar** to the Indian Institute of Technology Guwahati in partial fulfilment of the requirements for the degree of Doctor of Philosophy, is a record of bonafide research work done under my supervision. To the best of my knowledge, the contents of this thesis have not been submitted in whole or in part for consideration for any other degree or qualification in this or any other institute or university. Therefore, this thesis is worthy of consideration for the award of the degree of **Doctor of Philosophy**.

January 2025

Dr. John Jose
Thesis Supervisor
Associate Professor
Department of Computer Science Engineering
Indian Institute of Technology Guwahati

Acknowledgements

It has been a wonderful journey of technical enrichment as a research scholar at IIT Guwahati (IITG). I was lucky enough to work with a great teacher, researcher, and mentor, Dr. John Jose, my PhD supervisor. I am expressing my profound thanks to Dr. John Jose for his invaluable advice and encouragement, which enabled me to think critically and investigate research problems throughout my PhD period. I was often amazed by how he treated his students and the many interesting things we talked about. I was also fortunate to be part of all his NPTEL courses, of which I'm very privileged now. I was part of his Multicore ARchitecture and Systems (MARS) Lab in the department of Computer Science and Engineering (CSE) at IIT Guwahati. My research work was in alignment with a project titled "Security Features of On-Chip Networks in Modern Multicore Processors," sponsored by SERB-DST, Govt. of India, in which Dr. Jose was the principal investigator and Prof. Sukumar Nandi was the co-principal investigator. As a result, I had many fruitful interactions with Prof. Nandi, discussing various aspects of research objectives and their solutions. I am grateful for all the time he spent with me. Every discussion concluded with a brilliant idea, ultimately contributing to my thesis. I take this opportunity to express my gratitude to Prof. Nandi. I also had the opportunity to work with Dr. Ruchika Gupta. She was a postdoctoral fellow with Dr. Jose. She was also helpful in giving suggestions and helping me write manuscripts. I sincerely acknowledge her kind help for me.

I am a faculty member in the CSE department at NSS College of Engineering (NSSCE), Palakkad, Kerala. In order to pursue my PhD, I joined IITG as a QIP scholar. Since QIP is an All India Council for Technical Education (AICTE) initiative, I would like to express my gratitude to AICTE for this wonderful initiative so that teachers in technical institutions can go for higher studies. I thank NSSCE management, college authorities, and the Directorate of Technical Education, Govt. of Kerala, for giving me consent to join the PhD programme at IITG in QIP mode. I had been selected for the prestigious Prime Minister's Research Fellowship (PMRF) in the December 2021 cycle. Fortunately, I could be the first one to receive the PMRF from IITG's CSE Department. I thank Prof. T. Venkatesh and Dr. Jose for the support I received while applying for PMRF. I thank the PMRF National Coordination

Committee (NCC) for this amazing recognition. I also thankfully acknowledge PMRF review committee members from various IITs and IISc for their insightful and critical comments.

I also thank the doctoral committee (DC) members Prof. Diganta Goswami (DC Chair), Prof. T. Venkatesh, and Dr. Sreenath J. G. for the insightful suggestions. The MARS lab to which I belong provided a great working environment. During my tenure, I could interact fruitfully with all the research scholars in the lab.

In the end, and with tremendous significance, I am very grateful to my parents, my brothers, and my parents-in-law for their unconditional encouragement and support. I would like to give special appreciation to my spouse, Haritha M, for taking care of everything and managing my new-born son, Vedant Harisankar, even in my absence. She is always a true source of care, love, and support. Once again, I acknowledge everybody who is part of my research journey at IIT Guwahati.

Abstract

Multi-Processor Systems-on-Chips (MPSoCs) combine multiple hardware Intellectual Property (IP) components on a single chip. Modern SoC vendors use 3rd-party IPs to develop in-house chips in order to reduce design costs and compete with time-to-market constraints. With the rising popularity of fabless manufacturing and agile development, the use of third-party IP for design is becoming more accepted. However, such practice can sometimes significantly compromise the security and reliability of SoC computing systems. Modern MPSoCs, or Tiled Chip Multi-Processors (TCMPs), employ multi-hop packet-based Network-on-Chip (NoC) as their communication backbone due to their low-latency, high-bandwidth, and scalable topology. Adversaries use NoC IP as a major platform to launch security attacks and degrade SoC performance due to its close proximity and location criticality with respect to inter-core communication. Malicious circuits, such as Hardware Trojans (HT), use NoC as a carrier to float attacks such as eavesdropping, application profiling, packet modification, Denial-of-Service (DoS), and so on. Since on-chip communication plays a critical role in determining the SoC's performance, any HT attacks targeting this communication may impact the smooth running of applications on the processor cores or may even leak sensitive SoC data to external adversaries. This thesis details three countermeasures to enhance the security features of on-chip communication using lightweight cryptosystem, trust-aware routing, and anonymous communication against HT attacks. These three measures defend packet leakage, DoS due to packet alteration, and application profiling, respectively.

This thesis, in its first contribution, targets building lightweight cryptosystems with two primary goals: not even a single NoC packet should reach a colluding application, and the average packet latency should be minimal compared to the state-of-the-art methods. The second contribution focuses on a direct-trust relationship between NoC routers as opposed to trust building with delegated trust packets in the existing approach. We prove that the proposed trust relationship allows more NoC packets to bypass the HT routers without creating any extra traffic. The final contribution highlights the need for a new anonymous communication framework in NoC. The existing anonymous communication frameworks fail to preserve packet anonymity with the proposed traffic-profiling-capable HTs. Our approach thoroughly explores the packet diversity in NoC to defend such an HT. The thesis shows that the three works can significantly sustain the SoC performance in a secure environment with justifiable area and power overheads.

Table of contents

List of figures

List of tables

List of Acronyms

Acronym	Expansion
CMP	Chip Multi-Processor
TCMP	Tiled Chip Multi-Processor
SoC	System-on-Chip
MPSoC	Multi Processor System-on-Chip
IP	Intellectual Property
QoS	Quality of Service
NoC	Network-on-Chip
HT	Hardware Trojan
DoS	Denial-of-Service
VC	Virtual Channel
MPKI	Misses Per Kilo Instruction
TROP	TRust-aware OPportunistic Routing
SARON	Secure Anonymous ROuting protocol for NoC
Sec-NoC	Secure NoC
3PIP	Third-party Intellectual Property
MRCA	Malicious Router and Colluding Application
PCR	Packet Corruption at Routers
RBUFF	Re-transmission buffer
NI	Network Interface
MR	Malicious Router
ACK	Acknowledgement
NACK	Negative Acknowledgement
SRS	Source Router Set
ReTAB	Retransmission Table
CA	Colluding Application
MAC	Message Authentication Code

Chapter 1

Introduction

The paradigm switch from uni-core to multi-core systems [1] has been caused by a number of scenarios, including Moore's Law [2], the Power Wall Problem [3], and the limits of instruction-level parallelism (ILP) [4]. Thus, a single die that incorporates several cores operating at lower frequencies, called Chip Multi-Processor (CMP) [5], results in improved parallelism, decreased power consumption, and enhanced overall performance. Later CMPs evolved into Tiled Chip Multi-Processors (TCMPs), as they comprise a 2D grid of identical compute elements, referred to as tiles. Each tile is a powerful computing unit containing a processor core and caches[6]. Intel Xeon Phi 7200-series TCMPs contain up to 72 cores, and AMD's EPYC 9754 CMP houses up to 128 cores. According to the 2015 ITRS (International Technology Roadmap for Semiconductors) plan [7], there will be thirty times as many cores by 2029 because of the growing need for processing power. With more and more cores packed onto a single die, there must be a reliable, efficient, and scalable way to interconnect them. The Network-on-Chip (NoC) [8] has evolved as a remedy for the same by replacing bus-based, crossbar-based, and ring-based communication to connect all of the available tiles.

The NoC supports communication in the form of packets (packet switched network), and it basically serves cache misses and coherence messages. It connects all the tiles in a TCMP to each other via links and routers. A topology of routers connected by point-to-point links builds up an interconnection network. Fig. 1.1 shows a TCMP with 16 tiles, interconnected with the NoC framework following a mesh topology [9]. Each tile contains a processor core, L1 cache, shared L2 cache, network interface, etc. Owing to the significance of NoC in ensuring the performance of TCMP, extensive research has been done on router microarchitecture [10], [11], power optimisation [12], and topologies [13], [14]. In fact, it has been observed that NoC is responsible for 60% to 75% of applications' cache miss latency [15]. The average packet latency in NoC is a crucial parameter that impacts the

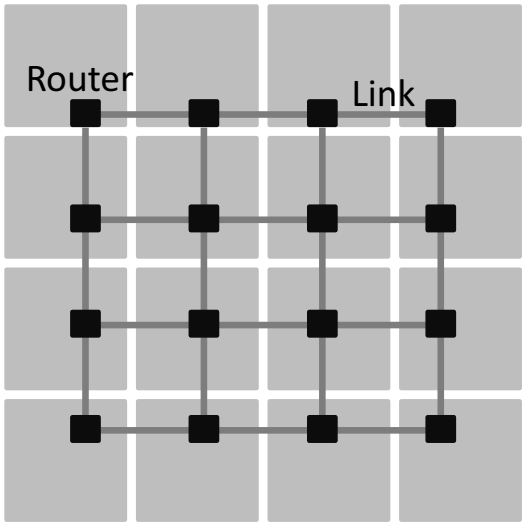


Fig. 1.1 A TCMP design with 16 tiles interconnected with a 2D mesh NoC.

functioning of the applications running on the core. The nature of traffic and the router delay at each hop primarily influence the packet latency.

Sonics, Intel, and Arteris are the top three companies in the NoC market that aim to provide NoC-based solutions for efficient and reliable communication among various components of a Multi-Processor Systems-on-Chips (MPSoC), or simply System-on-Chip (SoC). The growing integration of numerous cores and intellectual property (IP) blocks inside an SoC has accelerated the adoption of NoC solutions. From basic handheld devices to sophisticated supercomputers, SoC is now an inherent feature for both computation and communication. For instance, 100–200 different IP blocks from various third-party manufacturers may be included in a standard automotive SoC [16]. Two interconnect IPs from Arteris IP, namely FlexNoC and NCore, are used in several SoCs. STNoC and Tilera's iMesh are the other two commercially available NoC architectures [17]. Sondrel recently announced that they employ Arteris' FlexNoC IP [18] as their SoC's NoC communications backbone. Fig. 1.2 depicts an SoC in which IP cores such as memory, processors, controllers, etc. are interconnected through an NoC. NoC IP adoption is increasing as a result of its application on a wide range of devices, including tablets and mobile phones.

Modern SoC vendors use 3rd-party IPs to develop in-house chips in order to compete with time-to-market constraints. With the rising popularity of fabless manufacturing and agile development, the use of third-party IP for design is more of an accepted practice [19]. However, it can sometimes lead to security breaches that can also degrade the SoC's performance. Adversaries from untrusted third-party IP providers can use NoC as a major

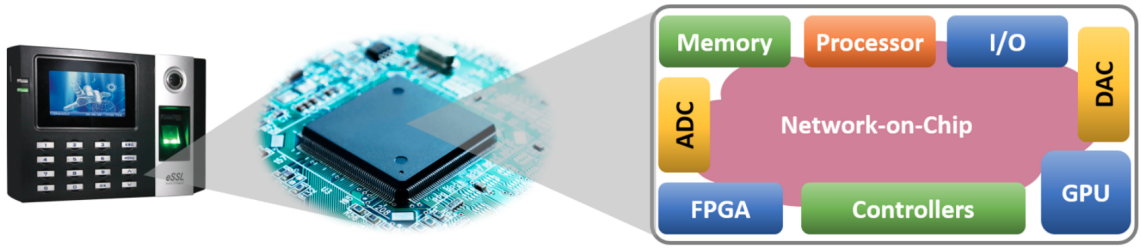


Fig. 1.2 An SoC comprising IP blocks interconnects with the NoC framework.

platform to launch attacks and degrade SoC performance, as it closely aligns with the functional specifications and chip floor-plan architecture. Therefore, these third-party NoC IPs may intentionally incorporate harmful implants, such as hardware Trojans (HT), to introduce unwanted functionality into the SoC. Hardware-level vulnerabilities are exploited by the attackers (or adversaries) to launch various forms of critical attacks like Rowhammer [20], Meltdown/Spectre [21], CacheOut [22], and Load Value Injection [23]. According to the literature, removing vulnerabilities at the hardware level improves overall system security [16]. Since NoC-based attacks are prominent in SoC, NoC designs incorporate security services too, along with the features for quality of service (QoS) and fault tolerance. In the early days of NoC, the emphasis was on electrical NoC, or the electrical lines that joined NoC components together. In recent years, new NoC technologies, including wireless NoC [24] and optical NoC [25], have emerged. Most security solutions are also relevant to wireless and optical NoCs, even though the emphasis of this thesis is on security threats and remedies in electrical NoCs. Since this thesis primarily targets NoC-based attacks by HTs, we shall discuss that further.

1.1 Security in Network-on-Chip based SoC

The current SoC design is vulnerable to numerous security threats. In an MPSoC, different applications that run on the cores send and receive data through the NoC in plaintext form [26]. Given that the data of applications is in its raw form as it passes through the NoC, the first security vulnerability is the potential extraction of the applications' data (eavesdropping attack) by abusing the NoC communication [27]. An HT and a colluding application (CA) that run on a core can establish a covert communication channel to launch an attack of this type [28]. In order to carry out these security breaches, attackers had to implant an HT in a router. Because of its dormant nature, HT's actions may remain undetected during post-silicon testing. Only during runtime, the HT receive a CA trigger to execute its malicious

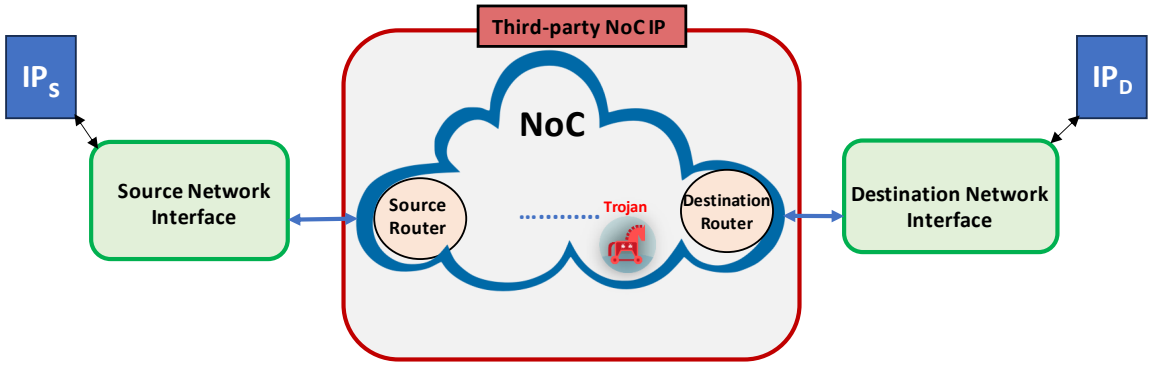


Fig. 1.3 Overall threat model: The communication of two IP cores over an unsecured interconnect framework: HTs in NoC to downgrade SoC's functionality.

activity. Furthermore, a CA has the capability to execute a denial of service (DoS) attack [29], [30] on the NoC by injecting unnecessary packets that slow down the NoC, resulting in a decline in the overall system performance. Flooding the packets consumes the available bandwidth at the NoC level in a DoS attack, resulting in deadlock and livelock situations that prevent applications from running on the cores. HT can also cause the SoC to degrade through packet modification. Despite enabling mechanisms to address this integrity violation, multiple HTs at run-time can even attack the recovery packets, such as ACKs and NACKs, which in turn generate more traffic, resulting in congestion at the network. The scenario worsens when multiple HTs at routers are activated at random during runtime.

The NoC communication is vulnerable to side channel attacks as well, particularly information leakage through timing-based attacks [31], [32]. Cache hits and misses cause noticeable variations in total execution time. This enables the attackers to monitor it through NoC traffic flow. Researchers propose the first practical timing attack on MPSoC using the Prime+Probe approach [33]. Here, the spy process and the victim process operate on distinct IP cores, and the NoC's traffic throughput is monitored to leak critical information. A compromised MPSoC setup can even worsen Bernstein's cache timing attack, which explains the AES vulnerability [34]. A black hole router attack [35] can drops packets without sending them to the next hop, triggering retransmission and throughput degradation. This causes significant damage to application functionality. The most popular countermeasures to mitigate the attacks discussed above fall into four categories [36]: firewalls [37], [38], [39], [40], [41], security zones [42], [43], [44], [45], encryption [46], [47], [48], [49], [43], [50] and secure routing [51], [52], [53].

Fig. 1.3 depicts the overall threat model in which two IP cores communicate through an untrusted NoC IP that contains a malicious HTs with the intention to induce security threats

and attacks, including sensitive data leakage or integrity violations. Each packet that enters the NoC is associated with certain components that facilitate packet movement. We refer to the IP core that generates the packet as the source IP, and the IP core that receives it as the destination IP. The source network interface divides the packet into flow control units (flits), subsequently injecting them into the NoC. The packet routing process begins at the source router. To reach the destination router, the packet takes multiple hops (routers) by following an underlying routing method. The destination router forwards the packets to its associated network interface, which then combines the flits into a packet. The destination IP core finally receives the packet.

The thesis adheres to this general threat model, where we assume the presence of HTs at the router with malicious payloads such as packet leakage, packet alteration and traffic profiling. The thesis considers three secure design aspects of NoC: the first method proposes a lightweight secure NoC communication to prevent packet leakage along with an integrated key-encapsulation mechanism; the second method enables NoC packets to follow trusted paths without being impacted by the HT capable of packet alteration; the third method integrates anonymity into packet transmission to block traffic profiling.

1.2 Problem Statements and Summary of Solutions

This section provides a concise summary of the problems addressed and proposed solutions as part of the thesis. The following chapters discuss each of them in greater detail.

1.2.1 Lightweight Security Framework

We assume the presence of an HT that is capable of both packet data leakage and packet modification. A covert communication is established between an HT router and a CA to leak sensitive content from packets. The HT, which is intermittently triggered by the CA, transmits a copy of the packet communicated between target cores to the IP core, where the CA is running. Our experiment shows that when all of the routers are targeted (only once) by HT to leak packets originated from it, an average of 28% of the packets leak from a 4×4 mesh NoC. The leaked packets eventually reach the colluding application if no security services are integrated. As soon as a packet containing sensitive information reaches an attacker, it can apply any technique, like ML-based algorithms, to extract useful information. Even if packets are encrypted, the attacker can still apply complex cryptanalysis to extract the required data from them. To eliminate packet leakage from the NoC, the only solution is to implement cryptographic techniques such as integrity checks and authentication at

the network interface. We use these techniques to ensure that the network interface blocks every leaked packet from HT routers. We achieve this by taking into account the fact that every leaked packet is a modified packet. To implement this and block packet leakage, we propose a lightweight **Sec**ure **NoC** (Sec-NoC) framework that integrates the notions of lightweight symmetric cryptography using PRINCE cipher to facilitate confidentiality, elliptical curve cryptography to facilitate key exchange mechanisms and to ensure secret key origin authentication, and a lightweight MAC producer using SipHash to facilitate integrity. The network interface invokes all these security services. Sec-NoC transmits symmetric encryption keys as special NoC packets through Key Encapsulation. The encryption key is enciphered using public-key encryption, encapsulated inside a packet, and sent over the network to the designated recipient. We use the ACK-NACK method at the NI level [54] as an error recovery measure. In addition to blocking packet leakage, we also focus on reducing average packet latency. In terms of average packet latency, Sec-NoC outperforms NoC with AES-128, Fort-NoC [50], and P-Sec [55] by 87%, 77%, and 73%, respectively. Due to Sec-NoC's integrity check, 100% leaked packets get blocked at the network interface itself. Sec-NoC is covered in depth in **Chapter 3**.

1.2.2 Trust-aware Routing

Time-critical applications cannot tolerate significant NoC traversal delays. Attacks, such as packet modification by HT during traversal, may cause a significant delay for a packet to reach its destination due to integrity checks, acknowledgement packets and re-transmissions. Error correction through retransmissions can cause traffic hike in NoC when there are many HTs. This is because an active HT can even alter the retransmitted packets. NoC traffic can consume more energy too. This can happen if the underlying routing is static, like XY routing. The continuous retransmission of packets can even lead to a denial-of-service attack in NoC. To address this issue, the only option is to reduce the rate of re-transmissions. To solve this problem, we can choose a downstream router based on trust in the route computation stage. The packet routing based on trust facilitates the selection of a downstream router that is likely to be free from HT. The history of the packets' re-transmission patterns builds trust between adjacent routers. Future packets use this trust information to determine the most trustworthy downstream router. We propose a mechanism called **TR**ust-aware **OP**portunistic Routing (TROP) with the aim of making decisions based on trust when choosing the next best neighbouring router opportunistically in order to decrease the amount of packets passing through HT routers. This in turn reduces the packet re-transmission rate. By preventing re-transmitted packets from following the same route, the mechanism reduces the possibility of HT modification. The TROP works locally by selecting the most trusted downstream router

for a packet to route. The existing trust-based routing [56] uses a trust delegation approach in which trust-packet broadcasting happens between adjacent routers. The motivation for TROP stems from the fact that altering these trust packets results in the selection of untrustworthy output ports. As a result, TROP avoids these trust delegations and creates an accurate direct trust building between adjacent routers. An experimental study demonstrates that TROP lowers the rate of re-transmissions by up to 40% for certain workloads when compared with existing trust-based routing and also exhibits an energy savings of 12%. **Chapter 4** provides a comprehensive explanation of TROP.

1.2.3 Secure Anonymous Routing

Sending packets anonymously makes it more difficult for attackers to identify the parties conversing. As a result, attackers are unable to decode the properties of the applications running on the cores. In an attempt to break the NoC's anonymous packet transmission, we propose a new threat model involving HTs at routers that can perform flow-based traffic profiling and launch targeted performance degradation attacks. Traffic profiling entails identifying the packet's precise source-destination pair, either through direct content analysis or traffic analysis. Typically, we anticipate that a secure, anonymous system prevents targeted attacks by concealing the identities of the communicating parties. However, the proposed HT type's traffic profiling capability breaks the anonymity and identifies the source-destination (SD) pair of the NoC packet. With our attack model, we prove that two existing NoC's anonymous systems [51]-[52] are susceptible to traffic profiling and failed to preserve source anonymity during NoC packet transmission. We propose a safe and anonymous NoC system called SARON (**Sec**ure **A**nonymous **RO**uting for **N**oC Communication) to protect against the suggested attack model. It combines the benefits of dynamic routing, route diversity, and an encrypted source information in the packet header. These three ingredients make the HT fail to predict the SD pair of packets with 100% accuracy. As a defense against sniffers, SARON provides lightweight end-to-end data encryption. We employ a deadlock-free dynamic routing method known as CFS routing. We explore the route diversity such that packets can be transmitted securely over different paths each time they are sent between an SD pair. The SARON defines the source router set (SRS) with respect to the router port and the underlying routing. The SRS plays a significant role in blocking flow-based traffic profiling and ensuring anonymity. We observe that SARON reduces the targeted packet leakage rate in an anonymous environment by 88% when compared with a state-of-the-art anonymous system. A detailed description of SARON is given in **Chapter 5**.

1.3 Overview and Thesis Contributions

Here, we summarise the general overview and salient features of the thesis and its potential impacts as far as building a secure NoC is concerned.

1. **Literature survey on HT-based attacks on NoC**

   We perform a state-of-the-art literature survey on the vulnerabilities of NoC communication and the impact of HT attacks on NoC-based SoCs. A background on the features of NoC communication and HT is also presented along with the security study. Chapter 2 illustrates this in detail.

2. **Lightweight security system blocking leaked packets from reaching any colluding application.**

   We use lightweight security ciphers at NIs instead of implementing computationally intensive ciphers. We investigate the fundamental fact that leaked packets are always modified packets. So, an integrity check can prevent the leaked packets from reaching the colluding application. Only unmodified packets can advance to the upper level through the Sec-NoC's integrity check at the destination NI. Chapter 3 [57] discusses the ways of implementing the most lightweight security system in the NoC framework.

3. **Usage of key-encapsulation method in NoC other than key-agreement protocol.**

   Key encapsulation is often simpler to implement and more efficient in terms of computational and memory resources in NoC than the existing key-exchange protocols. This is well explored in Chapter 3 [57].

4. **Tolerate PCR (Packet Corruption at Router) effects in interconnect during runtime.**

   When multiple HTs at routers trigger randomly and have the potential to corrupt packets, the problem becomes more challenging. Repeated packet corruptions and subsequent re-transmissions make the network congested. As a result, regular packet movement slows down, affecting applications. The only way to handle such a situation is to bypass the Trojan-impacted routers during runtime. We can make this bypassing possible by building trust between the routers. TROP regulate the packet latency to prevent it from escalating due to HT impacts. Chapter 4 discusses the ways of building trust and possible HT bypassing [58], [59].

5. **Dynamic trust building between adjacent routers.**

   Due to HT's intermittent behaviour, trust values fluctuate. A low trust value at some point in time indicates a router with triggered HT, and a high trust value at some point in time denotes either a router with dormant HT or no HT. To facilitate bypassing when we encounter low trust values, a deadlock-free dynamic routing has to be employed. Chapter 4 illustrates this in detail [58].

6. **Preferring adaptive routing to fight against PCR and MRCA (Malicious Router and Colluding Application).**

   A static routing algorithm, such as XY routing, is the most popular choice for routing packets in the interconnect. For example, Arm's CI-700 interconnect [60] uses XY routing as the default routing method in a mesh network. In terms of security, XY routing can be considered the least preferred as it provides a single routing path between any pair of communicating IPs. Therefore, the presence of any HT in that path affects the packets until the HT becomes dormant. With the XY routing method, both packet alteration and traffic profiling (MRCA) are severe. To circumvent it, we need an adaptive routing approach. Both Chapter 4 [58] and Chapter 5 [61] illustrate this in detail.

7. **Exploring path diversity to block traffic profiling.**

   We use anonymous packet transmission to prevent external attackers from profiling applications running on the system by exploiting NoC communication. It is believed that encrypting packet headers makes it more challenging for HTs to profile packets and decode the source-destination (SD) pair of packets for more focused attacks. However, we demonstrate with our proposed HT that, even without decrypting packet headers in existing anonymous NoC frameworks, we can identify the SD pair of a packet with higher accuracy due to the nature of topology, characteristic features of routing algorithms, and predictable patterns of packet movement across routers. Unlike internet communications, where the anonymity set is extremely large, the NoC's anonymity set is very small. Existing anonymous frameworks have not considered this fact. Since the set is small, the only solution to preserve anonymity is to explore path diversity. That is, packets can be transmitted securely over different paths each time they are sent between an SD pair. This also achieves effective load distribution between an SD pair. Chapter 5 [61] and [62] goes into more depth on this.

8. **Develop the features of an anonymous routing in NoC.**

   Routing is an essential component of any system that uses anonymous communications. Unlike the conventional approach of transmitting data from source to destination along a single path, our routing policy transfers data through various paths to reach its destination. In order to make it challenging for the adversary to infer the SD pair of the packet, we propose three desirable properties in the routing scheme, along with a lightweight encryption technique. This is well explored in Chapter 5 [61].

1.4 Chapter Summary

Over time, interconnect systems have progressed from basic buses to more scalable NoC IP. Since NoC is what drives SoC's or TCMP's performance, researchers pay close attention to their security requirements. Adversaries from unreliable third-party IP providers can use NoC as a significant platform to implant malicious circuits like HT. Since it is directly related to the chip's functional specifications and floor plan design, any security breaches would malfunction the whole SoC itself. So, our thesis studies three important security problems caused by HTs and identifies the gaps in the existing solutions. We show that, with negligible overheads, the proposed approaches can ensure the secure transfer of NoC packets, achieving the best trade-off between performance and security. Fig. 1.4 summarises our contributions towards designing a secure NoC communication system against various HT attacks.

In the remainder of this thesis, Chapter 2 presents relevant background on NoC and security attacks on it. It also details the overall simulation framework. Chapter 3 details the implementation of a lightweight security system in NoC. Chapter 4 details our work on trust-aware routing on NoC. Chapter 5 continues with a discussion of how to preserve anonymity during packet transmission. Finally, Chapter 6 concludes the thesis and discusses possible future research directions.

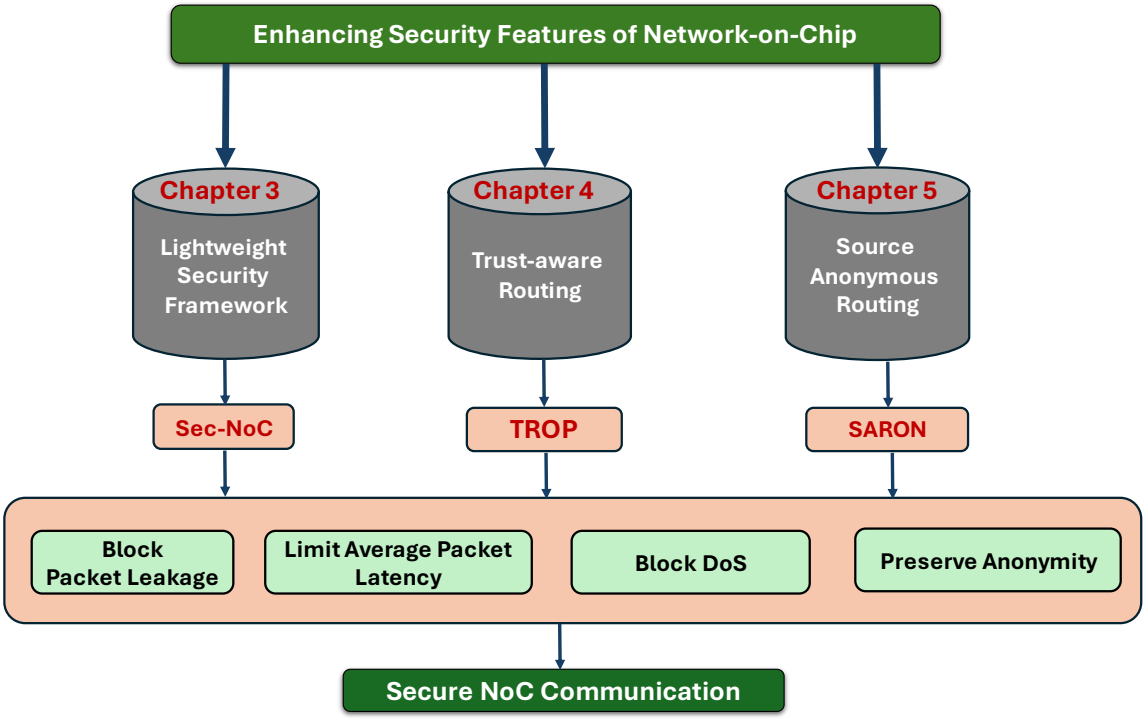


Fig. 1.4 Summary of contributions in the thesis.

Chapter 2

Background and Literature Survey

This chapter presents the key features of the NoC communication infrastructure to enable the reader to understand the thesis. The chapter also provides the necessary background on the HT and its impact on NoC communication. We primarily focus on two NoC attacks by Trojans: eavesdropping and data integrity violations. Finally, we present an evaluation methodology that explains the tools and system parameters that are used to evaluate the performance of proposed the solutions in the thesis.

2.1 Network-on-Chip Overview

MPSoCs are multi-tenant complete computational systems that integrate multiple hardware IP elements in a chip [63]. These IP components are configured to meet the different criticality and performance requirements in various domains like IoT applications, video applications, automotive and aerospace applications, etc. The state-of-the-art high-end MPSoC, such as the TILE-Mx100 from EZchip, is a 64-bit ARMv8 processor with 100 cores. Computation, storage, and communication are the three main structural parts associated with each MPSoC. Our focus is on communication, which involves the data exchange among the different IP cores (processor, memory, controllers, converters, input/output devices, peripherals, and so on). These IP core components are either from third-party vendors or developed in-house. The MPSoC structure can be organised in a tile-based manner, where each tile is comprised of a single or multiple computations and storage IP cores. These are also called TCMP. Intra-tile communication is established through a bus structure. Inter-tile communication can happen through crossbars or NoCs [64]. NoCs, the programmable interconnection systems [65] [66], are structures that use routers and links to facilitate packet transfer between a source IP that injects packets and a destination IP that receives data packets. The NoC primarily serves cache misses, memory request, and coherence requests. The efficient usage of interconnection

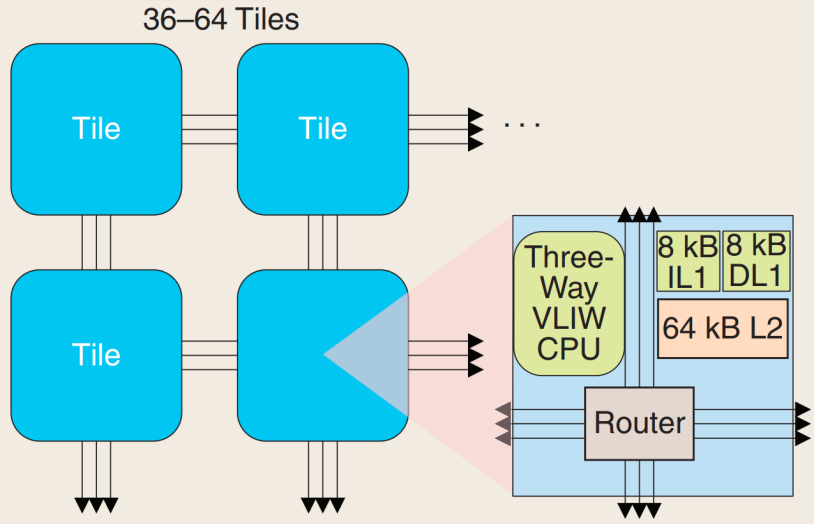


Fig. 2.1 Tilera TILE64 block diagram

frameworks is already proven in IBM's TrueNorth (4096 cores are interconnected) chip and in Altera's Arria 10 SoC FPGA [67]. Fig. 2.1 depicts the block diagram of Tilera's TILE64 multi-core processor [68]. The TILE64 is the first implementation of the tiled processor architecture in which 64 tiles are connected by a large NoC. Each tile houses a Very Long Instruction Word (VLIW) core, caches, and a router. A directory-based coherence policy is followed in this DSP-focused processor. The design of any NoC comprises various building blocks, such as topology, routing, flow control, router micro-architecture, and link architecture. NoC IP usage is exponentially increasing due to its ubiquitous availability as an essential IP component in various mobile phones, tablets, and automotive devices. Fig. 2.2 shows a general MPSoC architecture with a 4x4 mesh NoC, where each tile consists of a processing core, caches, memory controllers, network interface, etc. Routers connect these tiles in a 2D mesh fashion.

This MPSoC tile occupies a private L1 cache and a shared L2 cache, similar to the Intel KNL [69] and AMD Thread-ripper architectures. L2 cache is shared and spread over all 16 tiles. During an L1 cache miss, the request message is forwarded from the source tile of the L1 cache to the L2 cache tile to fetch the required cache block. The source tile receives a response message containing the required cache block. If there is an L2 miss, a pair of request and reply messages are exchanged between the L2 tile and the main memory controller before forwarding the requested cache blocks to the L1 tile. Instruction Fetch (IF), load (LD), and store (ST) operations lead to data movement between various levels of the memory hierarchy. IF and LD miss at the L1 cache trigger control messages to flow across

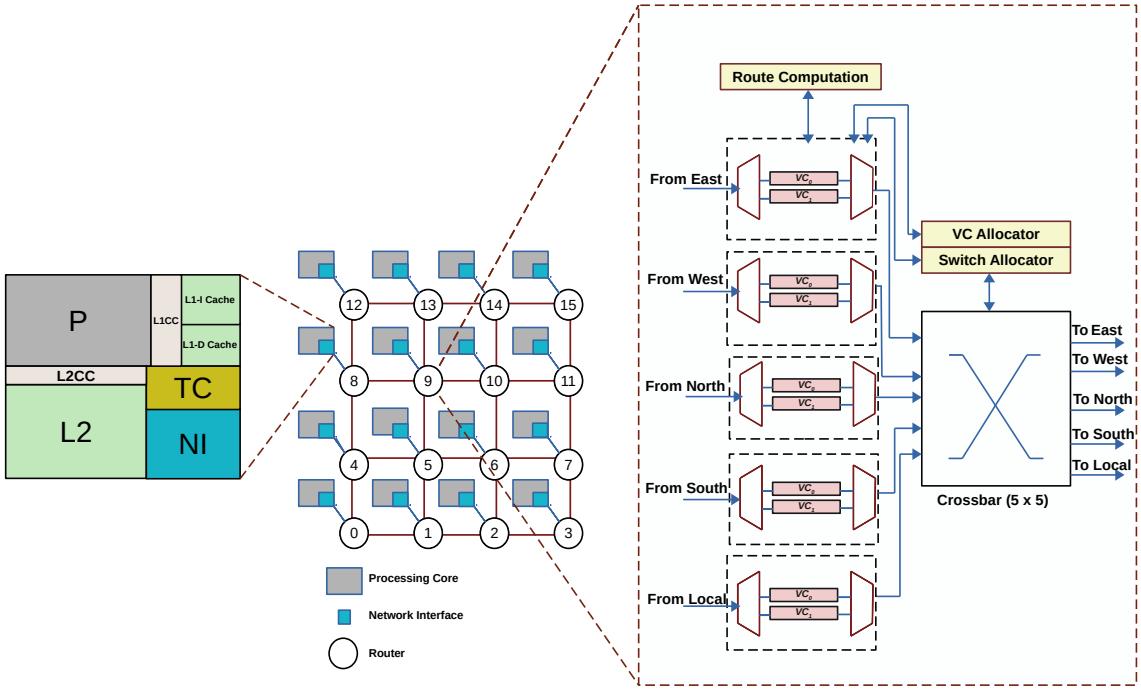


Fig. 2.2 MPSoC with 16 processor tiles organised as 4x4 mesh NoC along with its router micro-architecture. Each tile comprises a processor core (P), L1 cache controller (L1CC), L2 cache controller (L2CC), L1 instruction and data cache (L1-I and L1-D), L2 caches (L2), and a tile controller (TC).

the network, while ST misses trigger data messages. Additionally, messages related to cache coherence protocols also contribute to the NoC traffic. The network interface (NI) segments a message into packets, then further divides it into fixed-length flow control units known as flits. In general, a message is encapsulated in a single NoC packet. The NI split the packets into flits according to the flit channel width, say 128 bits. A data packet is a combination of a head flit, two or more body flits, and a tail flit, whereas control packets consist only of a single head-tail flit. A miss request packet consists of a single head flit, whereas a reply packet consists of a head flit followed by multiple body flits and a single tail flit (H, B_1, B_2,..., B_n, T). NoC follows a wormhole switching [70], where body/tail flits follow head flits. The breakdown of messages into packets and packets into flits is depicted in Fig. 2.3. Each flit contains some basic information, such as flit type (FT), virtual channel ID (VCID), and so on. A head flit primarily carries the address, and the cache line is distributed across the body and tail flits. Request, forward, response, and unblock are just a few of the message types (message classes) that are often used by the underlying cache coherence protocols. Usually, control messages (request, forward, and unblock) fit within a single flit, while data messages (response) require multiple flits. While using a 128-bit channel width by the NoC framework,

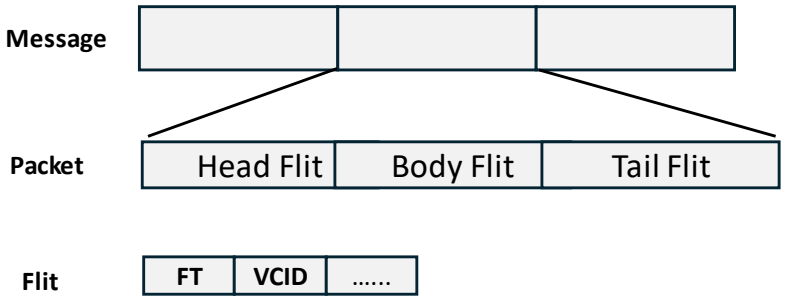


Fig. 2.3 Message Composition: Flit type (FT) and virtual channel ID (VCID).

a 64-bit addressing, and a 64-byte cache line by the cache-coherent system, a control packet requires 1 flit, and a data packet requires 5 flits.

A state-of-the-art 5-port NoC router microarchitecture is also shown in Fig. 2.2. It consists of input port buffers, a crossbar, a virtual channel allocator (VCA), a route computation unit (RC), and a switch allocator (SA) [71]. The input port buffers are organised as virtual channels (VCs), and each port contains one or more VCs. VCs receive incoming packets before forwarding them to the downstream router. VCs are essential in NoC flow control to prevent head-of-line blocking (HoL) [72]. The router's RC stage determines the output port to move to the downstream router based on the underlying routing technique and destination information from the head flit. The VCA assigns the free virtual channel from the downstream router to the current packet. The SA arbitrates among the flits that request the same output port. It uses mostly either round-robin or ageing as the underlying arbitration policy. Finally, the crossbar switch physically transfers the winning flits from the input port to the output port. In modern interconnect, NoC routers can be designed with 5-stage, 2-stage, or even 1-stage pipelines [73], [74], [75]. Protocol-level deadlock is prevented by requiring messages from different message classes to use distinct sets of buffers (VCs) inside the network. Virtual networks (VNETs) are used to do this. Each router may have more than one VNET, and each VNET may contain one or more VCs.

An NoC topology determines the physical arrangement of routers and links so that they connect the tiles (or processing elements) in various ways. Three common topologies, such as ring, mesh, and torus, are shown in Fig. 2.4. The thesis considers the 2D mesh topology for all its studies. Most of the popular industry-based interconnects are mesh-based. Therefore, we adhered to this topology in our work. For packet routing, mainly two types of schemes are available: deterministic and adaptive. XY routing is one of the popular deterministic routing algorithms in the NoC framework. With XY, packets take the X direction, followed by the Y direction, to reach the desired destination router. We also use the adaptive routing

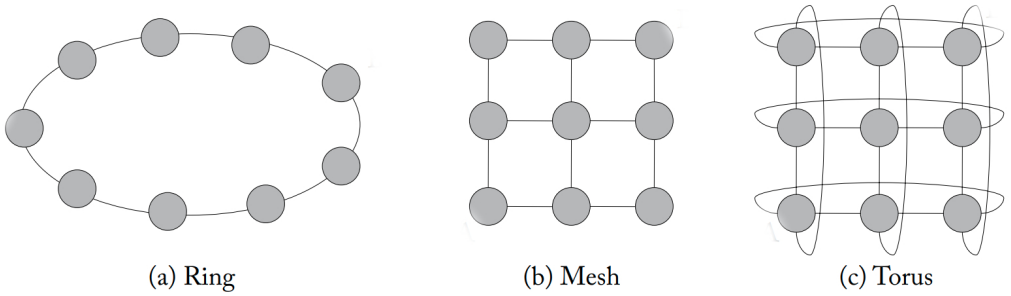


Fig. 2.4 NoC Topologies

algorithm DyXY to route packets in scenarios where we need to determine routes based on specific parameters such as congestion, trust, etc. A detailed study on NoC's topology, routing, flow control, and router microarchitecture is found in [76], [71], [77]. Since our thesis primarily targets secure NoC communication, we focus more on how vulnerable the NoC is to malicious alterations like HTs and identify the existing solutions to mitigate such impacts.

2.2 Hardware Trojan

Malicious alteration of hardware during design or fabrication has become a significant security risk. Due to the worldwide nature of the semiconductor design and fabrication process, ICs are more susceptible to malicious modifications. The fabless chip design model and the usage of third-party IPs are the two major contexts in which any rogue element can alter an IC's functionality.

A harmful and purposefully covert alteration to an electronic device, such as an IC, is known as a hardware Trojan. A Trojan can be introduced by an adversary (or a rogue element) that is intended to disable or destroy a system at some future time, or it might secretly leak information and security keys to the attacker [78]. Fig. 2.5 depicts the degree of trust at various stages of a typical IC life-cycle [79]. Each entity involved in the development and production of an IC has the potential to be an adversary that inserts HTs. Moreover, IP cores from third-party vendors can be used by an attacker to implant a Trojan [80]. A great deal of study has been done on the fabrication attacks that might arise from an untrusted foundry [81],[82],[83],[84]. The ICAS framework [85] also details the severity of fabrication attacks.

Trojans can be implemented as hardware modifications to ASICs, microprocessors, micro-controllers, network processors, or digital-signal processors (DSPs) [80]. The attacker implants the Trojan with the intention of inducing the following degrading outcomes:

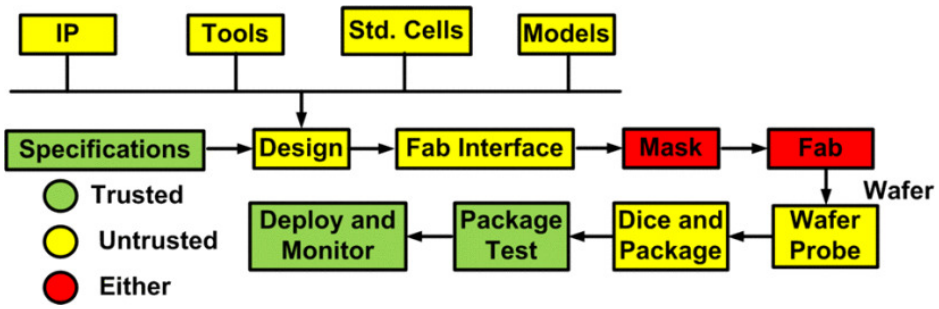


Fig. 2.5 Vulnerable steps of a modern IC life cycle

a. Modification in Functionality: Its impact may result in a change in functionality that was not intended by the specifications for the integrated circuit. For instance, a Trojan installed in the GPS can alter the positional information that the GPS produces.

b. Reduce Reliability: The Trojans were created with the intention of lowering the circuit's performance. It makes the device operate badly following specific actions, increases system error, and results in frequent system failures.

c. Information Leakage: A Trojan may be installed that solely releases private data while leaving the system's performance and functioning unaffected.

c. Denial of Service: By repeatedly demanding services from the IP cores or servers, an HT can prevent other users from accessing the system.

It is extremely difficult to detect malicious changes in systems for a variety of reasons. These are:

a. Because there are so many soft, firm, and hard IP cores used in SoCs, and because today's IP blocks are so complex, it may be very challenging to spot even the smallest malicious change.

c. By design, Trojan circuits often activate under extremely precise conditions, also known as rare circuit conditions. The rare conditions for the Trojan's activation might not be realised during the testing time because of the limited size of the test set. Some Trojans are triggered only by signals from some malicious application (or colluding application).

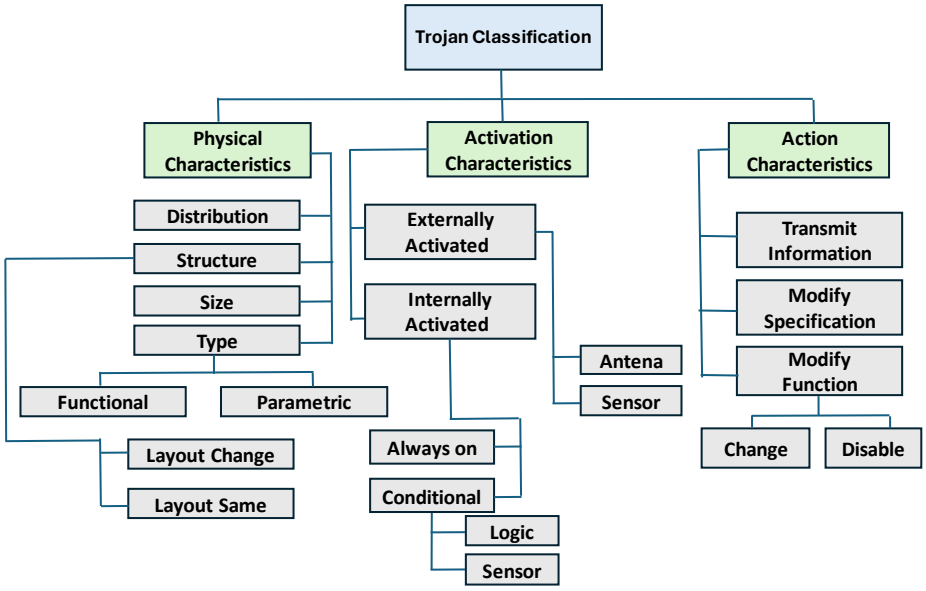


Fig. 2.6 Taxonomy showing physical, activation, and action characteristics of Trojans

2.2.1 Trojan Design and Taxonomy

Since harmful changes to a chip's structure and functionality may take many different forms, Wang et al. divided the Trojan taxonomy into three primary types based on their physical, activation, and action properties. Fig. 2.6 shows this classification [86]. Trojans that are physically realised by adding or removing transistors or gates fall under the functional class, whereas Trojans that are physically realised by modifying existing wiring and logic fall under the parametric class. The size category takes into consideration the number of components that have been added, removed, or compromised from the chip. The Trojan's actual placement on the chip is described by the distribution category. The structure category describes the scenario in which an adversary must recreate the layout in order to introduce a Trojan.

Activation characteristics refer to the requirements that a Trojan must meet to activate and perform its disruptive function. We can divide Trojan activation features into two groups: externally triggered (by an antenna or sensor that can communicate with the outside world) and internally triggered, which further subdivide into always on and condition-based. "Always on" denotes that the Trojan is always active and has the ability to interfere with the chip's operation at any moment. The condition-based subclass includes Trojans that remain dormant until a specific condition is satisfied. The sensor's output, which tracks temperature, voltage, or any other form of external environmental state, may serve as the activation condition. Alternatively, the activation condition could rely on a specific input pattern, an internal counter value, or a state of the internal logic. The disruptive behaviour

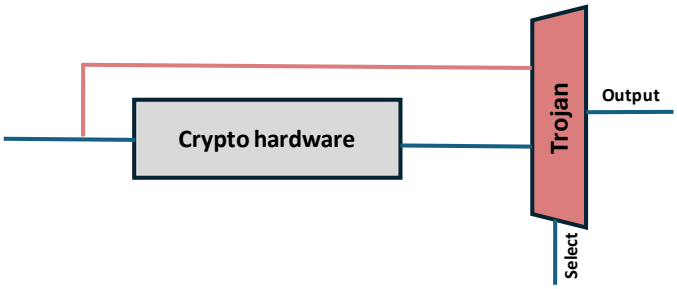


Fig. 2.7 A simple Trojan operation

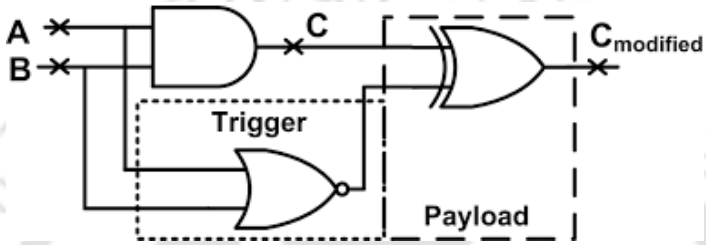


Fig. 2.8 A combinationally triggered Trojan

introduced by the Trojan is recognised by its actions. Three types of Trojan behaviour are classified as follows: information transmission, modification of function, and modification of specification.

Hardware Trojan - An example

HTs are primarily composed of trigger (activation mechanisms) and payload circuits (the part of the circuit activated after trigger excitation). While the trigger section starts the HTs' activity, the payload disrupts the circuit. A simple Trojan operation is depicted in Fig. 2.7. When the chip behaves normally, the output receives encrypted data from the crypto-hardware. When the Trojan is active, it sends the plaintext to the output, bypassing the encryption module. Even a single multiplexer has the ability to act as a Trojan when its actual design is altered [87]. A combinationally triggered Trojan [79] is illustrated in Fig. 2.8. A payload node C receives an inaccurate value at $C_{modified}$ due to the condition A=0 and B=0 at the trigger nodes A and B. In order to make it highly unlikely that the Trojan would activate during a routine manufacturing test, an adversary would often pick an extremely unusual activation condition.

Synchronous counter ("time-bomb") Trojans, asynchronous counter Trojans, hybrid counter Trojans, and Analog Trojans (triggered based on logic value or circuit activity) are

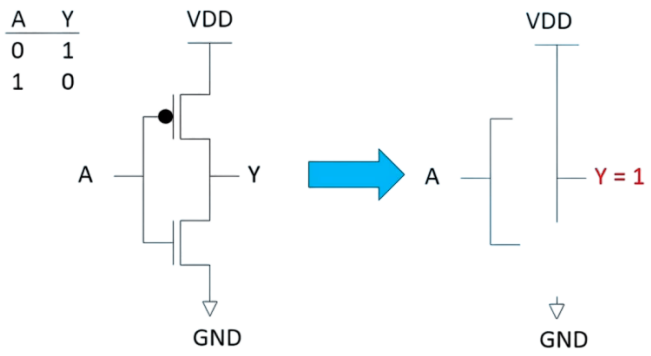
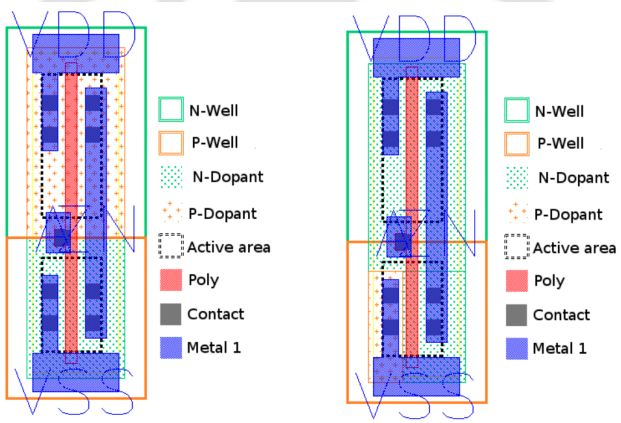


Fig. 2.9 Inverter Trojan always outputs one



(a) Original (b) Trojan

Fig. 2.10 Unmodified inverter gate (a) and a Trojan inverter gate with a constant output of 1 (VDD) in (b)

examples of other forms of trojans. All these Trojans have in common is that they are inserted at the HDL level. Figure 2.9 shows an inverter Trojan that always outputs 1. Unlike the ones listed above, this inverter Trojan modifies the gate's initial design by modifying the dopant polarity in specific regions of its active area. This is referred to as dopant-based Trojans [81]. Fig. 2.10 depicts a trojan of that kind in which the dopant polarities of a CMOS inverter gate are modified to form an inverter Trojan. Several of the standard Trojan testing techniques, like optical inspections, cannot pick up on the modifications at the dopant level. Dopant-based Trojans can compromise the security of Intel's True Random Number Generators (TRNG) used in Ivy Bridge processors.

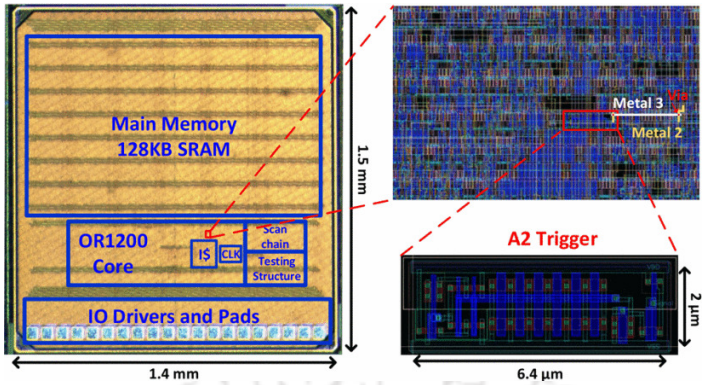


Fig. 2.11 Malicious OR1200 chip with a zoom-in layout of inserted A2 trigger

2.2.2 Potential HT attacks

In September 2007, Syria's sophisticated air defence system was inactive throughout the operation when Israel launched an assault on a nuclear reactor. Adee [88] predicted in 2008 that Syria's air defence system had been turned off via a built-in kill switch that could be accessed and triggered remotely. A backdoor that was installed in commercially available microprocessors in the Syrian air defence system was thought to have been activated during the strike. In January 2014, the New York Times reported that nearly 100,000 computers have been infected with malicious radio frequency hardware [89] by the National Security Agency (NSA) of the US, which is targeting not just Russian military networks but also trade organisations in the European Union. In 2016, Yang et al. [82] introduced a small malicious HT named A2 in an open-source processor, the OR1200. A2 permits unprivileged software to have complete control over the CPU so as to implement a privilege escalation attack in OR1200. A2 is introduced during the fabrication time of the processor and is no bigger than a single gate. Fig. 2.11 illustrates the portion of the A2 trigger in the chip area. Researchers proved that A2 could evade HT detection during testing. In October 2018, Bloomberg published an article titled "The Big Hack: How China Used a Tiny Chip to Infiltrate U.S. Companies" that shocked the whole globe [90]. The article highlights the presence of malicious microchips embedded in Supermicro motherboards used by tech giants such as Apple and Amazon. Bloomberg asserts that Chinese companies manufacture these microchips to intercept confidential information and technical data.

2.2.3 HT Detection

The detection methods [91] attempt to identify Trojans in the pre-silicon or post-silicon stages. Before incorporating third-party intellectual property (3PIP) cores into a design, pre-

silicon detection techniques are used to validate them. Either destructive or non-destructive testing is required for post-silicon Trojan detection. Destructive testing implies reverse engineering. Although this approach is costly and time-consuming, it guarantees Trojan detection in a single IC. Non-destructive methods use functional testing (similar to pre-silicon Trojan detection) and side-channel analysis. Using logic testing (pre-silicon), specialised test patterns are created and used to find HTs. Trojan detection via logic testing may be impractical for complex systems because the number of potential input patterns is exponential. Both the golden design and the test design use side channel analysis to evaluate transient current, power consumption, or path latency for Trojan detection. We consider the presence of a Trojan when there is a threshold difference between the measured signals from these two designs. The logic testing approach is effective for small Trojans. The test generation process is complex with logic testing, and large-scale Trojan detection is also challenging. For large Trojans, the side-channel approach is effective, and test generation is also easy. Trojan detection techniques based on reinforcement learning (RL), such as TGRL [92], DETERRENT [93], and ATTRITION [94], also exist.

2.3 HT Attacks in SoC and NoC

Modern SoC designs incorporate 3PIP blocks, obtained from a broad pool of trustworthy and untrusted suppliers, to address the unprecedented speed of market pressure and combat the escalating costs of design and verification. The widespread adoption of 3PIP components has introduced a region of suspicious hardware within the SoC, resulting in a catastrophic security flaw. Intentionally installing HT in crucial 3PIPs can have severe financial effects on both IP clients and the end user.

2.3.1 SoC Threat Model

Fig. 2.12 depicts a typical scenario where a SoC integrator purchases multiple components from a wide pool of 3PIP suppliers [95]. In this case, SoC Integrator-4 received a 3PIP NoC that included an HT. There may be a malicious circuit in one or more of the untrusted 3PIP blocks. The compromised NoC, with the help of accomplice software, can perform a variety of harmful actions at runtime that harm the SoC integrator's reputation. Only components designed in-house undergo thorough security and functional testing by the SoC integrator. Here is a summary of the threat life cycle that results from insertion of a malicious circuit during the IP block's design.

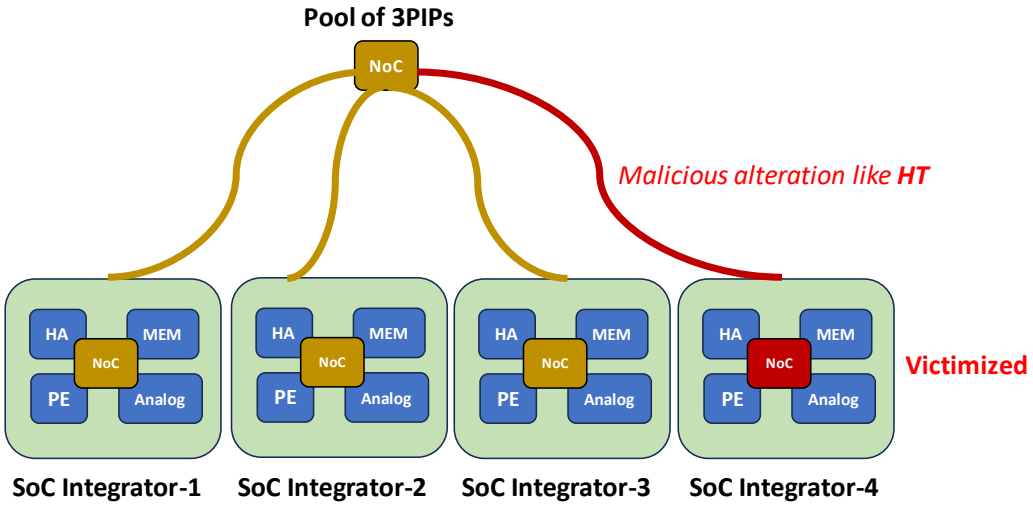


Fig. 2.12 One of its SoC integrators receives a third-party NoC from a source that is sporadically equipped with an HT.

- Trojan insertion - One or a small group of key IP design engineers may implant malicious circuitry during the IP RTL design process. An IP provider may offer Trojan embedded IP to its clients as a result of the actions of a few workers.
- Trojan activation - Trojans are intended to stay inactive in order to avoid detection during IP verification and SoC integration testing. Internal or external triggers can initiate Trojans at run-time through sequential or combinational rare event triggers.
- Trojan operation - Once activated, the Trojan payload during run-time is capable of a wide range of assaults, including data corruption, function modification, information leakage, and denial of service.

2.3.2 Unique Challenges in Securing NoC-based SoCs

Unlike computer network domains, the adoption of security measures in NoC-based SoCs introduces overhead for area, power, and performance. Although the realm of computer networks may support sophisticated security countermeasures, the resource-constrained nature of embedded and Internet of Things (IoT) devices presents extra difficulties [16], as shown below.

Conflicting Requirements

NoCs must meet a wide range of criteria in order to enable communication between IPs, including real-time constraints, security, privacy, and energy efficiency. It is challenging to meet competing demands like security and energy efficiency. For instance, with IoT devices with limited resources, it would not be able to perform conventional security methods like text encryption using the AES cypher and employing SHA hash algorithms. Similarly, it's possible that security needs and domain-specific requirements conflict.

Increased Complexity

Since SoC designs are so complicated, it is hard to do thorough security validation. For competitive reasons in the market, suppliers typically provide IPs as black boxes. Because of this, verification engineers cannot see the whole design. Modern verification techniques often aim to uncover missing or incorrect functionality, potentially concealing security flaws.

Diverse Technologies

Although creating NoC-based SoCs typically involves electrical communication, new NoCs can also enable chip-scale photonics (optical NoC) and wireless communication (wireless NoC). As a result, security solutions for NoCs must not only handle security over electrical cables but also take into account the new problems posed by data transfers across photonic waveguides and wireless channels.

2.3.3 Security attacks in NoC

Security flaws in NoC-based SoCs come in a number of forms. Fig. 2.13 shows the five classes of security attacks in NoC [16]. Below is the summary of five different categories of security attacks.

1. Eavesdropping: Listening to secure NoC packets carrying private information sent over the NoC while other IPs are conversing.

2. Spoofing/Data integrity: Data that has been intentionally corrupted to lead to anomalous behaviour and/or system failure, whereas spoofing is a form of impersonation attack that passes off communication from an untrusted IP as coming from a trustworthy IP.

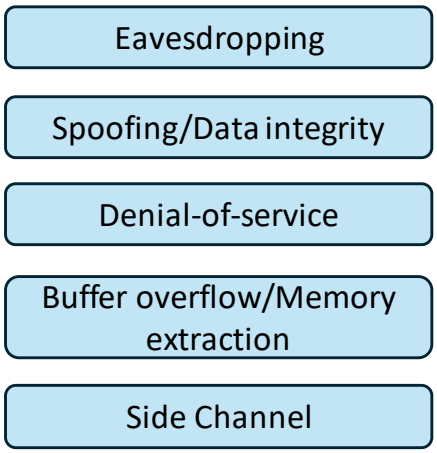


Fig. 2.13 Five classes of security attacks in NoC.

3. Denial-of-Service: Attacks that seek to overburden network resources in order to degrade performance, and violate real-time assurance.

4. Buffer overflow/Memory extraction: Unauthorised privilege levels and memory locations can be accessed through this attack.

5. Side Channel: Use non-functional elements like time, power, and electromagnetic radiation to launch attacks.

There is an immediate need for reliable and secure communication since the Network-on-Chip (NoC) serves as a hub for sensitive data transfer and crucial device coordination. Our study primarily focuses on eavesdropping and data integrity attacks.

A. Eavesdropping attack

The term eavesdropping attack, often referred to as snooping or sniffing, describes when an attacker listens covertly to on-chip communication in an effort to get sensitive data. The attacker aims to gradually leak information without drawing attention to themselves. For instance, an adversary could potentially jeopardize national security by implanting an HT in military systems, thereby exposing sensitive military data. One commonly explored threat model to launch eavesdropping attacks in NoC is the Malicious Router and Colluding Application (MRCA) [16], [96]. In MRCA, a malicious NoC IP works together with an associated malicious programme running on another IP to conduct an eavesdropping attack. An example of MRCA is shown in Fig. 2.14. Through node X, communication between two trustworthy apps (A and B) operating in nodes S and D is possible. The router at X becomes

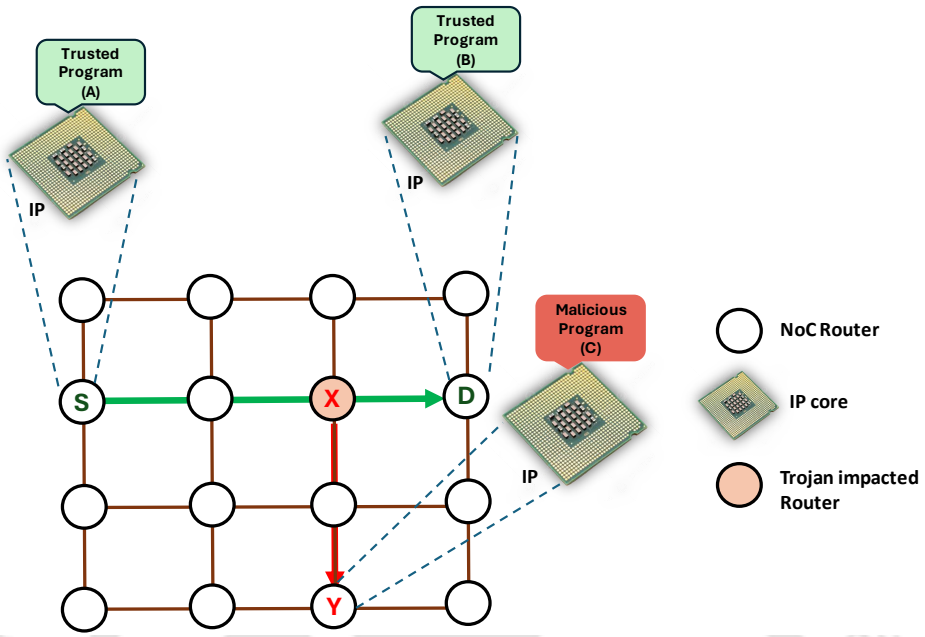


Fig. 2.14 An eavesdropping attack: A hostile router (X) duplicates packets flowing through it and sends them to a malicious application running on (Y).

compromised as a result of an HT that was introduced into the NoC IP during design time. The compromised router duplicates whatever packets it receives and transmits them to node Y, where the malicious programme (C) is running. As a result, S and D's conversation is overheard by the malicious programme at Y [50]. A block diagram of a router infected with the HT that can aid this attack is depicted in Fig. 2.15. The malicious application subtly sends commands to the Trojan to initiate packet duplication and transmission. When the Trojan is active, it gains direct access to the contents of buffers (virtual channels) at the input ports. In the NoC framework, researchers consider Router to be the prime location for HT implantation [47], [35], [97]. Researchers have shown that insertion of a Trojan circuit into a router logic causes its area to hike only by 0.2%, which is too tiny to be detected [97].

Defences against eavesdropping

Identifying the location of the security service implementation in the NoC framework is a crucial decision when designing a security solution for NoC communications. Most of the researchers have considered NI a trusted entity where security solutions can be implemented, as it is assumed to be an in-house design element [47],[98],[97].

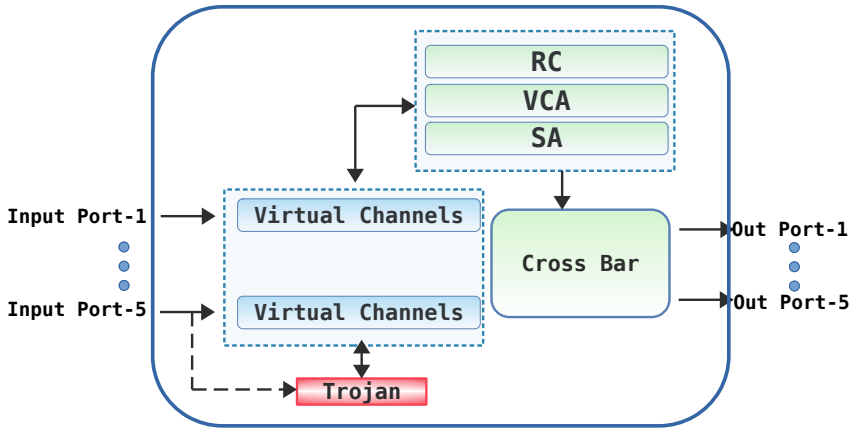


Fig. 2.15 A router infected with an HT that can facilitate eavesdropping attack. Router pipeline stages such as route computation (RC), virtual channel allocation (VCA) and switch allocation (SA) are shown.

Catherine et al.[99] did the first study on the need for safe NoC communication. They describe a symmetric key cryptographic design that uses the Advanced Encryption Standard (AES) to make key transfers safe. For this purpose, they propose a special key-keeper core. Key management with AES creates overhead in the resource-constrained NoC context. Researchers implement an authenticated encryption (AE) block inside NI to prevent unauthorised access to critical data from secure cores [46]. The authors employ the Galois/Counter Mode (GCM) method to achieve AE, which also incorporates AES. Even though the system is secure, it appears to incur a lot of overhead in modern multi-core NoC frameworks. A three-layer security system involving data scrambling (DS), packet certification, and node obfuscation (NO) is made available in a system called Fort-NoC [50]. Its main objective is to defend against information leak attacks. The DS layer of Fort-NoC employs the XOR cypher for data encryption. The NO layer experiences significant power overhead. The P-Sec [55] uses algebraic manipulation and detection codes (AMD) at the NI level to protect critical data in a compromised NoC. Performance metrics show that P-Sec has less overhead than Fort-NoCs. P-Sec aims to protect against silent data corruption attacks, but at the cost of significant latency and power consumption.

There is an idea for IP cores to communicate through a tunnel that uses the counter mode of AES and SipHash-based MAC authentication [54]. This approach makes no mention of key exchange across IP cores prior to establishing encrypted packet transfer. In addition, their strategy does not appear to include a recovery mechanism after HT-induced packet modification. It is not often possible to enable secure packet transmission in NoC using the bit shuffling-based HT defence [100]. The SIM+THANOS approach [96], in contrast to

FortNoCs and P-Sec, pinpoints the source of the spoofing attacks. Their attack methodology uses HT with NI, not NoC routers. A HT attack in NI is extremely unlikely to occur as NIs are developed in-house. A trust-based routing [56] approach is also devised to bypass the HT routers in the path of packets. As the HT behaviour is intermittent during the dynamic packet traversal, we need to explore more options to exploit complete and accurate re-routing of packets from the expected position of HTs.

The approach suggested in [101] calls for a central IP named Key Generation Centre (KGC) to generate the private keys. The KGC has a single physical link to each NI through which it exchanges the private keys. The key can then be used by any symmetric block cypher. When building a multi-core system with a larger number of PEs, adding separate links to each NI from KGC incurs additional overhead in terms of area and power. At the design stage itself, a few studies deal with HT detection in NoC components. Methods like physical inspection [102], functional testing [103], and side channel analysis fall under this category. However, design-time HT testing is still in its early stages. It becomes challenging when NoC components are becoming more sophisticated and HT exhibits sporadic behaviour.

The approaches outlined above, except for [101], do not state a suitable key-exchange mechanism in order to incorporate an effective symmetric cypher at the NI level. A central strategy like KGC in [101] can be more vulnerable to external or EM/power attacks too. A key-less encryption scheme combining chaffing and winnowing (C&W) and all-or-nothing transform (AONT) is also developed for NoC communication [47]. It is a cost-effective mechanism when compared with the AES-128 parallel CTR. The authors of [104] show the need for a lightweight block cipher, the Hummingbird-2, to encrypt the NoC packets in order to minimise the costs related to AES encryption. The NoC packet structure while using security services is rarely described in most techniques. A list of the articles we investigated that examined eavesdropping attacks may be found in Table 2.1. Both overhead (compared to the overhead of the default architecture without security) and effectiveness (security guarantees against eavesdropping attacks) are marked on a three-step scale (low, average, and high).

The initial effort to use machine learning (ML) techniques for a NoC-based multi-core system to identify HTs in real-time is explored in [106]. The paper details an ML-based eavesdropping sensing algorithm (ESA). The model analyses the NoC traffic data, and a trained ML model is stored in a dedicated IP core called a decision unit. Though it provides good accuracy in detection, its practical applicability in real systems is still unexplored. An RL-based adaptive router microarchitecture is also well explored [107].

To prevent confidentiality breaches and to defeat the action of an eavesdropper, researchers now focus on the **anonymous** transfer of NoC packets. Researchers have suggested

Table 2.1 Works detailing defences against eavesdropping attack in NoC.

	Defence	Overhead	Effectiveness
[99], 2003	Authentication/Encryption	Average	Average
[46], 2011	Authentication/Encryption	High	High
[50], 2014	Authentication/Encryption	Low	Low
[55], 2016	Authentication/Encryption	Average	Average
[54], 2017	Authentication/Encryption	High	High
[100], 2018	Information Obfuscation	Low	Low
[105], 2018	Authentication/Encryption	Average	Average
[56], 2020	Authentication/Encryption	Average	High
[47], 2021	Authentication/Encryption	Average	High
[49], 2022	Authentication/Encryption	High	High

anonymous communication (AC) systems as a practical preventive measure to reduce the risks of communications monitoring. David Chaum introduced the concept of anonymous communication for the first time in 1981, launching a brand-new field of privacy research. It is now relevant to on-chip communications as well, owing to serious hardware risks. The idea of anonymity shields the identities of the parties involved in the conversation. The sender anonymity protocol protects the sender's identity. Several recommendations for anonymity network architectures for the Internet, including Tor, Tarzan, BitTorrent, NetCamo, I2P, Crowds, Freenet, and Mixmaster, are documented in the literature. In the NoC framework, researchers deal with two approaches to handling anonymity in on-chip communications. One approach, called ARNoC [51], follows a two-phase mechanism in which the first phase handles route discovery, involving packet flooding and layered encryption and decryption steps, while the second phase triggers data transmission. This is a very costly mechanism on a resource-constrained platform, as well as performance degradation. The second approach involves embedding extra routing information into the packet, which can determine a path towards the destination by adhering to a dedicated routing policy [52]. We will refer to this method as ARPP (anonymous routing with a pre-computed path). The authors detail the behaviour of systems with secure and non-secure packets. They apply anonymity to secure packets that contain sensitive information. Every intermediate location must update the extra routing data in accordance with their routing policy. The encrypted destination address renders a traditional routing mechanism ineffective in this scenario. This method exchanges session keys and then transmits data. Before making a routing decision, it is necessary to compare the encrypted destination address at each intermediate router. This study fails to

explain how symmetric encryption maintains anonymity during session key exchange. Below, we detail the specific weaknesses of the two methods.

ARNoC: (a) Because of the intricate cryptographic computations, there is noticeable overhead during the route discovery phase. (b) Traffic analysis enables the conclusion of source-destination pairs, even if the nodes' identities remain secret during communication.

ARPP: (a) Traffic analysis can be possible with malicious implants as it uses traditional, non-adaptive routing methods. (b) It involves computations at intermediate routers during packet routing, which degrades NoC performance.

C. Data integrity attacks

One commonly explored threat model to launch integrity attacks in NoC is to implant a trojan at routers so that it can corrupt packets passing through them. The threat mode is generally called *Packet Corruption at Routers (PCR)* [16]. Fig. 2.16 illustrates the PCR model in a 4x4 NoC. We assume a Trojan is implanted at router 5 during the design phase. We depict the wormhole flow of a five-flit packet P_1, which consists of one head flit (H), three body flits (B), and a tail flit (T), from router 4 (S) to router 15 (D). The HT at 5 modifies the third body flit of P_1. Packet corruption of this kind degrades SoC performance, and even systems eventually fail to run. MalNoC [54] is a Trojan-infected NoC that is capable of carrying out multiple attacks on NoC packets, including data integrity attacks. A HT-infected router in MalNoC duplicates packets arriving at a router, changes the packet contents with the content in a malicious register, updates the destination address in the header to the desired IP, and injects it back into the NoC. A Trojan that tampers with flits entering a router's input buffers was explored by Kumar et al. [108]. However, they were primarily concerned with launching DoS attacks.

Defences against data integrity attacks

Authenticated encryption systems offer data integrity using authentication and data secrecy through encryption. If the authentication tag is calculated using the entire packet (header as well as payload), any packet corruption can be detected at the receiver's side when the packet is validated using authentication. In order to preserve data integrity and authenticity in NoC communication, we can mainly adopt three measures, such as error detection and correction mechanisms, the usage of hash functions or message authentication code (MAC), and applying physically unclonable functions (PUF) for authentication. The authors of [55] explain the usage of algebraic manipulation detection (AMD) and cyclic redundancy check (CRC) in ensuring integrity. The authors consider that HTs possess the ability to alter a

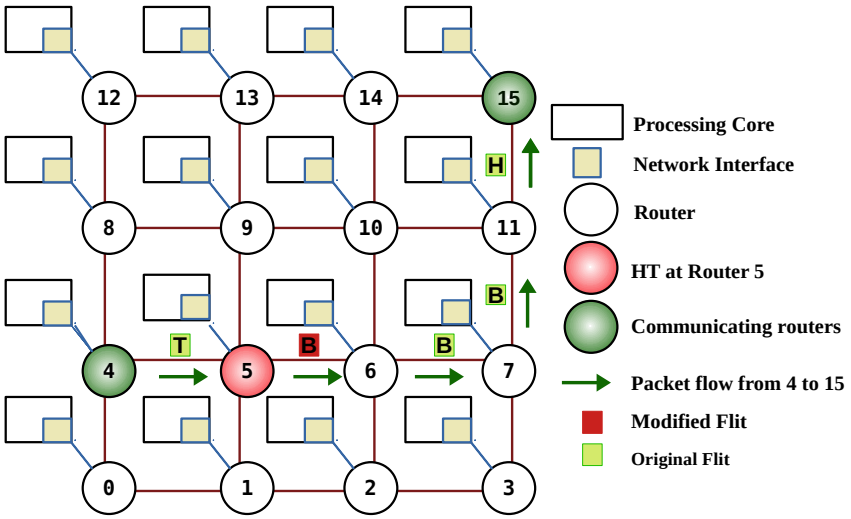


Fig. 2.16 A Trojan at router 5 corrupts one of the body flits passing through it.

packet without affecting its CRC. AMD mitigates this vulnerability by incorporating route information into the packet header. They use AMD code for sensitive packets and are given higher priority than regular packets. This approach yields significant area overhead. In order to ensure source integrity, the Fort-NoCs [50] design suggests a hash-based packet certification. The SoC firmware appends a tag to the data and injects it into the NoC, which is then verified at the destination. The overheads associated with the packet certification are negligible. The SipHash, a keyed hash function, is integrated in [54], ensuring both integrity and authenticity. To provide quick MAC computation for packets, SipHash repeatedly carries out a sequence of add, rotation, and XOR operations. In [109], the mCrypton in cipher block chaining (CBC) mode is used to ensure authentication (MAC). It is stated that this MAC calculation needs 39 cycles, and compared to the area of a state-of-the-art MPSoC, the area overhead is found to be 2.7%.

According to Hussain et al. [105], since the Trojan is seldom active to prevent discovery, authenticating every packet may result in a decrease in energy efficiency. In their study, they proposed a concept for an effective Trojan detection system that enables authentication only after the system triggers the HT. In NoCs, error-correcting codes (ECCs) have been utilised to fix bit errors brought on by particle strikes, crosstalk, and spurious voltage fluctuations. In [110], Y et al. presented a technique to identify Trojan-induced errors using ECCs. **Trust-based routing** [56] is proposed for packets to bypass Trojan routers. Here, the Trojan's payload is defined to alter the packet content. If the packet changes, the destination's MAC verification initiates a re-transmission. It's a dynamic routing approach in which each router considers the trust with its immediate neighbours and with routers that are two hops away.

At the routers, a sigmoid function calculates trust. The method comprises a broadcast of trust packets (delegated trust packets) to neighbours. It does not account for the Trojan altering trust packets at routers. Trust delegation also generates extra NoC traffic. Compared to the baseline router, the technique has an area overhead of 6%. Researchers also explore watermarking-based NoC authentication [111] and physical unclonable function (PUF)-based NoC authentication [112]. There are works that deal with the Trojan that modifies the control fields of the packet, like flit type [113], [114], and destination address [115]. A dynamic PUF-based flit permutation and integrity check is devised in [113] to mitigate the HT attack. Run-time caging and packet re-routing are followed in [114] to thwart the impact of HT. In order to counter the attacks, additional validation, error correction, or re-transmission is utilised. The additional hardware needed to execute the defences adds power, performance, and area overhead. Table 2.2 shows a list of articles detailing the integrity attacks in NoC.

Table 2.2 Works detailing defences against integrity attack in NoC.

	Defence	**Overhead**	**Effectiveness**
[46], 2011	Authentication/Encryption	High	High
[48], 2013	Authentication/Encryption	High	High
[110], 2013	Error correcting codes	Low	Average
[54], 2017	Authentication/Encryption	Average	High
[105], 2018	Authentication/Encryption	Average	High

C. Other threats

We divide DoS attacks [116] on NoC-based SoCs into three groups: flooding, packet corruption, and traffic flow manipulation. An illustrative example demonstrating the flooding-type DoS attack in a 3x3 NoC is shown in Fig. 2.17. Continuous packet corruption can potentially result in a DoS attack. To create a DoS attack, the packets are even handled unfairly at the router [117]. Denying fair access to the switch allocator manipulates the traffic flow.

The network interface (NI) of the NoC implements access authorization to protect against buffer overflow attacks. It is also proposed [42] to implement dynamic firewalls to monitor and filter the NoC traffic. In order to add an extra layer of security to access authorization, the whole NoC IP core is divided into secure and non-secure zones [42]. The cores in secure zones provide the security services. When considering side-channel attacks on NoC-based SoCs, two important categories of attacks come into play. There are timing and power/thermal side channels. Since this thesis does not address those attacks, we do not go deep into them.

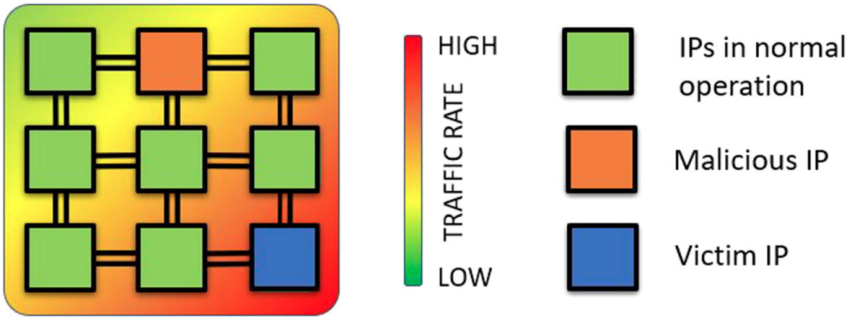


Fig. 2.17 Mesh NoC setup showing DoS attack from an attacker IP to a victim IP. High traffic is visible near the victim IP on the thermal map.

2.4 Evaluation Methodology

This section details the tools and system parameters used for the performance evaluation of various architectures proposed in the thesis work. The workload characteristics and benchmark compositions that are used to analyse the performance of the system are also discussed. We also study the area and power overheads.

2.4.1 Simulation Infrastructure

All the architectures proposed in this thesis are modelled on an event-driven gem5 simulator [118], and the system configuration is listed in Table 2.3. The configuration is similar to the Intel Xeon Phi 7235/7215 processor (64-core type) [119]. In line with the Intel Xeon Phi CPU, Intel Skylake, and Intel Cascade Lake server CPUs [120], our interconnection topology is also set to be 2D mesh. The Garnet [121] module of gem5 implements the interconnect part of the simulated multi-core system. Both the threat models and the countermeasures are designed with garnet module. The life of a message as defined by the Garnet module [122] is shown in Fig. 2.18. The cache controller generates control or data messages, which the network interface converts into flits before injecting them into the NoC framework. The router's underlying routing logic guides these flits as they traverse the network. As soon as it reaches the destination network interface through the destination router, the flits get converted back to messages and transferred to the cache controller. In order to generate NoC traffic and to analyse how vulnerable it is to HTs, we use synthetic traffic generators as supported by the Garnet and two real benchmarks, such as SPEC 2017 (multi-programmed) [123][29] and SPLASH 2 (multi-threaded) [123].

Table 2.3 System Parameters

Parameter	Specification
Processor	64 and 16, in-order x86 cores, 1.3 GHz
L1 Cache	64KB(I), 32KB(D), 2-way, 64B block, private
L2 Cache	32MB, 8-way, 64B blocks, shared
Cache Coherence	MESI-Two Level
NoC	8×8 and 4×4 2D mesh, 128-bit flit
VNET and VCs	3 VNET, 4 VCs/VNET
Routing (Default)	XY or DyXY
Packets	1-flit control packets, 5-flit data packets
Synthetic Traffic	Uniform-Random
Real Traffic	Workload using SPEC CPU 2017 and SPLASH-2 benchmarks
NoC Router pipeline	3 cycles for router + 1 cycle for link

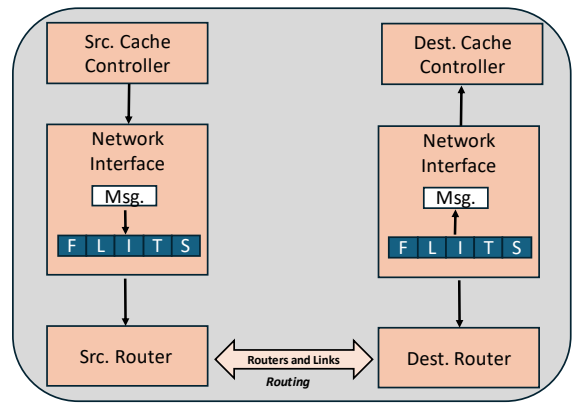


Fig. 2.18 Flow of a message from source cache controller to destination cache controller as defined by Garnet module in gem5 simulator.

To run a multi-core system with various applications, we basically do two types of simulations, one on a 16-core machine and another on a 64-core machine. For 16-core, we use multi-threaded SPLASH-2 applications, and for 64-core, we use a multi-programmed workload consisting of 64 independent SPEC applications.

The SPLASH-2 benchmark binaries, such as *Barnes*, *Fmm*, *Ocean*, *Radiosity*, and *Raytrace*, are taken for performance analysis. Fig. 2.20 shows the injected traffic patterns experienced with SPLASH-2 benchmark applications on a 4x4 mesh NoC. It is seen that *Ocean* experiences high NoC traffic and *Raytrace* has the least NoC traffic. These SPLASH-2 binaries can produce non-uniform, diverse NoC traffic. Additionally, four binaries, such as *Fft*, *Radix*, *Lu*, and *Fmm*, are randomly selected for an HT's impact analysis in Chapter 4.

We also create five workloads, taking into account the L1 cache miss rate of various SPEC 2017 benchmarks. Based on the MPKI (Misses Per Kilo Instruction), SPEC benchmarks are categorised into two groups: high MPKI (greater than 10) and low MPKI (less than 6). Our requirement is to create NoC traffic of varying intensity, so we arbitrarily selected two limits (10 and 6) so that the benchmarks in each group experience sufficient cache misses. For security analysis, we wanted to have a network with both high and low packet rates. From experiments, we find that benchmarks *lbm, fotonik3d, gcc,* and *wrf* belong to the high MPKI group, while benchmarks like *deepsjeng, blender, namd,* and *leela* belong to the low MPKI group. Table 2.4 shows the workload composition. For example, WL3 consists of 32 high MPKI (H) and 32 low MPKI (L) application instances. This 32H, 32L composition consists of 16 instances of *lbm*, *fotonik3d*, *deepsjeng*, and *blender* each. We have 64 independent application instances to run on a 64-core multi-core system. Similarly, the composition and behaviour of other workloads can also be defined accordingly. From the composition of the workloads, we can see that WL1 creates the maximum cache misses and thereby the highest NoC traffic, and WL2 creates the minimum cache misses with the lowest NoC traffic. The intensity of NoC traffic load in the other three workloads stays in between these two, and their range depends on the fraction of high MPKI benchmarks in the workloads. Therefore, we can observe the systems' responses under varying network loads.

Table 2.4 SPEC 2017 Benchmarks with Workload Details

Workload	**Composition**	**Benchmarks (No. of instances)**
WL1	64H	*lbm (16), fotonik3d (16), gcc (16), wrf (16)*
WL2	64L	*deepsjeng (16), blender (16), namd (16), leela (16)*
WL3	32H, 32L	*lbm (16),fotonik3d (16), deepsjeng (16),blender (16)*
WL4	48H, 16L	*lbm (16), fotonik3d (16), gcc (16), deepsjeng (16)*
WL5	16H, 48L	*lbm (16), deepsjeng (16), blender (16), namd (16)*

For a 1-million instructions, gem5 is configured to simulate a 64-core (8x8 2D mesh NoC) machine as per the parameters in Table 2.3. The result in terms of injected packet count is shown in Fig. 2.19. The X axis shows the five workloads, and the Y axis depicts the injected packet count. The behaviour is in accordance with the miss characteristics of workloads as specified in Table 2.4. The decreasing order of packet count (or NoC traffic) in workloads is as follows: WL1 > WL4 > WL3 > WL5 > WL2. For experiments, we make use of both 4x4 and 8x8 NoC. Throughout the thesis, we run the SPEC 2017 workloads on a

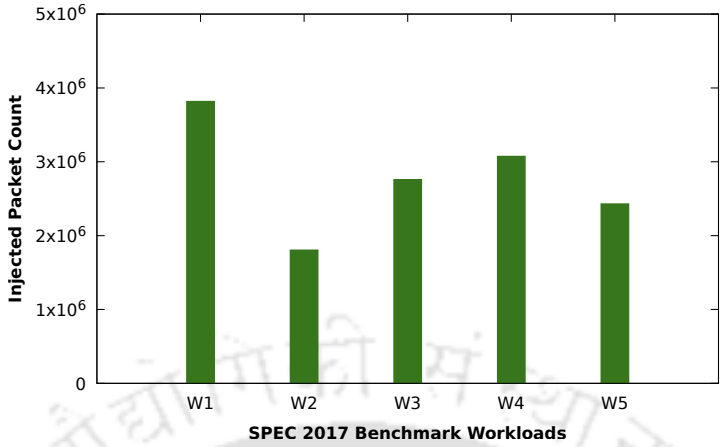


Fig. 2.19 Injected packet count on an 8x8 2D mesh NoC with 1 million instructions under SPEC 2017 workloads.

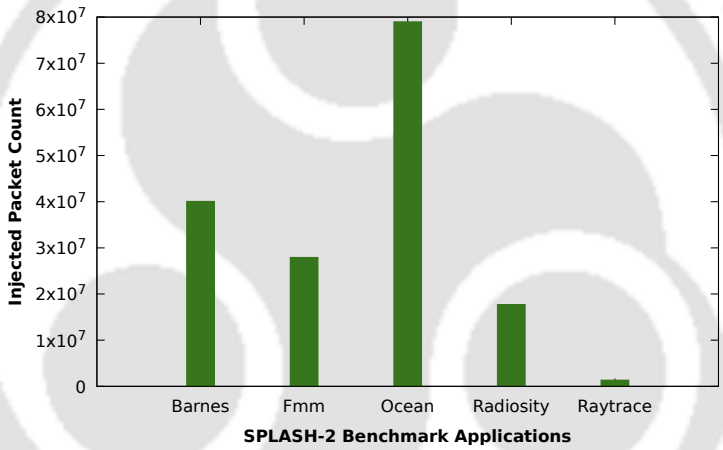


Fig. 2.20 Injected packet count on a 4x4 2D mesh NoC under Splash-2 binaries.

64-core machine with an 8x8 2D NoC and the SPLASH-2 benchmark on a 16-core machine with a 4x4 2D NoC. Hence, we perform security analyse with both multi-programmed and multi-threaded applications.

In order to assess the proposed solutions to each objective in terms of area and power, we implement the solutions as separate modules at routers or NI as per the requirements using Verilog and synthesise using 90 nm technology in Cadence RTL Compiler. So, hardware overhead is also taken into account. Table 2.5 shows the area and power associated with a 5-stage NoC router (Baseline) and Table 2.6 depicts the combined area of router and network interface. The area of Router+NI is required to be compared when both the router and the attached NI are modified as part of the proposed approaches. As illustrated in [52], and [124],

we also considered the total area and dynamic power of a 16-core MPSoC comprising 16 32-bit MIPS processors. Table 2.7 depicts the results of the same.

Table 2.5 Area and Power of a Baseline NoC router

Metric	NoC Router
Area (μm^2)	713702
Power (μW)	83688

Table 2.6 Combined Area and Power of NoC router and Network Interface

Metric	Router+NI
Area (μm^2)	791929
Power (μW)	84079

Table 2.5, Table 2.6 and 2.7 are considered baselines for comparing area and power overheads with the proposed approaches. Fig. 2.21 illustrates the overall experimental design. In the framework, we have a cycle-accurate gem5 simulator that gives performance statistics and a Cadence RTL compiler that gives the hardware overhead. In the following chapters of this thesis, the solution module refers to the proposed solution for each objective.

Table 2.7 Area and Power of a 16-core MPSoC

Metric	NoC Router
Area (μm^2)	1803245
Dyn.Power (μW)	3254

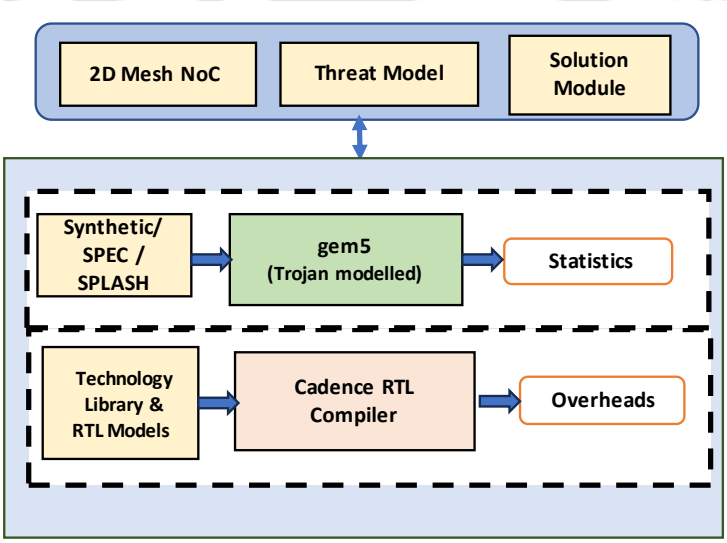


Fig. 2.21 Overall experimental design framework and tools

2.5 Chapter Summary

In this chapter, we describe the background of an NoC architecture and the features of its primary components. We conducted a thorough study on NoC communications vulnerabilities caused by HT and identified major countermeasures to secure NoC packet traversal. Our major requirement is to make the SoC function smoothly without any security breaches at the interconnect level. The chapter ends with the evaluation method, outlining the experiential frameworks and tools used to assess the effectiveness of each solution this thesis proposes.

The next chapter describes our work on implementing lightweight security framework.

Chapter 3

Sec-NoC: A Lightweight Security Framework

This chapter describes our first work, which proposes a lightweight cryptosystem that works against HTs capable of packet leakage. We address the two fundamental security violations, such as confidentiality and integrity, for NoC packets. We also devise a key-exchange method suitable for the NoC framework so that we can easily implement symmetric ciphers. The proposed work incurs lower average packet latency and less area and power consumption when compared to state-of-the-art methods.

3.1 Introduction

The communication patterns inside the system can be discovered by embedding an HT that takes advantage of the underlying interconnection network [125]. A hostile external attacker may obtain access to this data and exploit it to compromise the user profile by learning critical details about the applications that are currently operating on the machine. During packet transfer through NoC, the HT can leak some packets and send them to an external attacker through a colluding application running on a different core. Therefore, it is crucial to prevent information leakage during NoC packet transmission using HTs at routers. The packet leakage primarily poses two security violations [126]: confidentiality and integrity. The violation of confidentiality is straightforward, as an illegitimate third party gains access to the leaked information with the support of covert communication between an HT and a colluding application. Cryptographic techniques can ensure confidentiality in hardware. The choice of a cryptographic algorithm is critical in packet communication at the interconnect level. A resource-constrained environment like NoC requires the algorithm to be lightweight

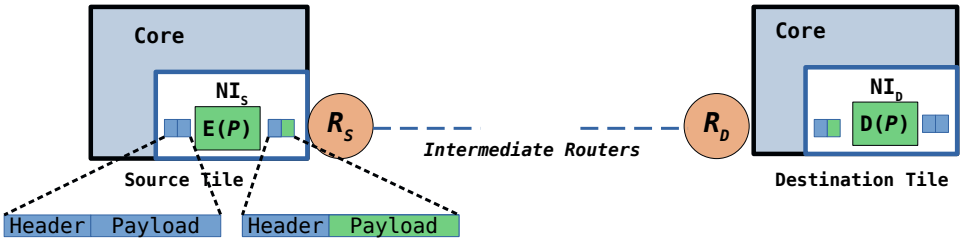


Fig. 3.1 Communication of two core tiles by means of encrypted packet: $E(P)$- encryption of packet P's payload and $D(P)$- decryption of P's payload.

and consume minimal area and power. The encryption procedure at the NoC level in a TCMP is shown in Fig. 3.1. The NI level implements both encryption and decryption algorithms. Due to our focus on confidentiality, we only encrypt the payload part of the packet, leaving the header open throughout the packet transfer to facilitate routing at intermediate routers. After the encryption process, we divide the packet into flits, which we then inject into the NoC. The flits are first received by the source router (R_S) attached to the same NI. The packet routing begins at the source router, and it eventually reaches the destination router (R_D) by passing some intermediate routers, if any.

To reroute the leaked packet to a different destination where the colluding application runs, there must be some modification in the leaked packet's header part, specifically the destination field. So, in order to leak packets, the HT must also have the capability to alter the packet's content.

3.2 Threat Model

We model an HT that is capable of both packet data leakage and packet modification. A covert communication between an HT-impacted router and a Colluding Application (CA) is established to leak sensitive content from packets. The HT, which is intermittently triggered by the CA, transmits a copy of the packet to the IP core, where the CA is running. Fig. 3.2 depicts an MPSoC with 16 PEs connected by a 4×4 2D mesh NoC framework. It illustrates our threat model, in which router 10 is an HT router and the core attached to router 3 runs the CA. The HT at router 10, which is in the XY routing path between 4 and 14, make copies of packets passing through it and sends them to router 3. The HT at router 10, located in the XY routing path between 4 and 14, replaces the actual destination (router 14) with an alternate destination (router 3), where the CA runs. So, it is obvious that the HT has the capability to alter the packets' content. The HT starts screening the source address field of every head flit passing through router 10. In this scenario, the HT's target (the victim router)

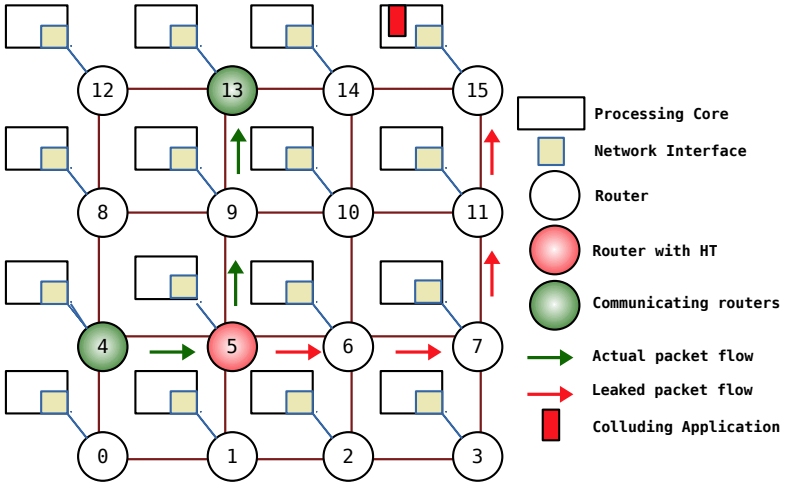


Fig. 3.2 Overview of a 4×4 NoC. Leaked packets are routed by the HT at router 10 to a Colluding application in the core connected to router 3. Here, the HT targets packets from router 4.

is to leak packets that originated from router 4. The HT can change its target according to CA's commands. Fig. 3.3 depicts the packet leakage count from the HT router 10. As soon as a packet is buffered at router 10, the HT checks to see whether the source ID of the packet is the same as that of the victim router ID. If there is a match, packets can be leaked out to a different core where the CA is running. Only the packets passing through the router 10 are leaked out when the HT is active. It is evident that if the target router is 10 itself, all the packets originating from 10 will simply leak out. The packets originating from the routers adjacent to router 10 are being leaked in large quantities if those nearby routers are targeted. We find that when all of the routers are targeted (only one time), an average of 28% of the packets leak in a 4×4 mesh NoC, and these packets eventually reach the colluding application if no security services are integrated. Packet leakage leads to application profiling, which gives attackers the opportunity to launch several attacks that degrade the overall SoC performance. This model is the same as the MRCA [16] model, which was well described in Chapter 2. We presume that the packets are being sent in their raw, unencrypted form. The CA can use the header information and raw payload for malicious activities.

To prevent unauthorised third parties from profiling applications and inducing packet modification, a lightweight authenticated encryption (AE) system is to be proposed for NoC communication to facilitate confidentiality and integrity. Since encryption keys are changed over time, a secure key distribution mechanism is also proposed. The NoC has to be integrated with a mechanism to block the leaked packets from reaching colluding applications

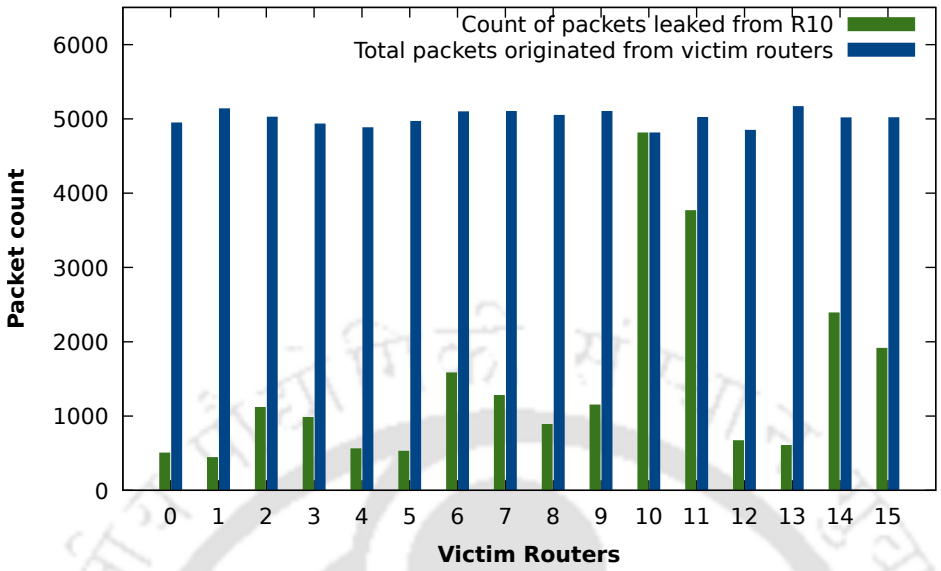


Fig. 3.3 Leakage count of packets from router 10 in a 4×4 mesh NoC. The numbers from 0 to 15 on the X axis show the router IDs. The packets originated from the victim routers are leaked out through router 10 when the packets reach router 10 during routing.

and their unlawful usage. After integrating the cryptosystem, the average packet latency in NoC must be lesser, so there is less impact on the application's performance.

3.3 Motivation

A Remote Access Hardware HT (RAHT) [125], is developed with the capacity to leak information about traffic patterns to an external attacker. This information is sent in packets. The attacker applies ML techniques to these packets to extract information about the applications running on the system and the processor's micro-architectural features. The fundamental concept is that deterministic routing creates a strong correlation between the number of packets passing through a specific router and the applications or architectural features. They used only randomness in the routing pattern as a defence against this RAHT. The randomness in selecting output ports creates non-optimal routing paths. Hence, there is a reduction in the correlation between the packet count through a path and the architectural features. They show that this reduction in correlation also led to a decrease in the attacker's accuracy in information extraction from packets (accuracy drops to less than 20%). Fig. 3.4 shows the RAHT threat model. Two communicating routers are marked as source (S) and destination (D). The XY path from S to D is marked green there. The HT also appears on the same path, analyses the packet flow, and sends the traffic information (in the form of packets generated

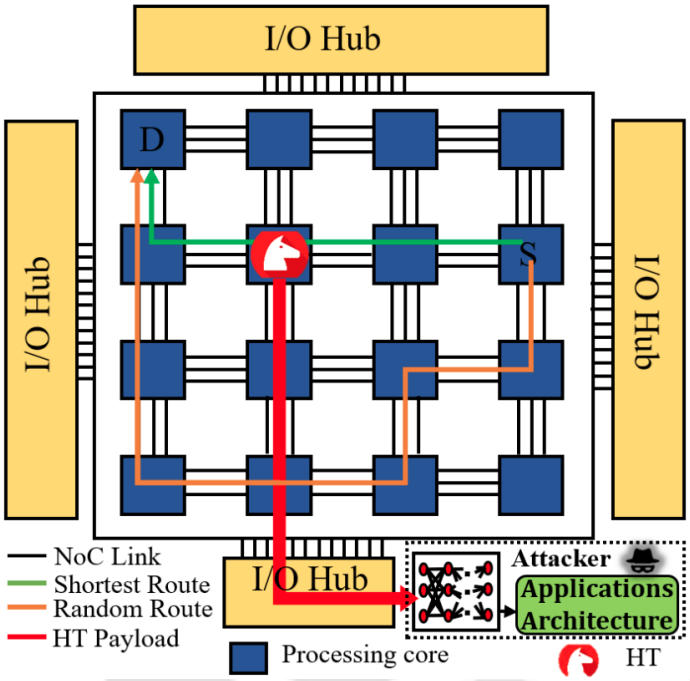


Fig. 3.4 The RAHT threat model showing HT packet leakage: shortest route and random route between source (S) and destination (D) is depicted.

by HT) to external attackers through the I/O hub. There is also a random route between S and D, marked orange, which serves as a countermeasure against this attack.

We can infer from this model that the attacker still retains the ability to extract information using ML techniques, as their approach does not reduce accuracy to zero. In this scenario, the primary problem arises when an HT-generated packet leaves the NoC framework and reaches the attackers for further processing. As soon as a packet (both regular and HT-generated) containing sensitive information reaches an attacker, their method uses an ML-based algorithm to extract useful information. If those packets are encrypted, the attacker can still apply complex cryptanalysis to extract the required data from them. To eliminate packet leakage from the NoC, the only solution is to use cryptographic techniques such as integrity checks and authentication. Through these techniques, we make sure that NI blocks every leaked packet from HT-impacted routers, ensuring that no single packet reaches the attacker.

The literature in Chapter 2 already detailed various approaches to cryptographic techniques in NoC. Unlike the existing literature, our focus is to establish a lightweight cryptographic system that takes into account both the confidentiality and integrity of NoC packets, with an emphasis on eliminating packet leakage. The main challenge is to establish an appropriate key-exchange mechanism while deploying a symmetric cipher at the NI level. The existing works rely on classical key exchange protocols like Diffie-Hellman [52] and

that involve some computational overhead, which is not suitable for the NoC environment. Our primary goal is to design a system that minimizes the impact of cryptographic primitives on average packet latency and prevents external attackers from accessing leaked packets. Now, we detail a general threat model showing packet leakage.

3.4 Secure NoC (Sec-NoC)

We propose a cryptosystem called Sec-NoC at the NI, wherein we integrate lightweight symmetric cryptography to facilitate confidentiality, elliptical curve cryptography (ECC) to facilitate key exchange mechanisms and authenticate the origin of secret keys, and a lightweight MAC producer to ensure integrity and block packet leakage. All of these security services are invoked in the NI.

3.4.1 Elliptic Curve Cryptography (ECC)

ECC is a popular method for lightweight encryption in hardware systems due to its strong security assurance and tiny key size. As it takes less processing power and memory than conventional encryption techniques like RSA, ECC is particularly suited for devices with limited resources (IoT/embedded systems) [127]. ECC can be implemented in hardware using special cryptographic processors, ICs, or field-programmable gate arrays (FPGAs). ECC algorithms are supported in Intel Atom processors like the C3000 and are also supported by a number of well-known ARM Cortex-M micro-controllers, including the STM32 from STMicroelectronics, the LPC from NXP, and the EFM32 from Silicon Labs. Additionally, there are several secure ECC-enabled element chips, such as Microchip ATECC608A and Infineon OPTIGA Trust X. Small key sizes help to decrease the computational, memory, and key generation overhead associated with ECC. In Sec-NoC, we also utilize ECC.

An elliptic curve over a finite field takes the following form:

$$y^2 = x^3 + ax + b(mod\ p) \tag{3.1}$$

where p is a prime number, and a and b are parameters. To reduce the computational and memory costs associated with ECC operations, we employ a 16-bit prime integer in building a secure NoC system. For all IP cores, the key pair generation technique for ECC is used to generate both public and private keys using the pre-fixed generator point (G) and the order (n) of the cyclic group. In ECC, a private key is a scalar number, while a public key is a point on the curve (x,y). The public and private keys of our cryptosystem are 32 bits and 7 bits, respectively. During the SoC integration phase, we store each NI's own private key and the

public keys of every IP core in the mesh NoC. A separate key storage space is created for this purpose. It can be implemented either with static random-access memory (SRAM) or with non-volatile memory (NVM) and is only accessible at the NI.

3.4.2 Lightweight Block Ciphers

Cryptographic algorithms known as lightweight block ciphers are created for situations with constrained resources, including smart cards, wireless sensors, and low-power devices [128]. They feature lower block sizes, shorter key lengths, and reduced computational overhead as compared to common block ciphers like AES. In order to manage keys and securely share information, lightweight block ciphers can be used in conjunction with public-key cryptography methods. PRINCE, PRESENT, Simon, Grain, and Piccolo are the popular lightweight block ciphers. Real-time encryption and decryption of the on-chip flash data of the LPC55Sxx (NXP micro-controller with ARM Cortex-M33 core) are performed using the PRINCE algorithm. The NIST Lightweight Cryptography Competition (2019–2023) has chosen ASCON [129] as the next lightweight cryptography standard. ASCON refers to a collection of lightweight authenticated encryption and hashing algorithms. With ASCON, resource-constrained IoT devices are anticipated to explore a new degree of security against attacks in the coming decades.

In Sec-NoC, we use the PRINCE cipher, which encrypts data in just one clock cycle and has a relatively small chip footprint [130]. Compared to PRESENT-80 [131] and AES-128 [132], PRINCE consumes 6 and 14 times less space, respectively. With a key size of 128 bits, PRINCE is a 64-bit block cipher. Each pair of communicating nodes exchanges a symmetric key before encryption. The Sec-NoC primarily encrypts all the packets at NI before injecting them into the NoC. Since there are millions of packets being transferred through NoC, we cannot apply a classical encryption algorithm to encrypt each packet, as it eventually slows down the applications' performance. So, we employ a lightweight cipher like PRINCE to do the same. At the source NI, the packet is encrypted, and at the destination NI, the packet is decrypted. If the packet faces any HT attacks like packet alteration, the Sec-NoC's recovery method is called to alleviate the issues. The packet integrity check of Sec-NoC facilitates blocking packet leakage, which we discuss towards the end. The exchange of a 128-bit secret key used with the PRINCE to encrypt the data is discussed in the following section.

3.4.3 Key Agreement

In our work, symmetric encryption keys are transmitted as special NoC packets. In general, the term *'Key Encapsulation'* refers to this. Public-key encryption enciphers the encryption key, encapsulates it inside a packet, and sends it over the network to the designated recipient.

A special data structure called a *key-pair table* is proposed at NI to facilitate the storage and access to the shared keys among nodes. The key is exchanged in the form of a packet, however encrypted with ELGamal ECC encryption (EEE) [133]. A packet called *Key-data packet* is introduced to exchange the key involving three flits: head flit (H), body flit (B), and tail flit (T). The 3-bit packet type (PT) field in the head flit identifies the type of packet. Table 3.1 depicts the types of packets used.

Table 3.1 Different types of packets in NoC

Packet type (PT) bits	Count of flits	Packet Description
000	1	Control packet (Request)
001	5	Data packet (Response)
010	3	Key-data packet
011	1	ACK packet
100	1	NACK packet

Our approach generates a 128-bit symmetric key at each NI without using a random number generator. As stated, as part of ensuring authentication, the system is pre-embedded with the public-private keys of nodes that are to be utilised for the EEE method. Even though we refer to one of the keys as 'public', only the trusted NI level has access to it. We perform key derivation by concatenating random public keys to form symmetric keys. A derived symmetric key has to be mapped to the underlying elliptic curve that involves significant area and power overhead. Therefore, the use of public keys avoids such mapping overheads.

Let us consider an NI, say A, attempting to send a packet to an NI, say B. A checks its key-pair table to see if any prior communications with B exist or not. Prior to the actual packet transfer, we generate and share a 128-bit symmetric key with B if the entry corresponding to B is not present. The key generator (KG) module randomly selects four public keys. We concatenate these four 32-bit keys to form the intended symmetric key. Subsequently, it is added to the source's key-pair table as a new entry. To achieve this, we separately encrypt each 32-bit key using EEE. KG applies EEE to each 32-bit public key, converting it into a 64-bit encrypted sequence, which then concatenates to form a 256-bit stream. We embed (flitisize) this into a 3-flit key-data packet and send it to the destination, NI (B). At the

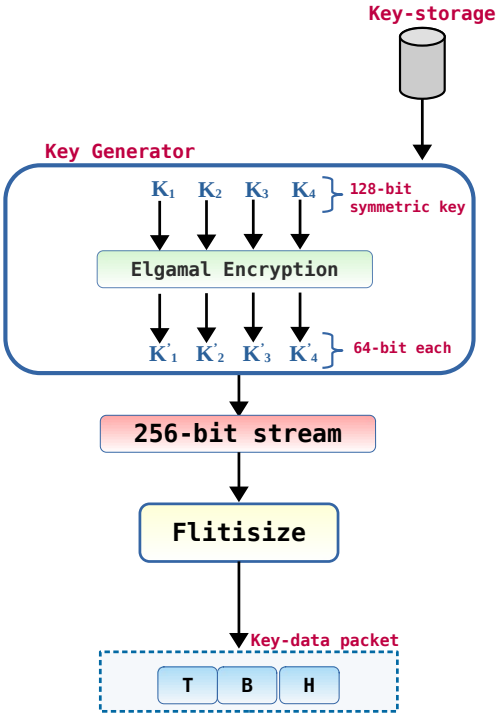


Fig. 3.5 Symmetric key encapsulation: 128-bit key generation and its transmission in the form of a 256-bit encrypted bit-stream in a key-data packet. The size of each flit is 128 bits.

destination, we retrieve the four 32-bit keys from this 256-bit stream by invoking ELGamal ECC decryption. Add this key information to the destination NI's (B) key-pair table as a new entry corresponding to A. Thus, both A and B now share the same symmetric key.

Fig. 3.5 depicts the overall key encapsulation mechanism showing the symmetric key generation and its transmission as a key-data packet. The K_i denotes a random public key (a 32-bit ECC key) available in the key storage, and the $K_i^{'}$ indicates the corresponding 64-bit EEE encrypted bit-stream. Key encapsulation is often simpler to implement and more efficient in terms of computational and memory resources than key-agreement protocols. Cloning-based attacks on key-data packets are not possible because authentication occurs before the use of keys.

3.4.4 Re-transmission Mechanism

At the NI level, we use the ACK-NACK method as an error recovery measure. In our system, we make use of the lightweight SipHash [134] function to generate the 64-bit MAC embedded in the head flit by the source NI. The SipHash takes the payload bits as well as the bits corresponding to the source, destination, and PT fields as input messages to generate the

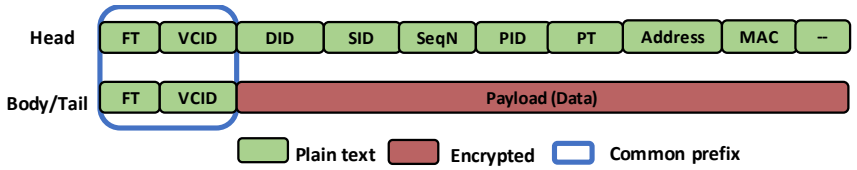


Fig. 3.6 Structure of a Data Packet

corresponding MAC. Even though the key separation principle suggests the use of a separate key for each security operation, the scope for cryptanalysis is insignificant in the NoC fabric. Hence, the 128-bit key used by SipHash is the same key used with PRINCE cipher. This avoids key overheads associated with key management. As soon as a packet arrives, the destination NI verifies its integrity. We send a NACK packet back to the source NI to initiate a re-transmission of the same packet when the integrity check fails. We use a retransmission buffer (RBUFF) at the NI to facilitate end-to-end retransmission. The RBUFF keeps track of the packets transferred from an NI. As soon as a packet is sent from an NI, one copy of the packet is also saved in RBUFF. This helps facilitate the re-transmission of packets upon receiving NACK. All types of packets, including key-data packets, include the 64-bit MAC in their header. In the key agreement stage, B also sends an ACK for a successfully verified key data packet. A confirms the use of a symmetric key with B only when A receives an ACK from B. The new key encrypts subsequent packets. Since an attacker can modify any arbitrary packet, including ACKs or NACKs, we incorporate a timer for each entry in the RBUFF. This ensures the re-transmission of packets even if an ACK or NACK is not received. Figures 3.6 and 3.7 depict the structure of a data and key-data packet, respectively. Each flit is 128 bits. These packet structures are developed for an 8×8 mesh NoC. The fields FT (Flit Type: 2 bits) and VCID (Virtual Channel ID: 3 bits) are additional control fields that are common prefixes kept parallel to the 128-bit flit channel to facilitate wormhole routing. These common prefix fields are part of all the flits of a given packet. The head flit consists of the following standard fields: SID (Source ID: 6 bits), DID (Destination ID: 6 bits), SeqN (Sequence Number: 3 bits), PID (Packet ID: 4 bits), PT (Packet Type: 3 bits), and Block Address (32 bits). We assume a maximum memory address space of 256GB. Since we use 64B cache blocks, the block address consists of 32 bits. Out of the 128-bit head flit, apart from the standard 50-bits discussed above, we add a 64-bit MAC. We keep the fields in plain text format. As previously discussed, we encrypt the payload bits before adding them to the body and tail flits.

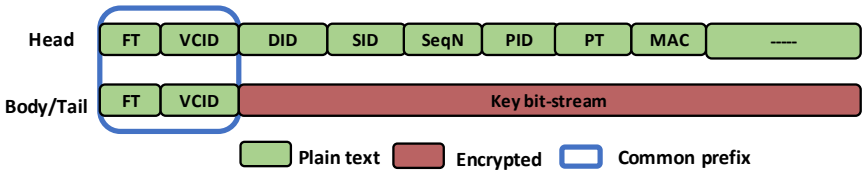


Fig. 3.7 Structure of Key-Data Packet

3.4.5 Blocking packet leakage

As discussed in the threat model, in order to leak the packet, the HT has to create a duplicate packet and then modify the destination ID. The new ID corresponds to the core where the colluding application is running. Hence, the duplicate packet is rerouted from the HT router towards this core. Here, we infer that the leaked packets are always modified packets. When these duplicate packets reach the corresponding destination NI, the Sec-NoC initiates a MAC check using SipHash. Obviously, the leaked packets show a MAC mismatch because of the alteration. Hence, the packets get dropped at the destination NI itself. The attacker, or the colluding application, is unable to get even a single packet.

3.5 Experimental Setup and Analysis

As already discussed, we use the event-driven simulator gem5 [118] to model both the 16-core and the 64-core x86 systems to make an analysis of the proposed Sec-NoC. Other parameters used to configure the multi-core system are shown in Table 2.3 of Section *Evaluation Methodology* of Chapter 2. We make use of the SPLASH-2 benchmarks as specified in Fig. 2.20 of Section *Evaluation Methodology* of Chapter 2. The Garnet framework is the on-chip interconnect model of gem5 used to model the HT with the capability to modify and leak packets. In order to counter the impact of HT, we have incorporated the SeC-NoC cryptosystem into the NI of the TCMP's NoC framework in Garnet.

We established the following metrics to demonstrate the effectiveness of the Sec-NoC's ability to tolerate HT attacks:

1. **Recovery Injection Rate**- It quantifies the volume of re-transmitted packets, ACKs, and NACKs injected into the network per cycle. Unlike regular request-response and coherence packets in NoC, we take into consideration only these packets that play a role in error recovery or integrity checks as specified by the Sec-NoC.

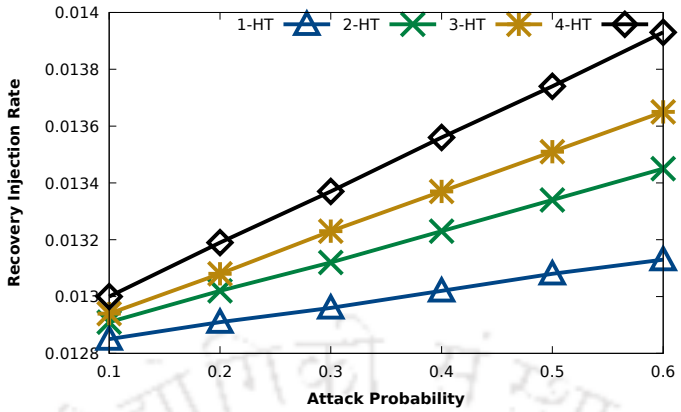


Fig. 3.8 Recovery Injection Rate

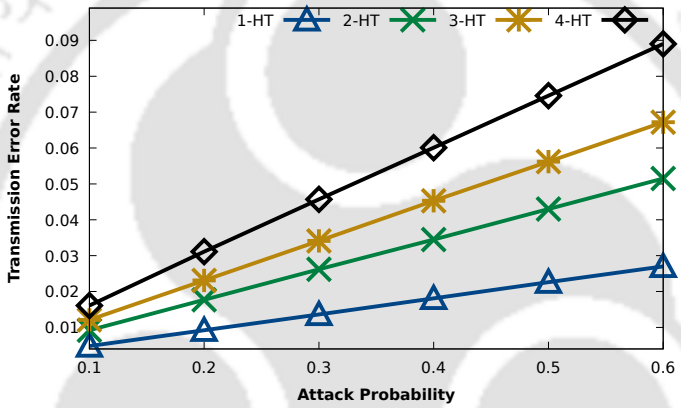


Fig. 3.9 Transmission Error Rate

2. **Transmission Error Rate**- It denotes the proportion of packets that have failed to successfully reach their destination due to HT impact compared to the total number of injected packets from the core. This ratio excludes ACKs, NACKs, and re-transmissions.

3. **Effective Packet Utilisation Ratio**- It shows the ratio of sent packets (regular NoC packets except additional packets of Sec-NoC) to the total number of packets such as retransmitted packets, ACKs, and NACKs.

4. **Average Packet Latency (APL)**- Average time a packet takes to travel from its source core to its destination core through an interconnect. This involves a NoC delay as well as encryption-decryption time at NI.

The recovery injection rate is shown in Fig. 3.8. It is clear that as the number of attacks rises, more NACKs and re-sent packets are injected into the network to handle the faults. Fig. 3.9 shows the impact of HT's packet modification on the interconnect. Around 9% of packets

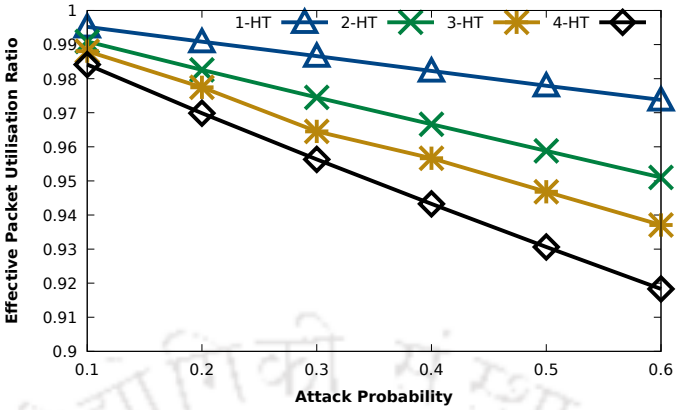


Fig. 3.10 Packet Utilisation Ratio

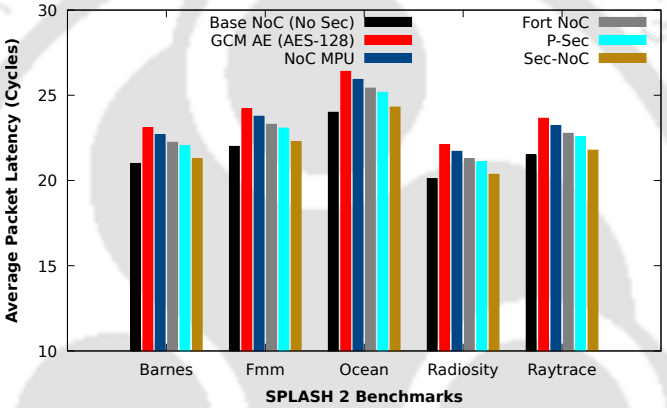


Fig. 3.11 Average Packet Latency (cycles): 4x4 mesh NoC with SPLASH-2 Benchmark programs.

get impacted by four random HTs with an attack probability of 0.6. This demonstrates the need for an efficient recovery method that Sec-NoC provides in order to prevent the latency of outstanding requests from escalating. Fig. 3.10 displays the effective packet utilisation ratio. If there are no HT impacts, only ACKs are on the network as extra packets. So, the ratio becomes 1. It is inversely proportional to the recovery injection rate. With more HT impacts, more NACKs, and more re-transmissions, effective utilisation is reduced. ACKs, NACKs, and re-transmissions create traffic, which also slightly increases the NoC delay.

Fig. 3.11 shows the comparison of Sec-NoC's APL with five state-of-the-art methods by considering five SPLASH-2 benchmarks that can generate diverse NoC traffic. Similarly, Fig. 3.12 shows the comparison of Sec-NoC's APL while running SPEC 2017 workloads. The behaviour of the SPLASH 2 binaries and SPEC 2017 workloads in terms of NoC traffic is well explored in Chapter 2 (Evaluation Methodology). By considering these two latency graphs, we find that, on average, GCM AE (AES-128) [46], NoC MPU [37], Fort NoC

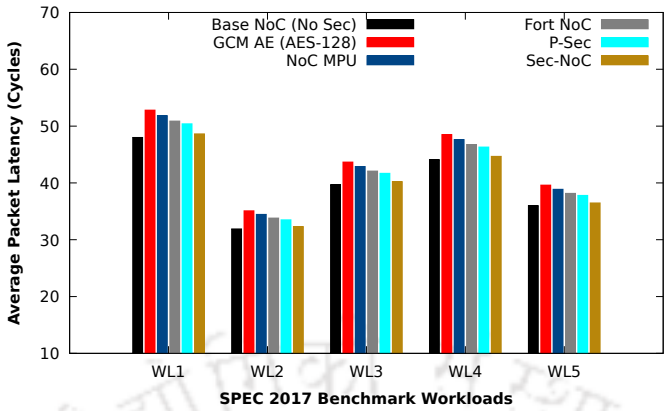


Fig. 3.12 Average Packet Latency (cycles): 8x8 mesh NoC with SPEC 2017 benchmark workloads.

[50], and P-Sec [55] experience a latency hike of 10%, 8%, 6%, and 5%, respectively. Sec-NoC induces a very slight latency hike of 1.3%. This is because of the use of lightweight PRINCE cipher. As discussed above, we find that leaked packets are invariably modified ones. Therefore, the Sec-NoC's integrity check at destination NI only permits unmodified packets to advance to the upper level. Others are discarded. Leaked packets can thus be prevented before they reach a CA for unlawful activity. So, the attacker in RAHT, as discussed above, fails to get a single packet for the extraction of useful information with ML techniques. The Sec-NoC can integrate any lightweight cipher that complies with NoC requirements to ensure confidentiality.

3.5.1 Overhead Analysis

A baseline MPSoC with 16 32-bit MIPS cores and a 4×4 mesh NoC as the interconnect is implemented with Verilog HDL and synthesised with a Cadence RTL compiler (90 nm technology) without integrating any security mechanisms. This is compared with the same MPSoC integrated with the Sec-NoC implementation. We see that Sec-NoC incurs an additional 10% area and 9% power than the baseline 16-core MPSoC *(same as baseline MPSoC overhead in Table 2.7 of Chapter 2)*. To assess the maximum overhead, we integrate an AES-128 block at each NI to encrypt packets of the baseline MPSoC. It was found that Sec-NoC consumes 80% less area and 47% less power than AES-based NoC.

3.6 Chapter Summary

This chapter deals with a lightweight, secure NoC system called Sec-NoC that is able to provide confidentiality and integrity in the on-chip interconnect's communications. The Sec-NoC integrates an encryption-decryption cryptosystem, recovery mechanisms, and a key-encapsulation procedure. An integrity check is proposed at NI to block any leaked packets from reaching the colluding application. Simulation results show that the proposed design can tolerate the faults through effective management of ACKs, NACKs, and re-transmissions. Sec-NoC is better than NoC with AES-128, Fort-NoC, and P-Sec by 87%, 77%, and 73%, respectively, in terms of average packet latency. The area and power overhead of Sec-NoC are negligible when compared with an AES-128-based NoC system. Since the latency hike is only around 1.3%, it is well suited for encrypting packets from time-critical applications.

Chapter 4

TROP:Trust-aware Opportunistic Routing

This chapter describes the second work of the thesis proposing the dynamic trust-aware routing of packets in an NoC framework comprising HTs. We propose a direct trust-building environment between NoC routers such that packets can bypass the HTs while routing to mitigate the effect of packet integrity attacks. When compared to a state-of-the-art architecture, the proposed work reduces re-transmission traffic and hence dynamic energy.

4.1 Introduction

Trust is characterised as the degree of subjective belief or opinion in the behaviour of a given entity [135]. Trust management is increasingly a critical component of network security services, particularly in the disciplines of mobile ad-hoc networks (MANET) [136],[137], wireless sensor networks (WSN) [138],[139], and IoT [140]. Routing decisions based on trust computation, update, and analysis are desirable in an NoC system as well, since they reduce the likelihood of packets being influenced by potential HT attacks like packet corruption. Fig. 4.1 illustrates a trust-based selection of a downstream router from the current router R for a packet P_1 in NoC. Assume that packet P_1 has two choices from R for making a move in the direction of its destination as per the DyXY routing algorithm. One is the East port, leading to Router P, and the other is the North port leading to Router Q. The router R keeps and updates trust information about neighbours using appropriate data structures and algorithms. As soon as a packet P_1 is received, the router R chooses a neighbour with a high trust value, with the speculation that this packet will reach the destination unaltered. The existing Trust-Aware Routing (TAR) [56] method enables packets to bypass the HT-impacted

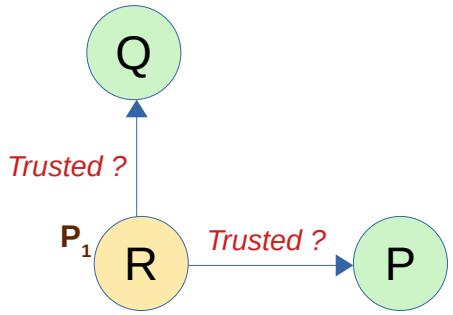


Fig. 4.1 Selection of a trustworthy neighbour for a packet P_1 from router R.

router during their traversal and reach their destination unaltered. Imagine that a packet originating from a source IP in a secure zone has to traverse through the non-secure zone in order to reach the destination IP in another secure zone. The routers in the non-secure zone are potentially malicious and can initiate attacks on packets or security breaches. Its trust model is based on computing both direct trust and delegated trust values between routers. The direct trust between routers in TAR is calculated based on a sigmoid function, as shown below.

$$S(x) = 2.\frac{1}{1+e^{-x}} - 1 \tag{4.1}$$

where x keeps track of the number of successful transmissions at a given router, and $S(x)$ lies between -1 (no trust) and 1 (high trust).

As shown in Fig. 4.1, routers R and P are neighbours. Initially, R has no trust information about P. Therefore, $x = 0$, and as a result, $S(x) = 0$. When $x > 0$, the value of $S(x)$ will become greater than 0. When R learns about P's behaviour, it changes the value of x and recalculates $S(x)$. For example, if R gets positive feedback about P as per their approach, direct trust of R on P ($T_{R\to P}$) is increased, and is calculated as, $T_{R\to P} = S(x+t)$, where t is a small positive number. Similarly, to reduce trust, they follow, $T_{R\to P} = S(x-t)$. The packets' re-transmission pattern determines whether direct trust with a neighbour should increase or decrease at each intermediate router during packet transfer. Once a communication is successfully completed between R and P, trust about the next hop ($T_{R\to P}$) is delegated to nearby routers (except P) by R. This is done by broadcasting a trust packet that contains $T_{R\to P}$. Because trust computation involves both direct and delegated trust, it can be an expensive process, especially in a resource-constrained environment.

This approach does not give details on the type of packets the malicious node targets, the specific location of the malicious element within the NoC structure, or the characteristics of its actions. Furthermore, when a router re-transmits a packet in a trust-building environment,

it updates its trust values. A router performs its trust value calculation based on a variable reflecting the number of successful transmissions, and this method doesn't state the storage of this variable within the router.

We analyse the performance deterioration in NoC when a HT alters the flits that arrive in a router's input buffer. Therefore, we adhere to the PCR basic thread model [16]. Systematic implementation of error correction by re-transmission can lead to denial-of-service (DoS) attacks in NoC. For instance, HT-impacted routers can alter packets on purpose and cause continuous retransmissions, leading to a DoS attack. The NoC performance deterioration is due to a high rate of packet drop caused by corruption, buffer space wastage, response delays, and re-transmissions [16]. Repeated re-transmissions cause the network to run out of resources, even induce deadlocks, and eventually lead the system to fail. Packet latency hikes and increased energy consumption are the two major issues with re-transmission. It is also a waste of NoC resources to attempt to decrypt a corrupted packet while following MAC-then-Encrypt (MtE) approach.

A message authentication code (MAC) ensures the integrity of the transmitted packet. The sender-NI adds the MAC tag to the packet's header and injects the whole packet into NoC. Upon receiving a packet, the receiver-NI recalculates the tag. A mismatch between the received tag and the recalculated tag identifies the packet as modified by some HTs during its traversal. Sending a negative acknowledgement initiates re-transmission.

This chapter discusses a method called TROP for routing packets across NoC that takes into account the trustworthiness of neighbouring routers, using a computed trust metric to select the most suitable router for forwarding packets. We refer to this mechanism as opportunistic because it chooses the most appropriate neighbour based on available trust information for forwarding packets. Trust computation is redefined to enable the dynamic calculation and updating of trust levels between two direct neighbours. The TROP utilises a re-transmission buffer (RBUFF) at the NI to facilitate end-to-end re-transmission, and implements credit-based flow control to prevent the loss of any packets, including acknowledgement packets, due to buffer overflow. TROP ensures effective packet delivery and prevents packet loss by keeping packets at the source (at RBUFF) until they receive a response from the destination.

Our main contributions can be summarised as follows:

a. We model a probabilistic HT that alters NoC flits, resulting in packet corruption, and assess the impact of the same on NoC.

b. We present a new dynamic approach to trust computation for routers based on the packet forwarding activity of adjacent neighbours.

c. We create trust-aware paths by using the proposed trust computation that facilitates opportunistic selection of the most appropriate neighbour.

d. We propose an effective management of a re-transmission buffer at the NI level to enable end-to-end re-transmission that helps in limiting the number of re-transmissions.

e. We experimentally demonstrate that proposed TROP not only decreases packet re-transmissions but also reduces end-to-end latency in successful delivery of messages without inducing any stall.

The detailed threat model and its impact, the proposed TROP system, and evaluation are described in the following sections.

4.2 Threat Model and Impacts

We perceived a threat model in which HT located in the NoC router performs a packet corruption attack [16]. The HT is positioned at the VCs associated with the input ports of the infected routers. Once triggered, the HT modifies the content of incoming packets, leading to packet corruption. The HT can modify either the control information or the payload part of the packet. When the HT modifies bits other than the essential routing control bits, the corrupted packets can reach their intended destination. However, the destination NI examines the packet integrity using MAC check upon arrival. The destination NI discards the corrupted packet and initiates packet re-transmission by sending a packet containing a negative acknowledgement (NACK) to the source NI. The source NI then re-transmits the packet that it has stored in its RBUFF. Packet corruption of this kind is identified as one of the transient effects that can lead to system failures in NoC gradually [141]. Destination NI sends an acknowledgement packet (ACK) to the source when it successfully receives a packet without modification. If the HT modifies control fields such as destination ID, it could lead to the packet reaching the incorrect destination and subsequent drop. When the sender NI receives no acknowledgement from the intended receiver within the timeout period, the packet in RBUFF is naturally re-transmitted [16].

In this threat model, even though HT is always active, it modifies the incoming packet with a certain probability, 'p'. Packet corruption can also lead to traffic congestion due to a higher number of packet re-transmissions and a longer turnaround time for packets to reach their respective destinations error-free. Consider four randomly selected tiles (say, Tiles 11, 22, 25, and 45) that are HT infected as shown in Figure 4.2. The wormhole flow of a five-flit packet P_1 (one head flit (H), followed by three body flits (B), and ending with a tail flit (T))

from router 10 (S) to router 36 (D) is illustrated there. Consider a case where an HT at router 11 modifies the third body flit of P_1. When D receives P_1, it checks the data integrity and subsequently drops P_1. D then sends a NACK packet to source S to initiate re-transmission. To travel through the network, both packets P_1 and NACK use the conventional XY routing method. At every source, NI's RBUFF keeps the packet to initiate re-transmissions. The re-transmitted packet is likely to be infected by the HT again. Therefore, we ensure that NACKs and packet re-transmissions will persist until the packet successfully reaches its destination.

As previously stated, the HT also modifies the control bits in a packet's head flit. Altering the control information, such as priority, source ID, destination ID, etc., belongs to the category of static effects (static fault) in NoC [141]. Even with one packet, the static effects can disrupt other network communications. Static effects have a different impact than transient effects like payload modification. The effective ACK-NACK combination can handle this. NoC researchers have already explored the impact of static effects in NoC. Our study aims to examine the tolerability of NoC when there are static and transient effects. It is indeed challenging to identify safe paths for packets in a NoC with multiple malicious implants that behave very erratically without any correlation.

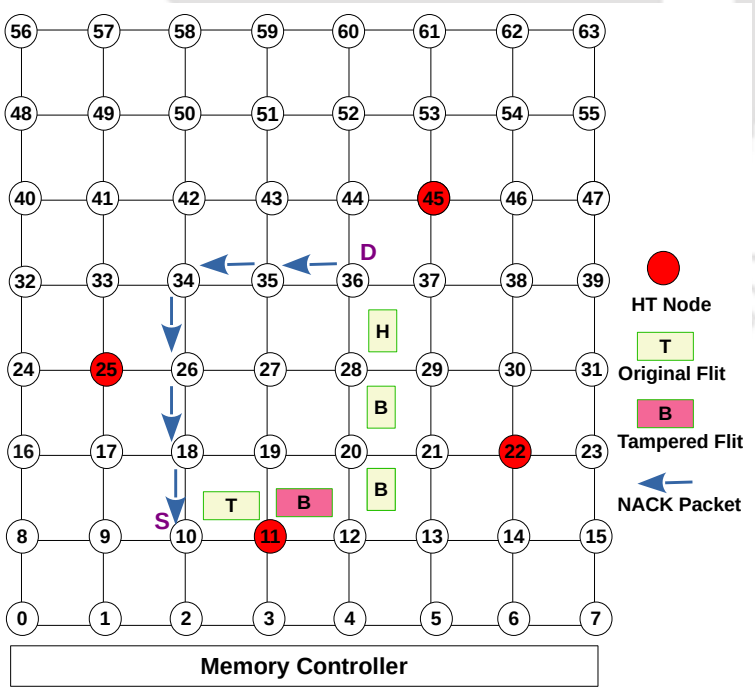


Fig. 4.2 Illustration of HT location, packet corruption and NACK packet path in an 8x8 mesh NoC system.

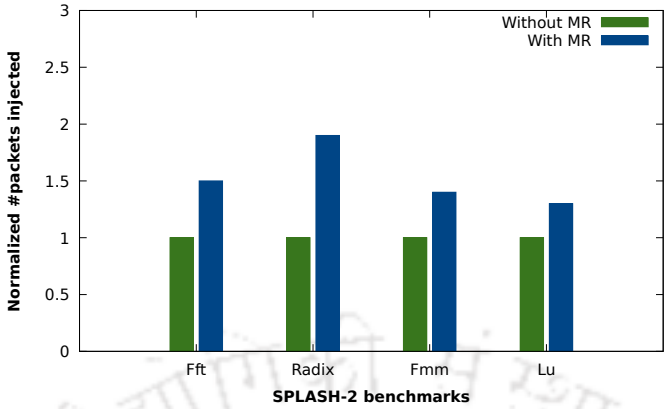


Fig. 4.3 Number of packets injected in comparison with and without the presence of an MR.

The source NI must re-transmit the packet if the MAC does not match at the receiver's end, leading to overhead in MAC calculation and retransmission. This is the attacker's intention when they corrupt packets. To analyse the overheads of packet corruptions and their subsequent re-transmissions, SPLASH-2 benchmark programmes such as Fft, Radix, Fmm, and Lu are made to run on an 4×4 mesh NoC-based SoC with 16 cores. The system follows the XY routing protocol and MtE approach. Each malicious router(MR) in the path can corrupt n consecutive packets after every p packet. For experimentation, we take p=20 and n=14. Overall, encryption, decryption, and authentication processes are assumed to take 20 cycles per transmission.

Both with and without a MR, the number of injected packets, the NoC delay (total NoC traversal latency for all packets), including encryption/decryption and MAC calculation time, and execution time, are measured and are shown in Fig. 4.3, Fig. 4.4 and Fig. 4.5, respectively. With the impact of MR in an NoC, more packets are injected into the NoC in terms of re-transmitted packets, ACKs, and NACKs, as evident in Fig. 4.3. This creates additional NoC traffic and slows down the movement of regular NoC packets, leading to a hike in NoC delay. This is clear in Fig. 4.4. The rise in NoC delay obviously affects the running time of applications, as it incurs additional cycles for the reception of response messages. So, the execution time has also increased, as depicted in Fig. 4.5. On average across all benchmarks, there is a 60% increase in the number of injected packets, a 67% rise in NoC latency, and a 5% increase in execution time. A mechanism that can reduce re-transmissions can improve the overall NoC performance.

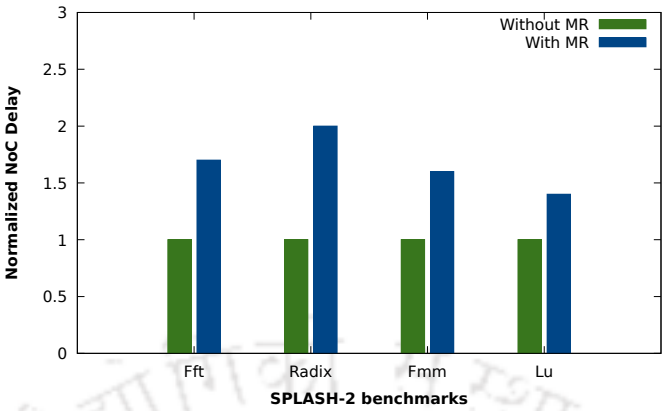


Fig. 4.4 NoC delay in comparison with and without the presence of an MR.

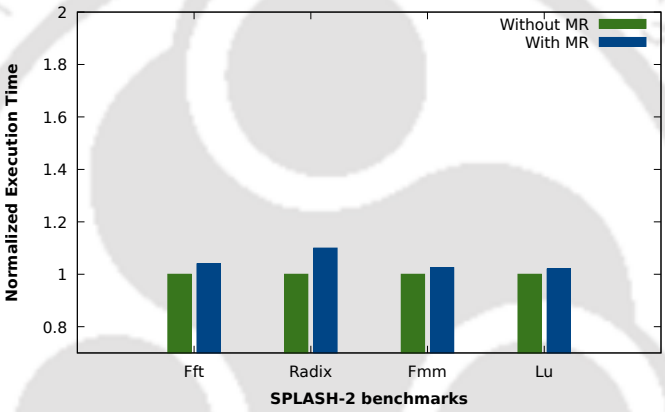


Fig. 4.5 Execution time in comparison with and without the presence of an MR.

4.3 Motivation

Error correction through re-transmissions creates more traffic in NoC. This can increase further if there are multiple HTs in the network. It can even lead to a denial-of-service attack in NoC. Higher NoC traffic consume more energy too. To address this issue, the only option is to reduce the rate of re-transmissions. As stated earlier, TAR uses both direct trust and delegated trust to decide the next hop in the routing path. Figure 4.6 shows the trust delegation approach followed in the existing TAR method. Once the router 4 completes its communication with the router 5 (next hop), it broadcasts the direct trust (T) between routers 4 and 5 to the nearby routers (1, 3, and 7). These neighbours then compute (or update) their delegated trust value with the router 5. There are issues with this kind of delegated *trust packet* broadcasting at the router level, which are listed below:

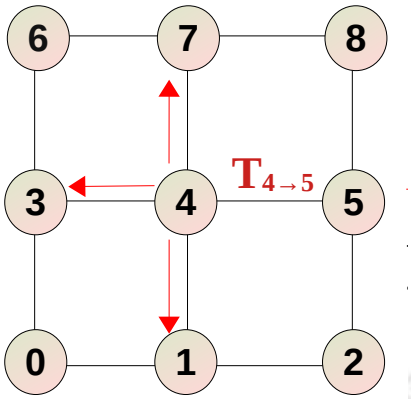


Fig. 4.6 A 3x3 NoC with broadcasts of direct trust between router 4 and router 5 from router 4 in TAR. Three such delegated trust packets are shown with arrows from router 4.

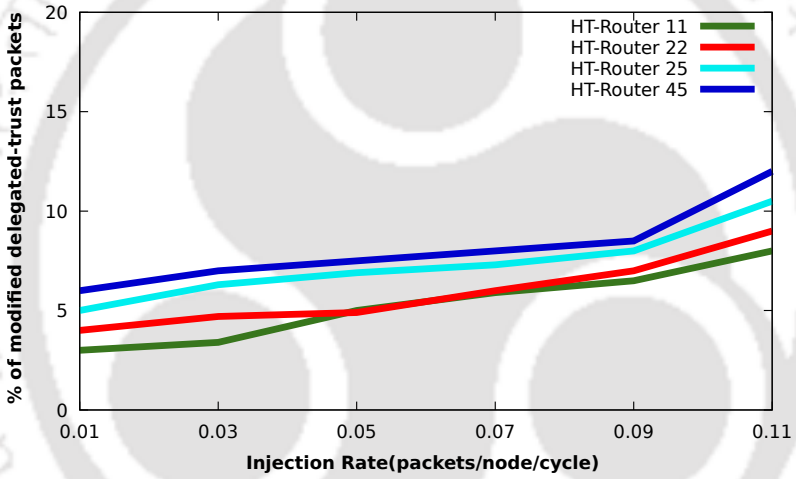


Fig. 4.7 Fraction of delegated-trust packets' modification due to HT routers in an 8x8 mesh NoC.

a. HT can even tamper with the trust packet, thereby causing an incorrect delegated trust value computation. This in turn causes incorrect selection of the downstream router to forward packets. So, in this case, TAR cannot ensure trusted path selection. Since routers are not employed with packet integrity checks, changes to trust packets go unnoticed. Typically, the trusted NI level integrates integrity and security operations. Based on the architecture given at Figure 4.2, the fraction of delegated-trust packets that are modified by HT at routers 11, 22, 25, and 45, chosen at random from, is shown in Figure 4.7.We observe that around 8% to 10% of delegated packets get impacted, which is a significant number. We also observe that this fraction increases with the injection rate (X axis).

b. A new packet, like a delegated trust packet, is generated and transmitted between routers. This requires extra logic at the router level, which is more or less similar to some functionalities in NI. The NI circuit has to be augmented to incorporate this special packet generation. Compared to conventional routers, it demands more area and power.

c. In terms of security, broadcasting vital information to nearby routers, such as trust, is inappropriate. There are situations where we authenticate all the packets. If so, trust packets also need to be authenticated. Implementing an authentication-verification algorithm at the router level is also expensive in terms of latency as it impacts the critical path of packet processing.

d. Regular broadcasting of trust packets can generate additional traffic, which may impact the movement of regular NoC packets.

Consider a scenario in which we neglect the modified delegated trust packet after conducting an integrity check. In that case, we primarily depend on the direct trust value as well as the old delegated trust value. This may result in inaccurate output port selection. A router cannot accurately estimate trust with four of its neighbours by using a single variable because each direction has different traffic patterns. A single trust variable cannot encapsulate both the success and failure patterns of packet forwarding. The environment in which TAR operates requires a selection of secure and non-secure zones. It deals with communication between two secure zones through non-secure zones. We expect to find the HTs in non-secure zones. In contrast to the TAR-based selection, we suggest a general trust-based routing method, assuming the ability to manipulate any router into acting maliciously. The NoC must tolerate the effect without compromising the SoC's performance. We address all of these concerns in TROP [58] to ensure that there is no notion of broadcasting trust packets. Thus, we can easily avoid the traffic created by the delegated-trust packets. With the proposed fine-grained direct trust calculation itself, we achieve trusted path selection from source to destination.

4.4 TRust-aware OPportunistic routing (TROP)

Correct trust calculation amongst a set of routers is inextricably associated with risk estimation in an NoC infected by multiple HTs. When trust between two neighbours is high, the risk of packets getting corrupted in the path between these two routers is often less. We employ a monitoring-based trust management system to measure the trust between a pair of neighbouring routers. The system depends on the direct observations of packet flow and the

history of packet re-transmission. A router calculates trust based on direct observations from four of its neighbouring routers, which include determining whether a received packet has arrived the router for the first time or is being re-transmitted.

We propose a mechanism called **TR**ust-aware **OP**portunistic Routing (TROP) with the aim of making decisions based on trust when choosing the next best neighbouring router opportunistically in order to decrease the number of packets passing through HT routers. This in turn reduces the packet re-transmission rate. By preventing re-transmitted packets from following the same route, the mechanism lowers the possibility of modification by the same HT. The TROP works locally by selecting the most trusted downstream router for a packet to route. A trusted or trustworthy path refers to the most trusted downstream router that TROP has locally chosen for a given packet.

4.4.1 Re-Transmission system

In the communication domain, packet integrity issues can be handled either by forward correction, or re-transmission mechanisms. Forward Error Correction (FEC) is used when retransmission is prohibitively expensive or impractical. FEC comes at the expense of additional data transmission and circuiting at the receiving end. In the NoC architecture, re-transmission is possible due to the short-distance communication as processing cores are compactly placed on a single die [142] [143]. Packet modifications can be identified using error detection with the help of a well-established lightweight message authentication code (MAC) check mechanism [142] like sipHash [144], implemented at the NI level. Since routers do not carry any integrity check mechanisms, they forward the packets to the destination NI. The receiver NI can initiate a NACK when the packets fail the MAC check, requesting the source NI to re-transmit the packet. The destination NI sends a positive ACK for successfully received packets. We use RBUFF at the NI to facilitate end-to-end re-transmission. RBUFF contains both regular and acknowledgement packets. Since we employ credit-based flow control, no packets, including ACK or NACK, are lost due to buffer overflow. So, acknowledgement packets are essentially required to reach the source NI before the time-out period. ACK and NACK packets are control packets with only one head flit .

We conduct extensive analysis to experimentally decide the timeout value, considering the worst-case round-trip communication latency under peak traffic conditions. We ensure that the timeout period is sufficient even with a reasonable worst-case time delay in receiving ACK and NACK. Upon receiving an ACK, the source NI removes the corresponding packet from the RBUFF. In traditional network communications, we know that congestion can also result in packet loss, which in turn causes packet re-transmissions. But, as already stated, because NoC uses credit-based flow control, there is no packet loss due to congestion. Table

4.1 shows the possible cases of transmission and re-transmission for a packet P_1 with its source NI as S and destination NI as D. The first two cases show the payload modification by HT in P_1. The third case indicates the alteration of the destination ID by HT in P_1 in which $D^{'}$ is the new destination. The fourth case indicates a success in the packet transmission; P_1 is no longer maintained in the RBUFF of S. The fifth case depicts an unsuccessful delivery of P_1, prompting a retransmission from S. The sixth case shows that S received a modified acknowledgement packet. So, the P_1 is re-transmitted. Once P_1 reaches its destination, the NI checks its RBUFF to identify any ACK or NACK entries corresponding to this packet. If NI finds a matching ACK entry against P_1, it sends the ACK packet to S without forwarding P_1 to the upper level of NI, and P_1 is dropped at the NI itself. If a matching NACK entry is found against P_1, the P_1 is further forwarded to the upper level, and an ACK is sent back to S. The last case shows that the ACK or NACK packets reached the wrong destination, so naturally, time-out at S, P_1, is sent again. Further, it is handled as stated in the sixth case when it is received at D.

Table 4.1 Cases of packet transfer in HT implanted NoC

No.	Case	Action at receiver or sender
1	P_1 reaches D with no modification	ACK from D is sent to S
2	P_1 reaches D with modification	NACK from D is sent to S
3	P_1 reaches $D^{'}$	Timeout at S, P_1 is re-transmitted
4	ACK reaches S with no modification	Remove P_1 from RBUFF of S
5	NACK reaches S with no modification	P_1 is re-transmitted from S to D
6	ACK or NACK reaches S with modification	P_1 is re-transmitted from S to D
7	ACK or NACK reaches $S^{'}$	Timeout at S, P_1 is re-transmitted

4.4.2 Trust-Based Routing

In order to incorporate trust-based routing, the router microarchitecture has to be modified. The detailed NoC router microarchitecture with the TROP module is depicted in Figure 4.8. As soon as a packet arrives at the router, the standard route computation unit selects an output port according to the trust-based routing strategy. TROP consists of three major components, namely the CounterUpdate Algorithm, Trust Counters, and a retransmission table (ReTab). The CounterUpdate algorithm updates the appropriate trust counters linked to the selected port with the help of ReTab.

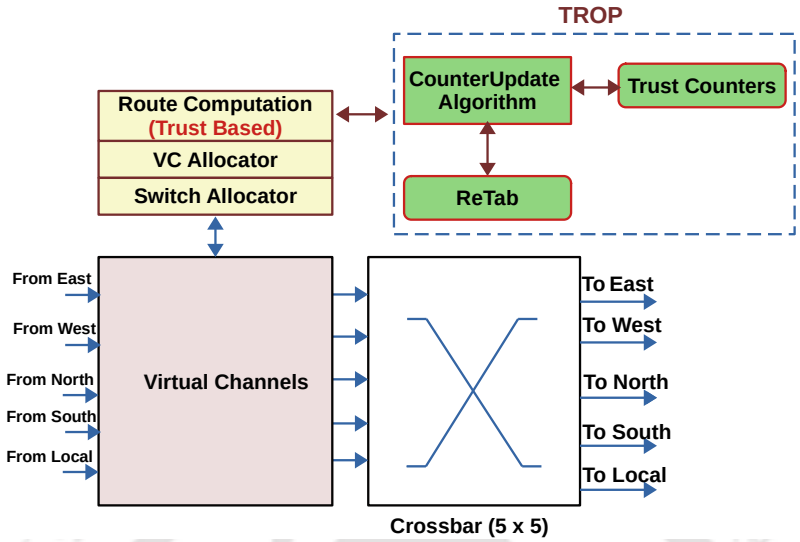


Fig. 4.8 NoC router with proposed TROP module

The trust calculation is based on the direct information observed from packet flow and the history of re-transmission of packets. A re-transmitted packet indicates the presence of a malicious node in the path traversed by the packet, and accordingly, trust levels are updated. The objective is to identify the most trustworthy next-hop neighbour among the choices given. This iterative process progressively generates a path from the source to the destination router that is aware of trust. Trust establishment and update are discussed in the following subsections. Our trust-based routing technique is analogous to DyXY routing, in which each packet follows the shortest path between source and destination [56]. One of several available paths is selected based on the greater trust value (the less-risky path) instead of conventional link congestion measures, which demonstrates an opportunistic router selection behaviour. If the destination is on the same row or column as that of the current router, due to single-path availability, the packet is forwarded without considering the trust value. This is because of the restrictions imposed by the DyXY routing algorithm. When two routers with the same trust value are examined for selection, the router with the most free virtual channels (VC) in their input port is chosen next. The free VC count can be obtained as part of the credit based flow control mechanism in NoC. Therefore, taking the VC count doesn't require any additional circuitry as it is already part of the normal NoC router.

Consider the case shown in Figure 4.2 with router 12 as the current router and 30 as the destination. DyXY routing returns two output ports, East and North, leading to routers 13 and 20, respectively. Router 12 computes trust with routers 13 and 20, and the neighbour with the higher trust is chosen as the downstream router. Figure 4.9 shows the sequential order of trust-based routing, packet forwarding, and trust counter updates happening inside a router.

In order to select an output port for the current packet, the DyXY routing algorithm employs trust computation. The algorithm computes trust using the current values of trust counters. The packet is forwarded to the selected output port. Following that, the CounterUpdate() algorithm updates the trust counters. Now, the router R is ready to receive new packets. From Figure 4.9, it is clear that the TROP counter update circuitry does not operate in the critical path. So, the packets do not experience any delay in the routing decision or its subsequent forwarding.

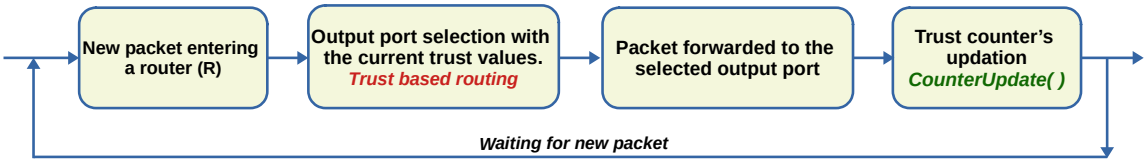


Fig. 4.9 The flow of packets utilising trust information

In our method, there is no case where some ports of the router are permanently marked as untrusted because of very low trust values, thereby blocking packets from being forwarded to those neighbours. We don't adhere to any stopping criteria for packets. This is because of the intermittent nature of HT. As a result, trust values can vary over a period of time. Because HTs are active and dormant intermittently, the safest route from a source to a destination at one moment may not be safe at the next. Our technique opportunistically selects the safest path based on the current trust values.

4.4.3 Trust Establishment

In order to compute trust, each router maintains a *success counter* and a *failure counter* per output port. The trust counters can store 4-bit non-negative values. A router R and its four neighbouring routers (R_e, R_w, R_n, and R_s) are taken into consideration for trust building, as indicated in Table 4.2. E_s and E_f of router R correspond to the *success* and *failure counters* in the east port, W_s and W_f to the west port, N_s and N_f to the north port, and S_s and S_f to the south port, respectively.

When a packet reaches a router and an output port is selected for the first time for the packet, the *success counter* (E_s or W_s or N_s or S_s) related to that output port is incremented. The router directly selects the output port, as there are no trust issues with the downstream router anticipating that the packet will reach its destination successfully. However, if the packet gets corrupted on its path to its destination, it gets discarded, resulting in the generation of a NACK for packet re-transmission. This indicates a failure instead of a success, prompting the adjustment of the relevant counters.

Table 4.2 Output port with trust counters

Output Port	Trust Counters	Neighbouring Router
E	E_s, E_f	R_e
W	W_s, W_f	R_w
N	N_s, N_f	R_n
S	S_s, S_f	R_s

The *success counter* associated with an output port is decremented when a router experiences its first re-transmission through the same port. The first re-transmission through an output port indicates that it is a re-transmitted packet, and this output port is chosen for the second time. Therefore, each time the router sends a packet across that port again, the corresponding *failure counter* (E_f or W_f or N_f or S_f) is incremented. Upon system startup, we reset all counters to their default value of zero.

Let T_{re}, T_{rw}, T_{rn}, and T_{rs} be the trust of a router R with neighbours R_e, R_w, R_n, and R_s, respectively. The following equations are used to measure the trust levels of each router:

$$T_{re} = \frac{E_s}{E_s + E_f} \tag{4.2}$$

$$T_{rw} = \frac{W_s}{W_s + W_f} \tag{4.3}$$

$$T_{rn} = \frac{N_s}{N_s + N_f} \tag{4.4}$$

$$T_{rs} = \frac{S_s}{S_s + S_f} \tag{4.5}$$

We initially assume that routers have a trust value of 1. The trust value changes as a result of packet corruption and subsequent packet re-transmissions. As a special case, anytime the denominator of the trust equation reaches 0, the trust value for the router is reset to 1. A trust value close to 0 implies a low level of trust and the potential presence of an HT through the neighbouring router reaching the destination. Therefore, it becomes less appropriate to be the downstream router.

4.4.4 Trust Update

Each router maintains a Re-transmission Table (ReTab) to keep track of packet re-transmissions and the management of trust counters. It stores the details of recent packets passed through the router. Each ReTab entry has five fields: Source Network-Interface ID (SNID), Packet ID (PID), Direction Bits (DB), Time-Counter (TC), and Re-taken Bit (RB). When a new packet arrives at a router, an entry corresponding to the packet is added in the ReTab. SNID and PID are both taken directly from the packet's head flit while DB is set in accordance with the output port assigned to the packet. For DB, we choose from 00 (East), 01 (West), 10 (North), or 11 (South).

TC indicates how long the entry has been residing in the ReTab. The TC of a newly added entry is initially set to zero. Once a new entry is added to the ReTab, TC is incremented for all the existing entries. When the TC of an entry exceeds the T-Threshold, the entry is removed from ReTab, making sure that ReTab holds only recent packets. Older entries in ReTab are removed over time in order to use the limited space judiciously. This does not cause any loss of critical information. We carry out comprehensive studies to fix T-Threshold at 50 cycles while taking into account the packet arrival rate and average re-transmission rate. We observe that T-Threshold is adequate to handle all 7 contexts outlined and subsequently required to ensure the correctness and progress of TROP.

RB indicates whether the packet is new or re-transmitted. Similar to TC, the RB of a newly added entry is also initially set to zero. When a packet arrives, if there is a matching entry in the ReTab, it indicates that the packet is a re-transmission. However, if RB is set to 0, it confirms that the packet has not yet been re-transmitted through the port specified by the DB for that specific entry. The first re-transmission of the packet blindly follows the same path as the first packet by referring to the DB entry in ReTab. This is to make sure that the appropriate counters are updated at every router in the path where the re-transmitted packet travels. This update is necessary to indicate that there can be one or more routers on the same path that are affected by HT. If the RB entry is set to 1, it indicates that the packet has been re-transmitted before, and some previous retransmissions have already used the same port specified by the DB entry. The way TROP is devised makes it necessary to make two unsuccessful attempts before exploring the opportunistic routing behaviour. The trust-based routing algorithm is not used during the first re-transmission; however, from the second re-transmission onwards, the downstream router is opportunistically selected by the trust-aware routing algorithm. During the first transmission of a packet, the success counters of all the routers in the path are increased with the assumption that this packet would reach the destination without being impacted by the HT. So, naturally, trust in the path would also increase. But, unlike the assumption, if the packet is corrupted, trust in the path has to be

decreased. For this, the first re-transmission passes the same path as the original path to decrease the success counter and increase the failure counter. Success and failure counters work in tandem. The values of both counters are used in trust value computation.

Various contexts resulting from DB and RB combinations, along with the subsequent actions, lead to trust-aware routing that reduces packet traversal through malicious nodes. Table 4.3 illustrates the possible values of DB and RB in the ReTab entries and the corresponding changes to be made on the trust counters for the packet P_i in various contexts. Table 4.3 is constructed with the assumption that the destination router of the packet lies in the East-North direction relative to the position of the source router. Hence, the packets either take East (DB=0) or North (DB=1) as the output port while following the trust-based routing algorithm. We use E for East and N for North in the last column of the table.

Table 4.3 Updation of RB values in ReTAB and corresponding Trust counter updates

Context	DB	RB_{old}	RB_{new}	Trust Counter's update	Remark
1	00	-	0	E_s++	Packet forwarded to E port
2	00	0	1	E_s--, E_f++	Packet forwarded to E port
3	00	1	1	E_f++	Packet forwarded to E port
4A	00	1	1	-	Packet forwarded to N port
4B	10	-	0	N_s++	
5A	00	1	1	E_f++	Packet forwarded to E port
5B	10	0	0	-	
6A	00	1	1	-	Packet forwarded to N port
6B	10	0	1	N_s--, N_f++	
7A	00	1	1	-	Packet forwarded to N port
7B	10	1	1	N_f++	

Context 1: This illustrates when packet P_i arrives at router R for the first time. The East port is designated as the output port; hence, DB is set to 00 (East) and RB is set to 0, indicating that it is a new entry. As the packet is being forwarded through the East port for the first time, E_s is incremented.

Context 2: This represents the case when a re-transmitted packet arrives. This is the first re-transmission of P_i through the East port (DB=00). In this case, no new entry is added as the entry with matching SNID and PID is already present. However, RB is changed from 0 to 1 in the same entry, showing that the re-transmitted packet is forwarded through the same output port (DB=00, East). Since it is a case of re-transmission, E_s is decremented and E_f is

Algorithm 1: CounterUpdate( )

```
Data: Packet (SNID, PID), Output Port
Result: Output Port
entry1 ← ReTab_Available(Packet) ;        /* Check for an entry in the router's ReTab for the current
  Packet  */
if entry1 == NULL then
    ReTab.add(SNID, PID, OutputPort, 0, 0)
    if OutputPort_s == MaxLimit then
        OutputPort_s = OutputPort_s >> 1; ;                    /* Right Shift by one bit - halving */
        OutputPort_f = OutputPort_f >> 1;
    end
    OutputPort_s += 1 ;                                                          /* Context 1 */
    Return OutputPort
else
    if entry1.RB == 0 then
        OutputPort = entry1.OutputPort;
        entry1.RB == 1;
        if OutputPort_s > 0 then
            OutputPort_s −= 1 ;                                                  /* Context 2 */
        end
        if OutputPort_f != MaxLimit then
            OutputPort_f += 1 ;                                                  /* Context 2 */
        end
        Return OutputPort
    else
        entry2 ← ReTab_Check(Packet, OutputPort) ;   /* a re-transmitted packet has already taken the
          Output Port for transfer?    */
        if entry2 == NULL then
            ReTab.add(SNID, PID, OutputPort, 0, 0)
            if OutputPort_s == MaxLimit then
                OutputPort_s = OutputPort_s >> 1;
                OutputPort_f = OutputPort_f >> 1;
            end
            OutputPort_s += 1 ;                                                  /* Context 4 */
            Return OutputPort
        end
        if entry2.RB == 0 then
            entry2.RB == 1;
            if OutputPort_s > 0 then
                OutputPort_s −= 1 ;                                              /* Context 6 */
            end
            if OutputPort_f !=MaxLimit then
                OutputPort_f += 1 ;                                              /* Context 6 */
            end
            Return OutputPort
        end
        if entry2.RB == 1 then
            if OutputPort_f !=MaxLimit then
                OutputPort_f += 1 ;                                          /* Context 3,5,7 */
            end
            Return OutputPort
        end
    end
end
```

incremented. This reflects the fact that packet transmission through the East port has failed in its path, and this is a reattempt, hoping for success.

Context 3: This denotes the second re-transmission of P_i with the East port (DB=00) as the chosen output port. As the East port has already been used, E_f is incremented as a penalty to decrease the trust value with R_e further.

Context 4: This depicts an additional instance of the second re-transmission. ReTab already has an entry corresponding to East (DB=00), shown as 4A. Since North is chosen by the trust-based route computation unit as the output port, a new entry (DB=01) corresponding to North is created with RB=0, indicated as 4B. N_s is incremented as North port is taken for the first time. Now, there are two entries (4A and 4B) for the same packet P_i in ReTab.

Context 5: This indicates a case where there are two matching entries for P_i in the ReTab. One for the East port (DB=00) is shown as 5A, and another for the North port (DB=01) is shown as 5B. Assuming the trust-aware route computation unit selects the East as the output port. The RB=1 value at entry 5A shows that the packet has already been re-transmitted through the East port in the past, which now turned out to be a less trustable path, and therefore its trust value must be lowered. Accordingly, E_f is further incremented for 5A. As we operate only through East port, entry 5B remains unchanged.

Context 6: This also illustrates a situation where there are two matching entries for P_i present in the ReTab. One for the East port (DB=00), shown as 6A, and another for the North port (DB=01), shown as 6B. However, the trust-based route computation unit selects North as the output port. The RB=0 value at entry 6B indicates that only one packet has been forwarded through the North port in the past. The North port path is considered risky as packets got lost; hence, its trust value needs to be reduced. As it is the first re-transmission at North port, RB is changed from 0 to 1, and N_s is decremented while N_f is incremented. The context 5 and context 6 are the same cases, but the outcome is different based on the output port chosen.

Context 7: This is another situation that indicates a case where there are two matching entries for P_i in the ReTab. One for the East port (DB=00), shown as 7A, and another for the North port (DB=01), shown as 7B. Since both ports (East and North) have already been unsuccessfully explored in the past, the RB value is 1 in both entries. However, the trust-aware route computation unit selects North as the output port. Here, N_f is incremented in entry 7B as a penalty, and entry 7A remains unaffected. If the East port is taken as output port, E_f would be incremented.

Algorithm 1 demonstrates the process of updating the counter values. Different contexts are labelled with numbers for better comprehension and understanding of the algorithm. When any of the success counters associated with an output port reach their maximum limit, both the success and failure counters of the same port get halved by shifting the bits by one position to the right in order to preserve the relative ratio of trust value. However, when a failure counter reaches its maximum limit, the value is retained. It is not necessary to store the exact values of the counters every time, as only the relative values are sufficient for calculating trust. The procedure *ReTab_Available()* checks if an entry matching to a

packet is present in the current router's ReTab (by comparing SNID and PID), whereas *ReTab_Check()* tests to observe if the path has previously been followed by a re-transmitted packet (by comparing SNID, PID, and DB against the output port).

As the packet traffic progresses (P_1, P_2,...P_i), the trust counters in their path are regularly updated as per Algorithm 1. Future packets make use of these updated counter values for their routing decisions. The algorithm's time complexity is linear, and it is of the order of packet count.

4.4.5 Learning of Trust-Aware Paths

In our work, the ability to select a trusted, opportunistic neighbour is primarily determined by NoC traffic. For instance, if a packet P_1 suffers data corruption that leads to a re-transmission; the subsequent packet, say P_2, with the same destination, obviously chooses an alternate downstream router at every intermediate router selection during its traversal. This is because the counter update from P_1 at every router is used by P_2 while calculating trust. Every update to the counter reflects that future packets are routed through the most trusted neighbour through opportunistic routing. In general, the computation of trust values through counter updates from the packets P_0, P_1, P_2....P_i is effectively utilised by the packet P_{i+1} in selecting its trusted, opportunistic neighbour. Regardless of the level of traffic, the process of determining trust values takes place at routers. The performance gain in terms of re-transmission count and packet latency at different traffic conditions will be discussed in Figure 4.10 and Figure 4.12 later in the performance analysis section.

4.5 Experimental Setup and Analysis

We use the event-driven simulator gem5 to model a 64-core multi-core system with an underlying fixed-frequency 8x8 mesh NoC-based interconnect. Other parameters used to configure the system are shown in Table 2.3 of Section Evaluation Methodology of Chapter 2. We use the SPEC 2017 benchmarks with workload details as specified in Table 2.4 of Chapter 2's Section Evaluation Methodology. The Garnet framework is the on-chip interconnect model of gem5. We modify Garnet to model the HT that creates the data corruption on flits, as discussed in the threat model. In order to counter the impact of HT, we have incorporated the TROP module into the router structure. HT is mounted on four randomly chosen routers and is triggered randomly to corrupt packets residing in the input buffer of HT-infected routers. A router has at most four neighbours. So, a router under consideration can have a maximum of four HT-impacted neighbours. Since the TROP finds the most trusted downstream neighbour

locally at each router, we just wanted to consider four HTs only. We assume that the HT does not modify the locally injected packets. HT is intermittent in nature and is activated with a probability $p = 0.3$ to induce randomness in its behaviour.

We use a standard uniform-random synthetic traffic pattern with varying injection rates from zero load to saturation for testing the performance of the proposed TROP technique. This gives an insight into how the system behaves under varying network loads and the impact on re-transmission. Additionally, it provides an understanding of how the trust value changes over time and how it affects the packet delivery throughput. By using benchmarks with varying cache miss rates, we can analyse the system response for different network loads.

We compare the performance of TROP with the state-of-the-art mechanism, TAR [56] as well as a standard system that does not use any trust consideration for routing packets. Considering the basic case of a system without an HT, we consider the following four architectures for comparison and analysis:

a. **NHT:** An NoC system with no HT in any of its routers.

b. **HTWOT:** An HT infected NoC system that does not employ any trust calculation for routing the re-transmitted packets. The re-transmitted packets also follow the same path and can be impacted by the HT again.

c. **TAR:** An NoC system affected by HT that employs Trust-Aware Routing for directing re-transmitted packets [56].

d. **TROP:** An NoC system affected by HT that utilises the proposed TROP routing method for directing re-transmitted packets.

4.5.1 Performance Analysis

The overall objective of TROP is to reduce the number of retransmissions needed for a corrupted packet. It is important to note that the key performance index for any trust-aware routing is the definition of trust metrics and how accurately they can be calculated. We accomplish this by routing packets through the most trusted paths that have a lower probability of being affected by HTs. Repeated re-transmission of packets can negatively impact the system's performance as it increases the end-to-end latency of a message. Figure 4.10 shows the comparison of the number of re-transmissions in various architectures using SPEC 2017 benchmark based workloads. As discussed in Chapter 2 (Evaluation Methodology), WL1 consists of applications with higher L1 cache MPKI; more packets are created and

injected into the NoC. Due to higher NoC traffic in WL1, more packets get impacted by HT. Subsequently, this leads to a higher number of packet re-transmissions. Since WL2 generates the least traffic among all the workloads, it is obvious to expect fewer re-transmissions. Naturally, performance is comparable both in TAR and TROP. We observe that TROP reduces packet re-transmissions in other workloads WL3, WL4, and WL5. While considering the re-transmission count, on average, there is a 57% reduction when compared with HTWOT and 32% on average with the state-of-the-art. The effectiveness of our technique is apparent in workloads with higher network loads. Figure 4.11 shows the comparison of the number of re-transmissions in various architectures using SPLASH-2 binaries. Ocean generates the highest number of re-transmissions due to HT impacts as it produces high NoC traffic while running. We can see the behaviour of other binaries as well. As discussed above, TROP has 57% fewer re-transmissions than HTWOT and 32% fewer than TAR. Detailed in-depth analysis reveals that the newly defined trust values used in TROP and the opportunistic routing of re-transmitted packets based on these values provide a better path for packets to reach their destination. One of the reasons that TAR shows poor performance is that it considers the modified delegated trust packets in updating the trust as routers are assumed to have no integrity check, and the alteration goes unnoticed. This results in an inaccurate trust-based output port selection.

As our technique prioritises routing packets through the most trustworthy path available, the end-to-end latency of the packets is a crucial metric for evaluating their performance. Here, we study the impact on average packet latency. Figure 4.12 illustrates the average packet latency in all four architectures using SPEC 2017 benchmark-based workloads. Although TAR and TROP utilise additional logic and counters for calculating and updating trust, it doesn't impact the critical path of the NoC routers. Therefore, we use the same clock cycle for all architectures in the study.

From Figure 4.12, we see that typically the average packet latency is highest for WL1 and lowest for WL2. This is because traffic load differences significantly contribute to congestion. Intuitively, we can observe that NHT (green bar) has the lowest latency as there is no data corruption and no packet re-transmission. In HTWOT, which has a simple re-transmission architecture, the re-transmitted packets are forced to take the same path even though the path contains HTs as it lacks a trust-based route selection mechanism. So, with a high probability of retransmitted packets being HT-impacted again, average packet latency goes high. Since TAR employs a trust-based approach, it explores more trustworthy paths, and hence latency is reduced when compared to HTWOT. TROP uses better trust model that offers the packets the best possible paths to reach their destination. Under heavy NoC traffic, TROP experiences around 9% less average packet latency than TAR. TROP also reduces the

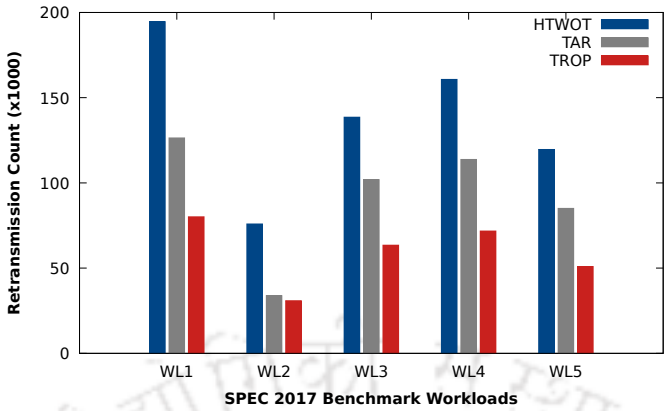


Fig. 4.10 Comparison of packet re-transmissions per one million instructions for different workloads

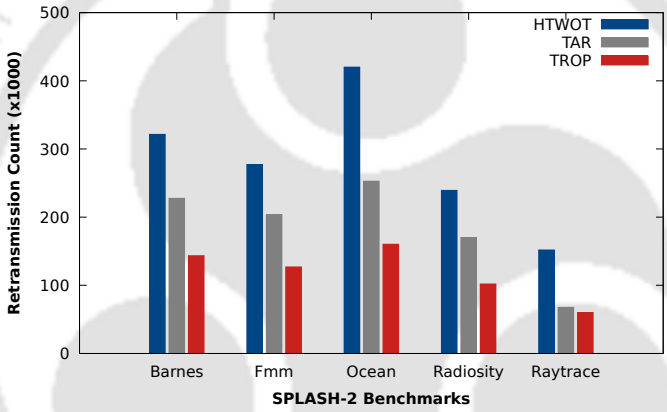


Fig. 4.11 Comparison of packet re-transmissions for different SPLASH-2 binaries

count of retransmitted packets, which results in latency pruning. Our latency statistics clearly depict these savings.

Our experiments involve HTs in 4 out of 64 routers in an NoC with a trigger probability of $p = 0.3$. Hence, only a small fraction of packets are impacted. Examining the overall average packet latency may not provide a detailed understanding of the impact of the HTs on corrupted packets. To gain a clearer understanding, we perform a focused analysis specifically on the packets that suffered data corruption by HT. We define delivery latency as the end-to-end latency of messages whose packets have been corrupted by HT. We calculate it as the time elapsed between the generation of a packet at the source and successful delivery at the destination. This includes the latency of multiple NACKs and subsequent re-transmissions. Figure 4.13 shows statistics on the delivery latency of HT-impacted packets. The solid bar indicates the average delivery latency, while the dashed line bar represents the maximum delivery latency. Across all the workloads, TROP demonstrates lower average delivery

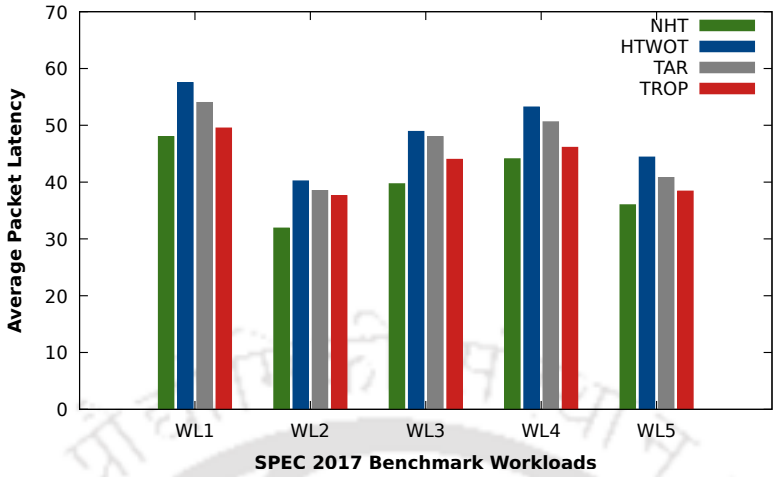


Fig. 4.12 Comparison of average packet latency (cycles) for different workloads

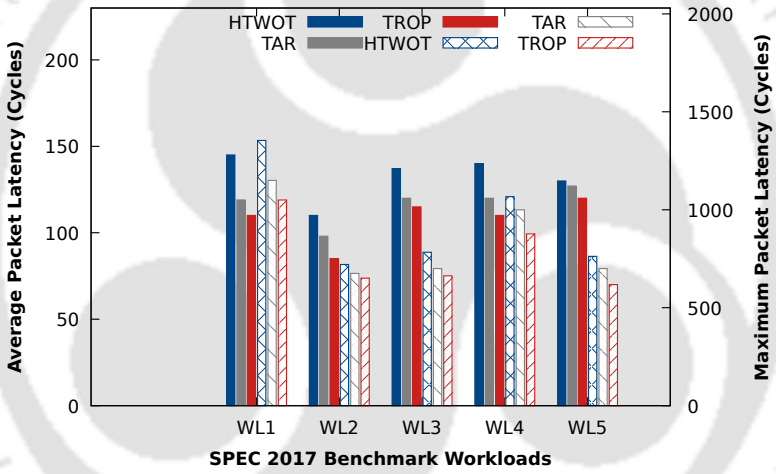


Fig. 4.13 Average and maximum delivery latency of impacted packets.

latency as well as maximum delivery latency. Opportunistic, trust-aware routing contributes to lowering the worst-case delay for successful packet delivery. A clear reduction in maximum delivery shows that fewer re-transmissions are required in TROP than HTWOT and TAR. These statistics support the claim that opportunistic trust-aware routing is cost-effective and meets the rigid timing constraints provided by critical applications.

4.5.2 Trust Dynamics

The proposed model dynamically updates trust between a pair of routers. We study the changes in trust values over time and observe the opportunistic routing taken by routers based on the potential trust-aware paths. For the discussion, we refer to the system instance

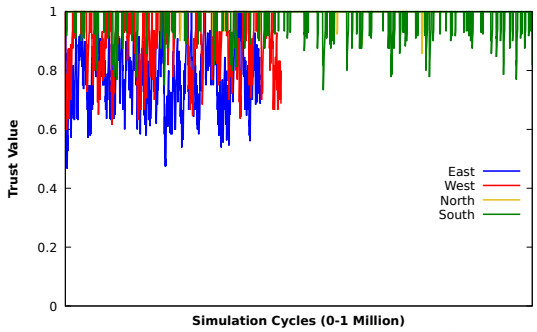


Fig. 4.14 Trust value variations of router 44 with its neighbours: any router with one of its neighbours affected by HT follows a more or less similar pattern.

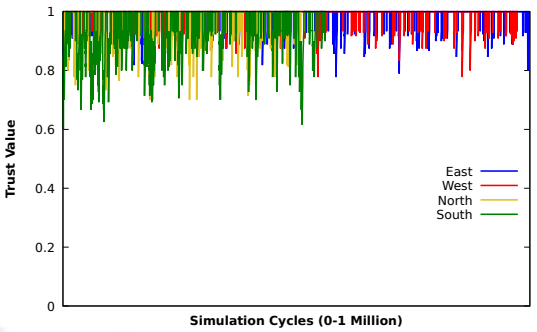


Fig. 4.15 Trust value variations of router 35 with its neighbours: any router with none of its neighbours affected by HT follows a more or less similar pattern.

shown in Figure 4.2. We take two routers (one with an HT-implanted router as a neighbour and the other with no HT neighbours), which are chosen randomly: routers 44 and 35, for the study of trust dynamics. The values are taken while running a uniform random traffic at an injection rate = 0.01. Except for the East neighbour of router 44, none of the other neighbours of routers 44 and 35 are HTs.

We plot the trust value variations of router 44 with its respective neighbours to the East, West, North, and South directions in Figure 4.14. Initially, the trust values in all directions start at 1. As time progresses, trust values with all neighbours change based on the analysis of re-transmitted packets. We know that router 45, the East neighbour of router 44, is an HT, in Figure 4.2. Eastward moving packets get corrupted, causing subsequent re-transmissions. This reduces trust with the neighbour to the East. At some point, the trust value gets close to 0.50. This reflects the presence of a risky path through East neighbour. As a result, packet traversal through router 45 (the East port of router 44) is stopped under opportunistic trust aware routing. The HT's impact on router 45 reduces the trust of router 44 with router 43. It is because the algorithm reduces the trust of all the routers in the path of a modified packet during its re-transmissions. Some of the westward packets from router 44 going through router 25 also suffer data corruption. This leads to trust values, with router 44's West neighbour falling low. Trust counters corresponding to an output port are updated only after that port is selected by the route computation unit. Since opportunistic routing does not use East and West as preferred choices, after a certain point, trust value updation also stops from West and East neighbours. We notice that router 44's trust value with its North neighbour is consistently high, and therefore, North is considered a better choice for forming a trustable path from router 44 to its North Eastern region. Although the trust with the South neighbour is not consistent, it often maintains a trust value close to 1. Accordingly, North and South

remain the most trusted adjacent neighbours from router 44 to reach the rest of the routers in the NoC.

Similarly, we plot the trust value variations of router 35 with its respective neighbours to the East, West, North, and South directions in Figure 4.15. The trust value of router 35 with all its neighbours except South is close to 1. Trust values with South are slightly lower than those of the other neighbours due to the presence of HT at router 11. As time progresses, we see that packet forwarding and trust updates significantly reduce with South port. All other neighbours of router 35 continue to benefit from a higher trust value, which is depicted by frequent trust value updates and opportunistic routing of packets through East, North, and West directions.

4.5.3 Sensitivity Analysis

In our work, we assume a threat model that uses HTs, which can corrupt packets. The count of HTs as well as the physical location of HTs impact the effectiveness of the proposed TROP technique. We vary the count of HTs, their physical location, and their orientation to analyse how sensitive these parameters are. We use uniform-random and bit-complement traffic patterns for our study. We present four cases based on the number of routers containing HT. Case A, case B, case C, and case D represent an NoC with only one HT router, two HT routers, three HT routers, and four HT routers, respectively. We randomly choose the positions of the HT routers. Figure 4.16 and Figure 4.17 show the comparison of the number of re-transmissions for the various architectures under study. The injection rate of the packets under each case is also marked on the X axis. For each case, we plot the number of re-transmissions at low load (injection rate =0.01) and higher load (injection rate =0.04). The purpose of this study is to understand the relationship between the number of HTs present in the NoC and the potential re-transmissions that are happening because of it. We observe that, irrespective of the load, across all the cases, TROP shows a lower number of packet re-transmissions than the other two. We observe that this reduction in packet re-transmissions is due to the successful exploration of trustable opportunistic paths using our approach.

Higher retransmissions in TAR show that the path used for routing packets has more HTs than the one used by TROP. Based on the dynamically updated trust values, TROP offers a more trustable path for packets. Results indicate that TROP estimates and updates trust more effectively so as to identify more trustable paths for packets. TROP reduces packet re-transmissions by 29% under single HT (Case A, low injection rate) and by 42% under 4 HTs (Case D, high injection rate) than TAR. This in turn reduces network traffic congestion due to the flow of fewer packets.

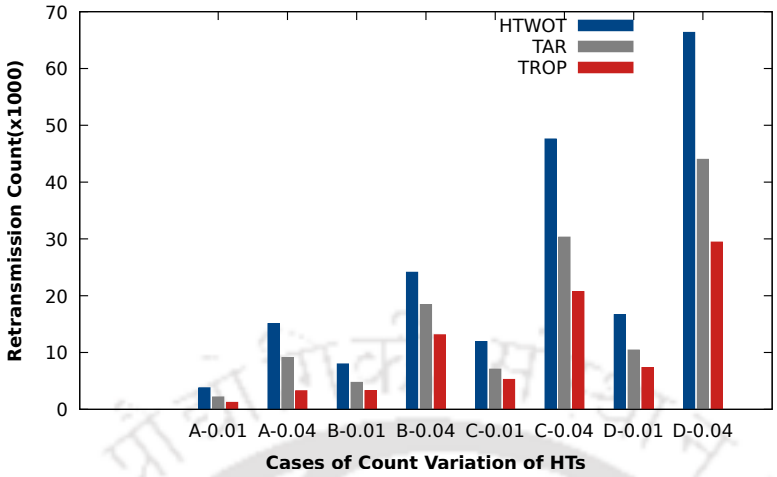


Fig. 4.16 Packet re-transmission count per million cycles for varying HT count at low and high injection rates (uniform-random)

We conduct all of our experiments with real workloads using case D, where 4 HTs are present in NoC. In order to demonstrate that our system is unaffected by the position of HTs in the NoC framework, we carry out a sensitivity study. In reference to the NoC layout illustrated in Figure 4.2, we identify and define five different scenarios for the physical placement of the HT location, as outlined in Table 4.4.

Table 4.4 Comparison study based on different physical orientation of HT location.

Case	HT Router Numbers	Remarks
D1	9, 10, 11, and 12	HTs at adjacent routers horizontally
D2	27, 35, 43, and 51	HTs at adjacent routers vertically
D3	0, 7, 57, and 63	HTs at corner routers
D4	42, 43, 50, and 51	One hop away HTs in the form of a square
D5	11, 22, 25, and 45	HTs at randomly chosen routers

The re-transmission count of packets against each case as per Table 4.4 is shown in Figure 4.18. Cases D1, D2, D4, and D5 show a more or less similar trend in the number of re-transmissions. Due to the dynamic packet routing based on trust values, the proposed TROP has the least number of re-transmissions in all five cases. D3 is the case where HT is present only in corner routers. This creates a scenario where the majority of packets flow without being impacted by HT. Moreover, because the HT is located in corner routers, only packets destined for those routers are impacted. Overall, the impact of case D3 is minimal

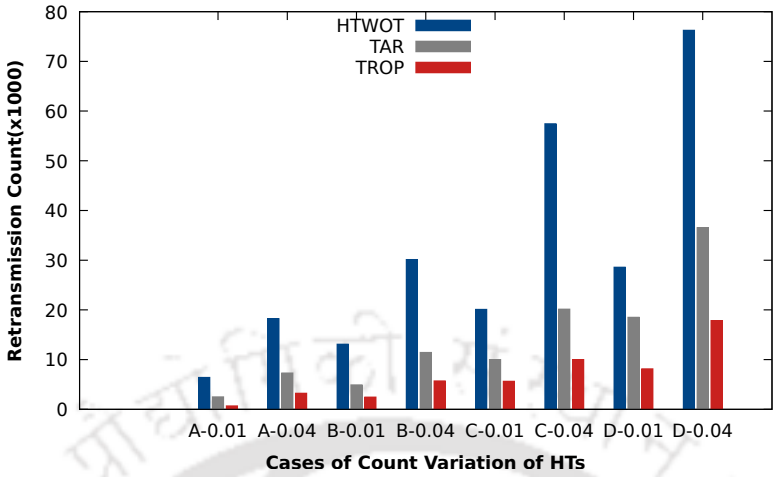


Fig. 4.17 Packet re-transmission count per million cycles for varying HT count at low and high injection rates (bit-complement)

and well visible in the number of packet re-transmissions. So we conclude that the position of HT is irrelevant to the effectiveness of the proposed TROP.

4.5.4 Overhead Analysis

TROP needs additional hardware for the implementation of ReTab and trust counters. Each entry in Retab takes 25 bits (SNID-8 bits, PID-8 bits, DB-2 bits, TC-6 bits, and RB-1 bit). As the upper limit of TC is set to 50 cycles, the maximum size of the ReTab is fixed accordingly. The trust counters need an additional 4 bytes. Altogether, a router with TROP needs an additional buffer space of 164 bytes, compared to 240 bytes in TAR. Additionally, the per-router memory requirement is 33% lower than that of TAR.

We design the TROP on the NoC router using Verilog HDL and synthesise using 90 nm technology in Cadence RTL Compiler [145]. We see that the TROP module incurs an additional 4.5% area and 6% power than the baseline router *(same as NoC Router in Table 2.5 of Chapter 2)* in NHT. Since the TROP circuitry does not operate in the critical path, we confirm that the NoC router with the suggested module meets all timing requirements related to the delay analysis compared to the router in NHT. A router with TROP can operate at the same frequency as a baseline router without TROP. Dynamic energy studies using McPAT with 65 nm technology show that TROP exhibits an energy savings of 12% over TAR due to a reduced number of re-transmissions and avoiding the traffic created by delegated-trust packets.

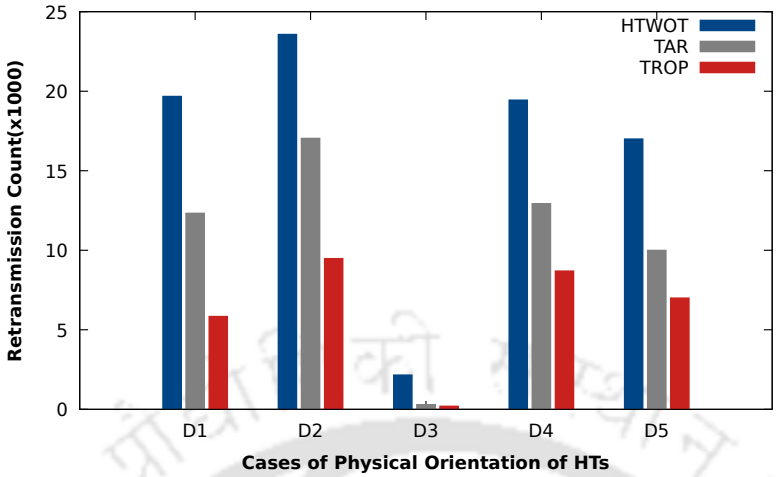


Fig. 4.18 Packet re-transmission count per million cycles for varying HT positions at injection rate 0.01

4.6 Chapter Summary

In this chapter, we presented a trust relationship between routers and their neighbours by exploiting the packet re-transmission pattern. When a packet's integrity check fails at the destination, the packet is discarded, and a request for re-transmission is sent to the source. A TRust-aware OPportunistic Routing (TROP) strategy is proposed in order to successfully avoid the effects of HTs on the path while ensuring that the packets reach their destinations unaltered. When packets are repeatedly re-transmitted, the router dynamically chooses a downstream router with a higher trust value. The experimental study demonstrated that TROP lowers the rate of re-transmissions by up to 40% for certain workloads with respect to existing mechanism [56]. With reasonable area and power overhead, TROP provided an opportunistic route for packets to reach their destination unaltered through the minimum HT-affected path.

Chapter 5

SARON: Secure Anonymous Routing in NoC

This chapter describes the third work of the thesis, proposing the source anonymous routing of packets in an NoC framework comprising HTs capable of performing flow-based traffic profiling. The number of communicating nodes in NoC is limited, and the usage of traditional routing algorithms following static paths can facilitates the application profiling by an HT. So we first propose a threat model that can break the anonymity through traffic profiling in the state-of-the-art anonymous frameworks in NoC. Secondly, we propose a method called SARON that can hinder unauthorised profiling and preserve anonymity during NoC packet transfer. When compared with state-of-the-art architecture, the proposed work significantly preserves anonymity with justifiable hardware overheads.

5.1 Introduction

In the fields MANET [146], WSN [147], and IoT [148], anonymous communication is becoming an increasingly important feature of network security services. The idea of anonymity shields the identities of the parties involved in the conversation. The identity of the sender (or source) is protected by the sender anonymity protocol [149]. Several recommendations for anonymity network architectures for internet communications, including Tor, Tarzan, BitTorent, NetCamo, I2P, Crowds, Freenet, and Mixmaster, are documented in the literature [150]. The sender anonymity of an IP core (or an associated NI or its local router) in an NoC framework is defined as the state of not being identified as the originator of a packet within the set of all cores of the TCMP. That set itself is called the sender anonymity set [151],[152]. The bigger the anonymity set, the more anonymous a sender's IP remains. Unlike having

millions of nodes communicating on the internet, the number of communicating nodes in NoC is very limited. So, it is challenging to preserve anonymity in NoC communications as there is a high probability of adversaries tracing back the communicating nodes. Hence, we need to explore other features specific to NoC architecture so that anonymity can be preserved. Once the packet is injected into the NoC framework that supports anonymous packet transfer, it should be impossible to deduce the exact source-destination (SD) pair of the packet. The SD pair of the packet is deducible through direct content retrieval from the corresponding packet's header fields or through traffic analysis. When we refer to the transmission of packets in an anonymous manner, we mean that the precise SD pair of the packet cannot be determined in the NoC framework either by direct content analysis or through traffic analysis. With the NoC framework as well, anonymous packet transmission makes it more difficult for malicious implants to do application profiling.

Lack of anonymity and lack of authenticated end-to-end encryption make the NoC more susceptible to targeted denial-of-service (DoS) attacks [153] or packet leakage. An attacker can clog some parts of the network with bogus traffic and cause DoS attacks. Knowing the source-destination ID of packets can allow HTs to execute targeted DoS attacks or leakage against packets communicated between certain processing cores. These types of attacks are initiated through passive communication from a colluding application (CA) [126] and an HT present in a router [52]. When packets between the targeted SD pair pass through the router containing HT, it can trigger specific attacks for packet leakage. Anonymous packet transmission by encrypting certain fields in the head flits are promising steps in this line to avoid targeted attacks on certain packets [52]. However, we show that head-flit encryption alone cannot secure packet profiling by malicious HTs. We propose a Secure Anonymous ROuting protocol for NoC, called SARON, that combines the effects of dynamic routing and an encrypted head flit to safeguard packets from HT-based traffic profiling. SARON does not change the basic NoC framework or the packet structure for its routing. We summarise the contributions of this chapter as follows:

a. We propose a threat model involving HTs at NoC routers capable of performing flow-based traffic profiling on header-encrypted packets to break anonymity.

b. We implement lightweight authenticated encryption that grants data confidentiality as well as SD pair anonymity.

c. We propose a secure anonymous routing protocol that explores route diversity between an SD pair and safeguards NoC packets from traffic profiling without changing the basic NoC structure and routing.

d. We prove that an effective combination of source ID encryption and the introduction of route diversity at source can reduce traffic profiling by HT.

e. We experimentally demonstrate the performance of SARON in terms of path diversity, source anonymity, and packet leakage rate.

5.2 Threat Model

We consider an NoC framework that facilitates SD pair anonymity by encrypting the source ID field in the packet header. Since the destination ID field is unencrypted for routing purposes, HT targets only identifying the source ID. The NoC follows XY routing. We assume that a compromised third-party NoC IP contains an HT that was undetected during the post-silicon tests. This HT incorporated into the NoC router can perform flow-based traffic profiling using a small additional circuit connected to its input ports. Since the HT can access the four ports (North, East, South, and West) of the router, profiling enables it to predict the SD pair of packets entering through these ports with up to 100% accuracy. This unauthorised profiling can eventually provide information about the nature and purpose of the applications running on the processing cores to an external attacker with the help of CA. The primary intention of this attack is to impose performance degradation attacks on NoC by targeting some cores, even in anonymous NoC frameworks. The CA running on a different core can activate the HT using a special message called an activation packet. Similarly, it can send a deactivation packet to make HT dormant. Once the activation packet is received, the HT initiates traffic profiling to identify the SD pair of incoming NoC packets, even though the source ID field is encrypted. Once the packet between a targeted SD pair is identified with its traffic profiling capability, the HT can perform malicious activities like delaying, dropping, modifying, re-routing, and leaking the packets. Since our HT is not a global adversary with complete access to network traffic, it can profile only those packets that are passing through the HT routers.

5.3 Traffic Profiling

In our context, traffic profiling by HT means identifying the SD pair of the packets in an anonymous NoC framework. Since the destination ID field is unencrypted, HT targets only identifying the source ID. The HT has the capability to predict the source ID of any packet by exploiting the features of the underlying routing algorithm. Whenever we discuss source ID prediction using profiling, we are referring to precisely deducing the packet's SD pair. XY

routing is the most popular routing algorithm followed in NoC. Here, packets flow in static, pre-determined paths from source to destination. The packets can enter a router through one of its five ports.

For instance, the HT is assumed to be located at router R. If a packet enters router R through its local input port, then the source ID can be predicted with 100% accuracy. If a packet enters R either through its East or West input ports, then the source of the packet and router R lie on the same row in an NoC with mesh topology. Since the number of routers in the same row is very limited, the prediction accuracy of the source ID by HT is high. However, if a packet enters the router R either through its North or South input ports, accurate prediction of the source ID of the packet is difficult because the possible number of routers from which this packet can originate and enter the North or South input port of R is high. The two approaches to facilitate HT's source prediction are as follows:

5.3.1 Profiling Based on Packets' Arrival Ports

A hotspot refers to a router position where HT can be implanted to profile packets such that the source ID of the packets can be predicted. Consider an $n \times n$ mesh NoC with router coordinates from $(0,0)$ to $(n-1,n-1)$. We define two columns in this NoC such that any routers in those columns can act as a potential hotspot with 100% source ID prediction accuracy for certain directions. These columns can be marked as column numbers 1 and $n-2$. The corresponding hotspots are $(1,y)$ and $(n-2,y)$, respectively, where y belongs to $[0, n-1]$. Consider Fig. 5.1 which illustrates these two categories of hotspots on a 4×4 mesh NoC. We have shaded two columns with locations $(1,y)$ and $(2,y)$ as hotspots. We can generalise this for larger meshes, and the following conclusions can be drawn:

1. If the hotspot is at $(1, y)$ where y belongs to $[0, n-1]$, it can predict the source ID of packets entering through its West input port with 100% accuracy as no packets other than those from the West neighbour enter its West input port while following XY routing.

2. If the hotspot is at $(n-2, y)$ where y belongs to $[0, n-1]$, it can predict the source ID of packets entering through its East input port with 100% accuracy as no packets other than those from the East neighbour enter its East input port while following XY routing.

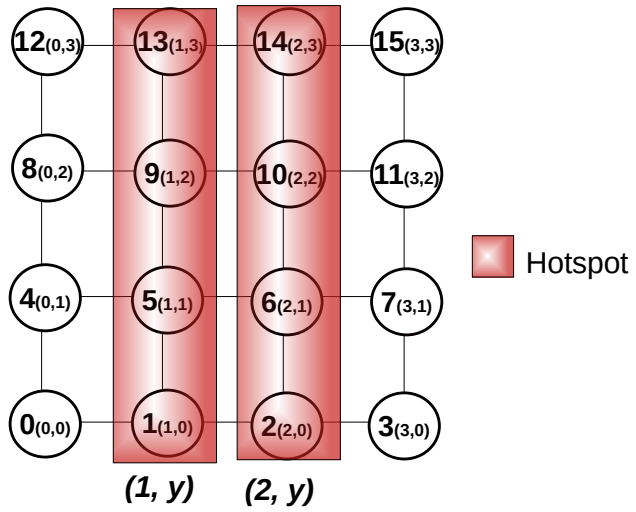


Fig. 5.1 4×4 NoC showing its hotspot location while using XY routing (n=4 and y belongs to $[0, n-1]$)

5.3.2 Profiling Based on Packets' Arrival Rate

Here we deal with another kind of profiling where the frequency of packet arrival and destination ID of such packets are used to predict the source ID of the packets. The source predictor logic of the proposed HT analyses the variations in the time delay (jitter) between packets and draws a conclusion about their source. An underlying clock and a counter can help measure it when it screens packets entering the router through a single port at a time. Consider a case where an application experiences a series of cache misses, which cause it to generate a series of packets in subsequent cycles destined for the same router. We call them similar packets because they travel between the same SD pair.

Let R be a hotspot through which these similar packets are travelling. With the help of analysing the jitter between the reception of such packets through a particular input port, R can predict whether these packets originate from neighbours or not. The packets originated from the neighbour of R do not experience congestion over multiple intermediate routers. However, packets originating from a non-neighbour of R experience switch arbitration and traffic congestion at multiple intermediate routers. This varying level of congestion creates an uneven delay between the reception of similar packets at R. Hence, the following conclusions about the source of the packets can be drawn:

1. If the HT in the router R experiences a uniform time delay between the reception of similar packets, then the probability that the packet originated from the neighbour of R is high. So, the HT can predict the source ID of the packet with high accuracy.

2. If the HT in the router R experiences a non-uniform time delay between the reception of similar packets, then the probability that the packet originated from the non-neighbour of R is high. This makes it difficult for HT to predict the source ID of the packet.

5.4 Motivation

Encrypting the packet's payload is the conventional way of securing NoC traffic. But the header of the packet has to remain in plain text form to facilitate routing operations at intermediate routers. Since the header of the packet contains the source ID field in unencrypted format, it is easy for traffic profiling. If the target is matched with the source ID, HT can initiate malicious activities on packets from the target core. The conventional way of hiding the source ID field in an NoC packet is to encrypt it so that intermediate routers cannot identify the origin of the packets. The HT model proposed above can perform traffic flow profiling and predict the source ID of NoC packets in source-anonymous communication. We implement the proposed HT in four random locations on an 8×8 mesh NoC [154], as shown in Fig. 5.2, to do traffic profiling on packets with encrypted source ID fields, as discussed in the previous section. Fig. 5.3 depicts the source ID prediction accuracy of the HT on packets entering it through its four ports. The X axis shows the four HT locations: router 25 [coordinate: (1,3)], router 12 [coordinate: (4,1)], router 46 [coordinate: (6,5)], and router 36 [coordinate: (4,4)]. The HT in router 25 can accurately predict the source ID of all packets entering through its West input port. Similarly, HT in router 46 can accurately predict the source ID of all packets entering through its East input port. We can see 100% accuracy against these two cases in Fig. 5.3. Since router 12 and router 36 are in the same column, the number of unique source IDs (four) for packets entering through their West input port is the same.Similarly, the number of unique source IDs (three) for packets entering through their East input port is also the same. This leads to their source ID prediction accuracy of 33% and 25%, for the West and East ports, respectively.

The above statistics show that the source ID of header-encrypted NoC packets can even be predicted with 100% accuracy by an HT capable of traffic profiling. Hence, it leads to source anonymity violations. Upon careful analysis, we observe that this is mainly because of the underlying XY routing, which is deterministic in nature, and also due to the limited number of source IDs possible for packets entering a particular port. The XY routing algorithm always provides a unique path between any pair of communicating nodes. Hence, traffic profiling is relatively easy for HTs. All packets from a given source always enter a router through the same input port. Hence, the count of unique source IDs mapped (enter) to the input port of a router is fixed. When this count is 1 (East port of 46 and West port of 25 in

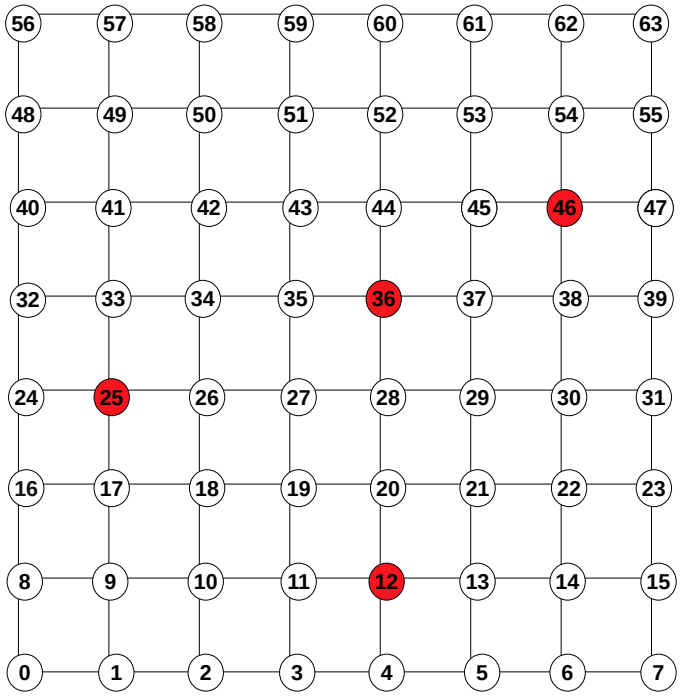


Fig. 5.2 An 8×8 mesh NoC with HT at four routers capable of traffic profiling.

Fig. 5.3), the proposed HT can predict the source ID with 100% accuracy, thereby breaking anonymity. The higher the count, the lower the prediction accuracy.

The source router set (SRS) of a port P, in an NoC router R, is defined as the set of all routers from which packets originate and enter router R through P [61]. When the routing is deterministic, the SRS of any port P is generally small, making it easy to predict the source ID of a packet by traffic profiling. The SRS of the West port of router 25 and the SRS of the East port of router 46 contain only one router. Hence, 100% source ID prediction is possible through traffic profiling. As the number of elements in SRS increases, it becomes difficult to figure out the origin of the packet. We are motivated by this fact and want to explore methods that can increase the cardinality of SRS against all ports of all routers in an NoC.

5.5 Related Works and Limitations

The concept of anonymous communication is an important field of privacy research from the early 1980s [155]. It is now relevant to on-chip communications as well. In the NoC framework, there are two classical approaches dealing with anonymous communication: ARNoC [51] and ARPP [52].The first approach, ARNoC, follows a two-phase mechanism in which the first phase handles route discovery, involving packet flooding and layered

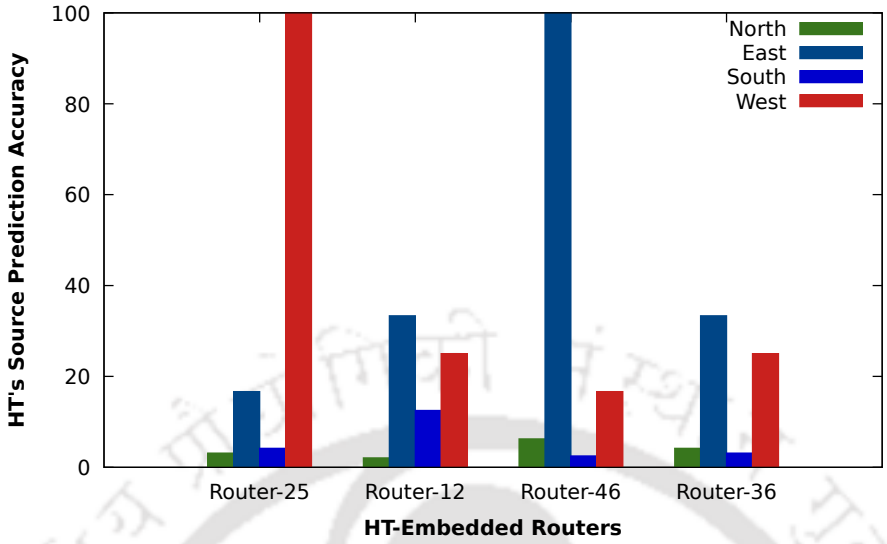


Fig. 5.3 Prediction accuracy of packet's encrypted source location by HT at four random locations in an 8x8 mesh NoC following XY routing in source anonymous mode.

encryption and decryption steps, while the second phase triggers data transmission. This flooding based mechanism increases congestion in the network and is a very costly approach on a resource-constrained platform like NoC. Our implementation of ARNoC shows that the flooding-based route discovery technique converges to a minimal path for data transmission. In certain instances, we observe that this minimal path is the same as that of the XY routing path. So, we are able to predict the source IDs with 100% accuracy using our traffic profiling. Although the nodes' identities are kept secret during communication through header encryption, the proposed traffic profiling reveals the source-destination pair. The second approach, ARPP (Anonymous Routing with a Pre-computed Path), encrypts the packet header but stores extra routing information into the packet, which is updated at every intermediate router. This method confuses adversaries by erratically choosing a routing path for each packet. It involves a session key exchange followed by data transmission with an encrypted header for anonymity. ARPP uses the XYX as the base routing algorithm with a pre-fixed count of X-hops and Y-hops in the packet header. Their routing offers more paths between a pair of source and destination than conventional XY routing. This increases the number of source IDs mapped to a given input port of a router. Our implementation of ARPP shows that for certain HT hotspots, we are able to achieve 100% source ID prediction. It also involves computation and updates at intermediate routers during packet routing, which also degrade NoC performance. Even though the above two approaches claim that they guarantee source anonymity, our proposed HT and its traffic profiling capability predict the source ID of packets correctly in certain cases. This happens mainly when the communicating

routers lie on the same row or column and also when they lie close to each other. Unlike these approaches, we concentrate on a source anonymity protocol that works well against the proposed threat model. It keeps the destination address field open in the packets so that packet forwarding at the intermediate routers becomes easier. We avoid costly operations like route discovery and key exchange before the actual packet transmission. We primarily employ path diversity in a conventional mesh-based NoC design, unlike network-coding-based NoC [156] that support path diversity with multi-cast traffic. The proposed method also integrates an end-to-end encryption method for NoC packets while concealing the exact identity of the source ID. Since the origin of the packet is not revealed and the source ID is not mapped to a specific input port, flow-based traffic profiling fails. Many research works [99], [46], [50], [55], [42], [108], [56], [42], show that NoC attack models are susceptible to performance degradation attacks through the proposed HT model. All these clearly show the necessity for developing a new source-anonymity protocol that can block HTs from both profiling the traffic and predicting the source.

5.6 SARON: Secure Anonymous Routing in NoC

We propose a Secure Anonymous ROuting protocol for NoC, called **SARON**, that combines the effects of dynamic routing and an encrypted source ID to safeguard packets from HT-based traffic profiling. The SARON provides not only path diversity between a pair of routers as a defence against profiling but also lightweight end-to-end data encryption as a defence against sniffers. Additionally, SARON adds short random delays to successive packets from a source to protect against jitter analysis. We believe that our work is the first to present an integrated solution applicable to the NoC platform that offers both anonymity and confidentiality against flow-based traffic profiling without altering basic NoC operations.

According to the threat model, HT targets packets from specified source IDs and can initiate attacks once the source ID and target are matched. In order to prevent traffic monitoring by HTs, packets from the same source ID intended for the same destination are transmitted over different paths chosen randomly using the proposed routing strategy. This breaks the pre-fixed mapping of source IDs to a router's input port. Hence, the HT fails to figure out the exact packet's source-destination pair. The SARON consists of two modules: a security module capable of performing encryption and decryption of packets at the network interface (NI) and an enhanced routing module at NoC routers. The detailed architecture of SARON is shown in Fig. 5.4.

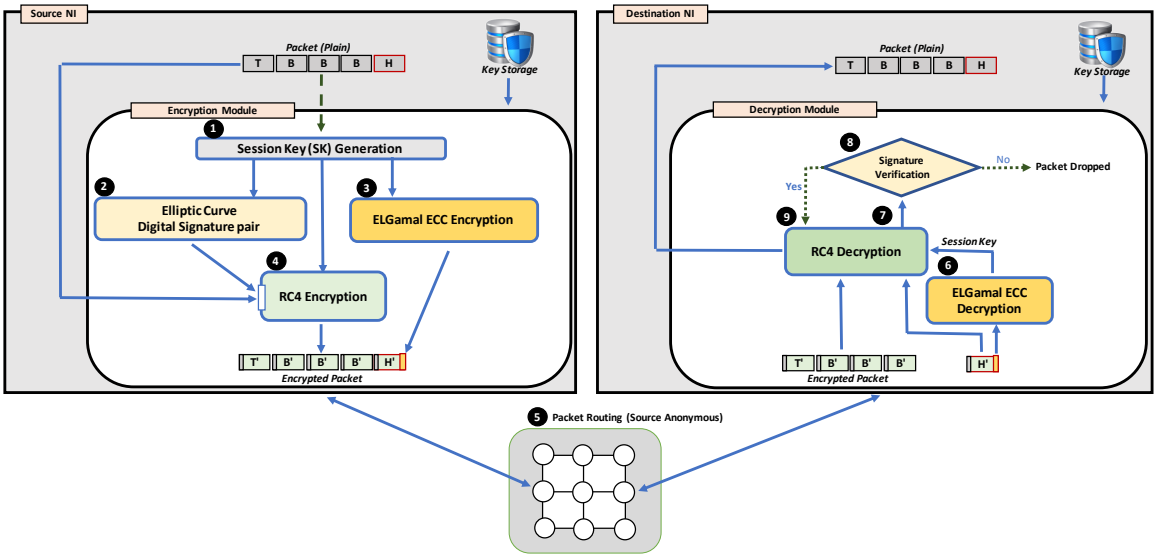


Fig. 5.4 Proposed Anonymous System in NoC Framework

5.6.1 Key Storage at Network Interface

We use a combination of Elliptic Curve Cryptography (ECC) and the RC4 encryption algorithm for the secure transmission of NoC packets. Towards this, the first step is choosing an elliptic curve that meets the security standards of an NoC framework. NIST curves [157],[158] like P-256, P-384, and P-521 are examples of some popular elliptic curves. Following that, the next stage is the key generation procedure. An elliptic curve over a finite field takes the following form:

$$y^2 = x^3 + ax + b(mod\ p) \tag{5.1}$$

where p is a prime number, and a and b are parameters. We use a 16-bit prime number, keeping in mind, the computational latency and memory constraints. Using the pre-fixed generator point (G) and the order (n) of the cyclic group, the key pair generation procedure for ECC is utilised to create both public and private keys for all NIs. A public key in ECC is a point (x,y) on the curve, and its corresponding private key is a scalar value.

We use a 32-bit public key and a 7-bit private key. Every point's coordinate is made up of 16 bits. The NI associated with each NoC router contains a small Key-Storage that is pre-loaded with the private key of the NI and the public keys of the remaining NIs in the TCMP. This one-time process is done in a secured environment during fabrication time. Additionally, Key-Storage also contains a set of fixed ECC domain parameters. The key storage space is accessed only at the NI level. As per our threat model, the HT is in the router,

none of the keys, including the public and private keys of any NI, are accessible to routers during communication.

5.6.2 Session Key Generation

Session Keys (SKs) are generated from the randomly picked two public keys (P and Q) selected from a pool of public keys available in the Key-Storage. We compute R ($R = P + Q$), the sum of P and Q under the elliptic curve addition. R is also a point on the same curve. We take the x and y coordinates of R, which are 16 bits each, and concatenate them to form a 32-bit SK. A new session key can be generated after every n cycles based on the perceived vulnerability of the system. When the NI needs to send a new packet, this session key is used as an input to the RC4 algorithm to encrypt the header and the payload (head and body flits) of the packet. This step is shown as ❶ in Fig. 5.4.

5.6.3 Elliptic Curve Digital Signature Generation

A digital signature ensures data integrity and confirms the identity of the signer (data origin authentication) by verifying the underlying data. The source NI that generates the packet is the signer, and the destination NI that verifies the signature is the verifier. The Elliptic Curve Digital Signature Algorithm (ECDSA) allows session keys to be signed exclusively by the source NI. Based on the order of the elliptic curve used, we use a 14-bit signature pair. For each packet, the 14-bit signature of the 32-bit current session key is generated using ECDSA. This step is shown as ❷ in Fig. 5.4.

5.6.4 ELGamal ECC Encryption

We have discussed the process of generating both the session key and its signature. For every packet starting from an NI, this session key and its signature have to be stored in the packet. We encrypt the session key by ELGamal ECC encryption using the public key (taken from the key storage) of the destination NI. The resultant key is called the ECC Encrypted Session Key (ESK). Since ELGamal generates a pair of points $((x_1, y_1), (x_2, y_2))$ as the cipher, the size of ESK is 64 bits. This ESK is appended to the head flit of the packet. This step is shown as ❸ in Fig. 5.4.

5.6.5 RC4 Encryption

To allow end-to-end encryption of the packet, the light-weight RC4 stream cipher is used. The secret key for RC4 is the 32-bit session key. RC4 is used to encrypt parts of the head flit, full body, and tail flits. Each flit is 128 bits. This step is shown as ❹ in Fig. 5.4.

We do ELGamal and RC4 encryption in parallel, as there is no dependency between them. The control packet (read/write request and coherence message) used in our NoC framework consists of a single head flit. The data packet (read/write reply and write back) consists of a head flit followed by four body flits. The proposed packet structure for 8×8 mesh NoC is shown in Fig. 5.5. The fields FT (Flit Type: 2 bits) and VCID (Virtual Channel ID: 3 bits) are additional control fields that are common prefixes kept parallel to the 128-bit flit channel to facilitate wormhole routing [159]. These common prefix fields are part of all the flits in a given packet. The head flit consists of the following standard fields: SID (Source ID: 6 bits), DID (Destination ID: 6 bits), Type (Packet Type: 2 bits), PID (Packet ID: 4 bits), and Block address (32 bits). We assume a maximum memory address space of 256GB. Since we use 64B cache blocks, the block address consists of 32 bits. Out of the 128- bits of the head flit, apart from the standard 50 bits discussed above, we add a 14-bit signature and a 64-bit ESK. We keep the DID and Type fields in plain text format. Other fields (56 bits) except ESK, have to be encrypted with RC4 using the session key. All the body flits and the tail flit of a packet are encrypted with RC4 using the same session key. From Fig. 5.5 we can see that some fields are unencrypted, a few other fields are encrypted using RC4, and one field is encrypted using Elgamal.

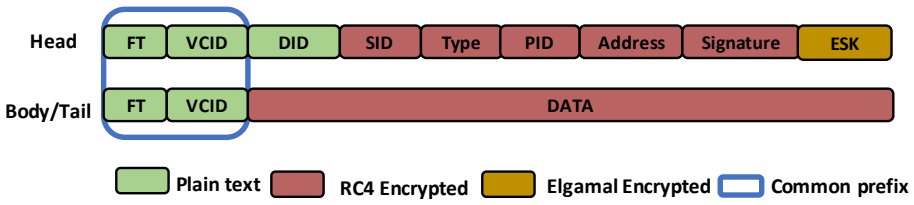


Fig. 5.5 Proposed packet format with a common prefix for all flits of a packet

5.6.6 Anonymous Packet Routing

We achieve anonymity by modifying the conventional DyXY routing [160]. In our routing mechanism, packets from the same source intended for a particular destination are transferred through different paths, as opposed to the conventional approach of transmitting them along a single path. This works well against adversaries that can only see local network traffic. To facilitate this, we make certain modifications to the conventional DyXY routing. We call this

routing mechanism CFS (Choose, Forward, and Start) routing. The CFS routing ensures that each packet follows a random path to its destination by providing several routes from source to destination. There are three steps in this routing. In the first step, the source router of the packet chooses a neighbour based on local congestion. In the second step, forward the packet to the chosen neighbour. In the third step, start routing the packet from the chosen neighbour to the destination using the DyXY routing mechanism. The DyXY routing algorithm takes the shortest route between the source and destination routers. It is adaptive in the sense that the downstream router for the packet is chosen based on the congestion conditions of the neighbouring routers in the direction of the destination.

A router can have four, three, or two neighbours in an 8×8 mesh NoC, depending on its position. If the source router has four neighbours and the destination is not in the same row or column as that of the source, two neighbours are identified as the minimal choices towards the destination, and the remaining two are identified as the non-minimal choices towards the destination. In the two-neighbour and three-neighbour cases, we also have the same kind of minimal and non-minimal choices. In step 1 of the CFS routing, we choose one of these possible neighbours based on the number of free virtual channels (VC) in the input buffer of the downstream router. The neighbour with the highest free VC count is chosen. A random choice is made when there is more than one neighbour with the same high free VC count. If source and destination are either in the same row or in the same column, then one of the neighbours other than the router, which is positioned opposite to the direction of destination, is chosen. The selection is also based on the free VC count, as previously stated.

In step 2, the packet is forwarded to the output port, leading to the neighbour chosen in step 1. The routing step, which generally operates based on the control fields of the head flit, is disabled in this step. At the end of this step, the packet reaches the neighbour of the source router. In step 3, the routing algorithm is enabled, the conventional DyXY routing algorithm is used, and the packet moves towards the destination. At every intermediate router, in order to control congestion and avoid deadlock, the selection of a downstream router is based on the free VC count. Since NoC operates on credit-based flow control, which is based on the availability of VC, no further circuitry is required to calculate the VC count. The selection of a downstream router from the current router can also be made at random out of the neighbours in the direction of the destination in DyXY (step 3) to explore all the paths, and to bring much randomness once we integrate a deadlock prevention method. This step is shown as ❺ in Fig. 5.4.

We summarise the above operations of CFS routing as follows: Consider a router R with its four (maximum) neighbours R_e, R_w, R_n, and R_s in the East, West, North, and South

directions, respectively. Router R has a packet P_1 to transmit to its destination. The three-step scheme is as follows:

Step-1: **Choose** a neighbour of R by following the checking of free VC count.

Step-2: **Forward** the packet P_1 to that neighbour.

Step-3: **Start** DyXY for P_1 from the neighbour.

To show the minimal and non-minimal choices of neighbours in step 1 of CFS routing, we use an illustrative example of a 4×4 mesh as shown in Fig. 5.7. In Fig. 5(a), a 4-neighbour case is depicted in which source router 5 has two minimal choices and two non-minimal choices towards the direction of destination router 11. In Fig. 5(b), a 3-neighbour case is depicted in which source router 8 has two minimal choices and one non-minimal choice towards the direction of destination router 2. In Fig. 5(c), a 2-neighbour case is depicted in which source router 0 has two minimal choices towards the direction of destination router 10. In Fig. 5(d), a 4-neighbour case is depicted for the source router 10 and destination router 2. Since these routers lie on the same column, the router (router 14) opposite to the direction of the destination is not considered a potential choice. Here, we call router 14 the blocked choice. In all four cases above, one of the choices is chosen as part of step 1 of CFS to forward the packet. The route computation unit of every router can operate in one of the two modes for a given packet: S-mode (SM) or NS-mode (NSM). If a packet is at its source router itself, then SM is activated at that router; otherwise, it is NSM. The route computation's NSM is identical to the regular DyXY routing algorithm. The conventional routing algorithm is disabled in SM. Step-1 and Step-2 of CFS routing work at the packet's source router (SM), while Step-3 of CFS routing works at the packet's non-source router (NSM). Fig. 5.6 depicts the modified router microarchitecture with SM and NSM operations during the route computation stage of each packet.

Illustrative Example: We describe the proposed CFS routing with the help of an 8×8 mesh NoC shown in Fig. 5.8. Consider a packet P_1 to reach its destination with the source as router 10 and the destination as router 45. Considering the position of the source and the destination, this is a 4-neighbour case, as discussed above. The router 9, router 11, router 2, and router 18 are the immediate neighbours of router 10. Out of these, two are minimal, and two are non-minimal choices. One of the four neighbours has to be selected as per the Choose step of CFS. Depending on whether the selected neighbour is a minimal or non-minimal choice, we describe the following two cases for packet P_1 to reach its destination:

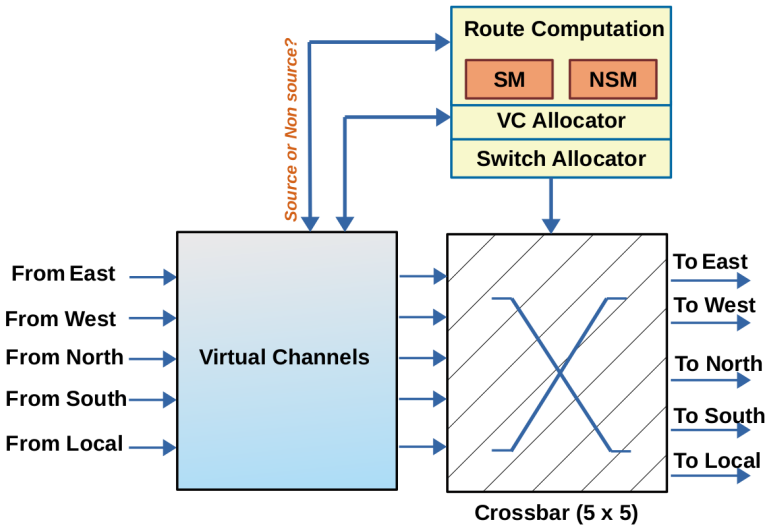


Fig. 5.6 NoC's router microarchitecture embedded with two modes (SM and NSM) of operation in the route computation stage.

Minimal Selection from Source

Here, router 10 selects either router 18 or router 11. Both belong to the minimal path in the direction of the destination. If router 18 is chosen based on the free VC availability condition from router 10, then the packet P_1 is forwarded to router 18. From now on, P_1 follows DyXY routing to reach its destination. This involves hoping through multiple intermediate routers. As per DyXY routing, P_1 at router 18 has two downstream router choices (router 26 and router 19) to proceed in the direction of the destination. One of these is selected based on the free VC availability condition. This way, P_1 eventually arrives at destination router 45. The path P_1 has taken is different from what it must have taken with conventional DyXY routing right from source router 10. The number of possible unique paths that P_1 can take using conventional DyXY routing is less than the number of possible unique paths that P_1 can take using our proposed CFS routing. Hence, the CFS routing reduces the source ID prediction accuracy of the proposed HT.

Non-Minimal Selection from Source

Here, router 10 selects either router 9 or router 2. Both belong to the non-minimal path in the direction of the destination. If router 9 is chosen based on the free VC availability condition from router 10, then the packet P_1 is forwarded to router 9. From now on, P_1 follows DyXY routing to reach its destination. As per DyXY routing, P_1 at router 9 has two downstream router choices (router 10 and router 17) to proceed in the direction of the destination. Since we adhere to the constraint that the input port and output port for any

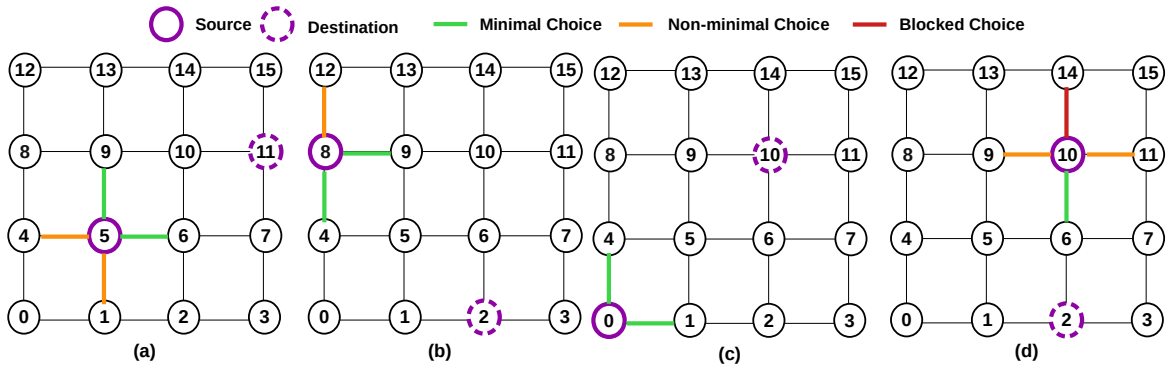


Fig. 5.7 NoC showing both minimal and non-minimal choices when the source and destination do not share the same row or column. **(a)** 4-neighbour case, **(b)** 3-neighbour case, **(c)** 2-neighbour case, **(d)** 4-neighbour case with a blocked choice.

packet cannot be the same at a router, the CFS chooses router 17 directly. At router 17, P_1 has two downstream router choices (router 18 and router 25). One of them is chosen based on the free VC availability. In this way, P_1 gets forwarded to various intermediate routers by choosing the best downstream router based on free VC availability until it reaches the destination.

Neighbour selection when source and destination share the same row or column

If the source and destination routers of any packet share the same row or column, the adjacent router that is located in the opposite direction of the shortest path to the destination is not considered for selection in step 1. Let the source and destination, for instance, be routers 25 and 29, respectively, as depicted in Fig. 5.8. The routers 24, 33, 26, and 17 are the immediate neighbours of 25. As discussed, one of these routers has to be chosen for the initial forwarding. However, router 24 is ignored as it is a blocked choice due to the restriction mentioned above. The blocked choice is a neighbour who is not eligible for selection (in step 1) when the source and destination occupy the same row or column.

Deadlock freedom: At every router, we implement the classical virtual channel bypassing technique [161] to connect packets from North and South input ports with the VC of the local input port. This will redirect packets from the Y direction to the local port before taking a turn in the X direction. From the local port, packets can take the X direction towards the destination like a newly injected packet. This eliminates the potential formation of deadlock that can occur due to the XY and YX turns adopted in our routing. Hence, CFS routing not only explores route diversity for packets, but also provides deadlock freedom.

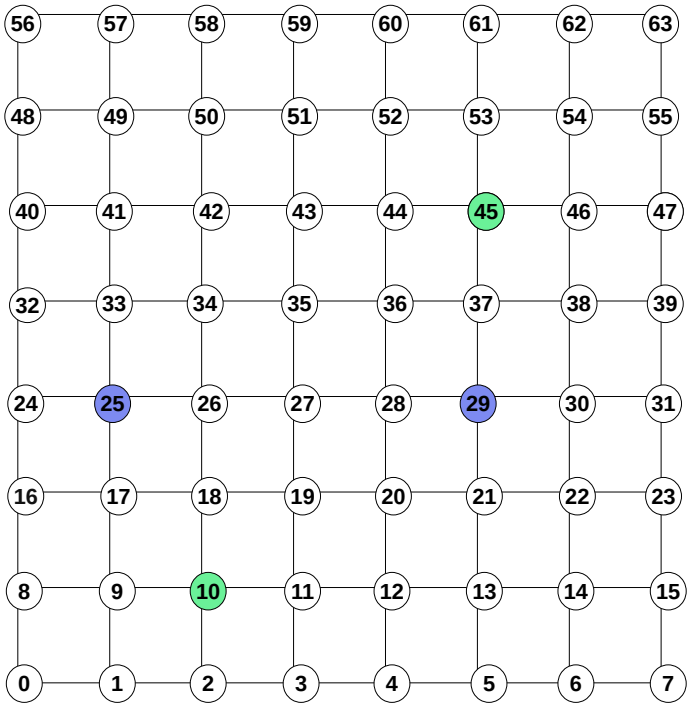


Fig. 5.8 An 8×8 mesh NoC to explore CFS routing.

5.6.7 Packet Processing at Destination NI

Once the packet reaches its corresponding destination NI, appropriate decryption techniques have to be initiated to extract control fields in the head flit and the payload in the body flits. Since the flits are encrypted using the session key, identifying the session key is the first step. As we know that the ESK is appended in the head flit at a pre-fixed position, we first extract the ESK as soon as the head flit reaches the destination NI. Since the ESK is obtained by encrypting the session key using the public key of the destination NI, we use the private key of the destination NI for decrypting the ESK to obtain the session key. This process is done by the ELGamal ECC decryption procedure (❻ in Fig. 5.4). With this session key, we extract the remaining fields of the head flit using RC4 decryption.(❼ in Fig. 5.4).

One of the fields in the head flit is the signature of the session key. To confirm that the session key's integrity has not been compromised and that the session key originated from an intended NI, the ECDSA signature verification module is used. However, if the signature present in the head flit does not match with the computed signature at destination NI, an integrity violation or authentication failure is flagged, and the relevant packet is dropped. Once the signature verification is successful(❽ in Fig. 5.4), RC4 decryption is used to decrypt body flits and tail flit (❾ in Fig. 5.4). In this way, our proposed technique ensures the secure movement of packets from source to destination. When the packet is dropped due

to a signature mismatch, re-transmission techniques are initiated as per the standard protocol [58].

5.6.8 Cryptosystem of SARON-An Overview

We primarily ensure both confidentiality and data origin authentication in the system. Confidentiality is achieved through the lightweight RC4 stream cipher. In addition, if stronger security is required, RC4 can be replaced with the ChaCha20-Poly1305 scheme [162]. However, this replacement is costly in terms of implementation and overhead in a resource-constrained framework like NoC. Given the significance of ECC for embedded security, session keys can be encrypted using an ECC-ELGamal approach. Our scheme also utilises ECDSA for signature generation and verification to authenticate the sender NI of the packet. We restrict the generation of signatures to session keys only. Signature generation for the whole packet is time-consuming as it involves encoding the packet to a point on the pre-defined elliptic curve. By considering the complexity of encoding, whole packet signature generation using ELGamal is avoided. However, other lightweight signature generation or message authentication code (MAC) using Sip-hash can be employed by adding an extra signature (or MAC) flit after the body flits. In this case, the signature flit has to act as a tail flit. Altogether, the proposed system is safe in its current form and is configurable to meet different needs based on the trade-off between overhead and security. SARON's cryptosystem is applicable to any NoC topologies like trees, ring, etc. without any modification. However, we need to select the appropriate topology specific adaptive routing algorithm to explore path diversity fully.

5.7 Anonymity Analysis of SARON

Here we attempt to prove that the proposed method offers greater source anonymity for packets. As discussed before, the HT makes use of probabilistic prediction of the source ID of packets by traffic monitoring. We consider two systems with an encrypted source ID field in the packet: a baseline system using XY routing and the proposed SARON. In Fig. 5.9, we use an illustrative example for a 4×4 mesh NoC to describe the SRS which we mentioned already. Here we study the baseline system and identify the potential SRS for three routers (router 10, router 2, and router 11) embedded with HTs. In Fig. 5.9, for instance, SR-N (or SRS-N) stands for the set of all probable source IDs (routers) of packets that can enter the specified HT router through the North (N) port. Similarly, SR-S, SR-W, and SR-E consist of potential source IDs with respect to the South, West, and East input ports, respectively. Since

Table 5.1 SRS members in an $n \times n$ mesh NoC with XY routing when the HT is at router location (p,q).

Ports	SRS members	Range (m)
N	$nm, nm+1, ...nm+(n-1)$	$m = q+1, q+2, ..n-1$
S	$nm, nm+1, ...nm+(n-1)$	$m = q-1, q-2, ..0$
W	$nm, nm+1, ...nm+(p-1)$	$m = q$
E	$nm+(p+1), nm+(p+2), ...nm+(n-1)$	$m = q$

the source ID field is encrypted in the packets, the only method to identify the source ID is through traffic analysis. If the cardinality of the SRS is high, source ID prediction accuracy is low. However, with XY routing, the size of the SRS corresponding to East and West ports is low. In Fig. 5.9 (a) and (b), the source ID of packets entering the HT router through its East port can be predicted with 100% accuracy, that is, router 11 and router 3, respectively. At the same time, in Fig. 5.9 (a) and (c), since the cardinality of SR-S is high, it is difficult for the HT to predict source ID correctly.

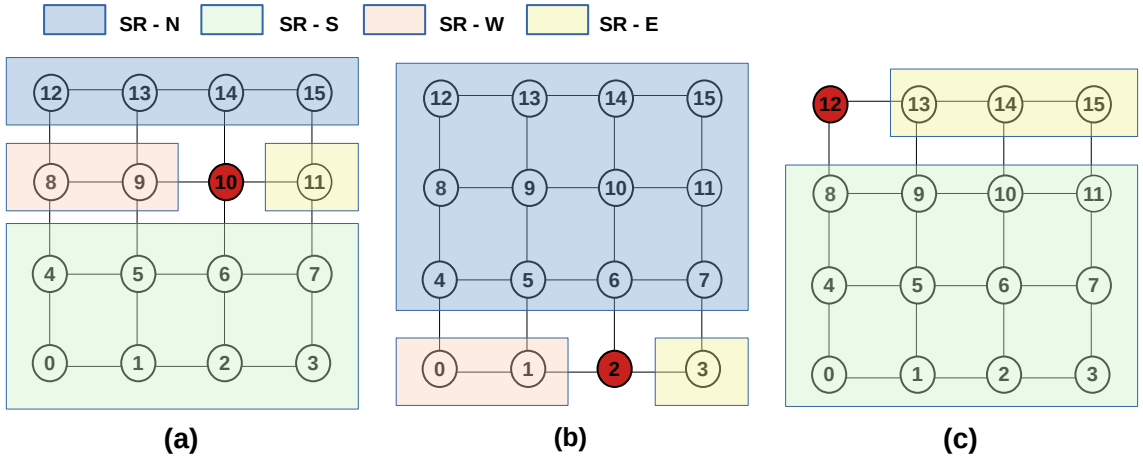


Fig. 5.9 Baseline: Dividing the routers of 4×4 mesh NoC (XY routing) into four SRS corresponding to an HT router: **(a)** HT at router 10, **(b)** HT at router 2 and **(c)** HT at router 12

Table 5.1 shows the possible members of SRS against each port in an $n \times n$ mesh NoC employing XY routing. The HT is assumed to be at location (p,q) in the mesh. Each value in the variable m corresponds to the routers in a row. For both East and West ports, m has just one possible value. This favours easy prediction of the source ID of a packet. Also, one more feature to favour traffic analysis is that every SRS is disjoint (non-overlapping) from every other set. The CFS routing of SARON aims at increasing the cardinality of all

SRS, especially the sets against East and West ports. It also ensures that the sets formed are overlapping, such that there are no longer disjoint sets with respect to ports. This is shown in Fig. 5.10 where the SRS of router 10 against all four ports is plotted. We can see that the SRS of router 10 with CFS routing is not disjoint. By comparing Fig. 5.9 (a) and Fig. 5.10 (a), it is clear that the cardinality of the SR-E of router 10 is increased from 1 to 7, and the SR-W is increased from 2 to 11.

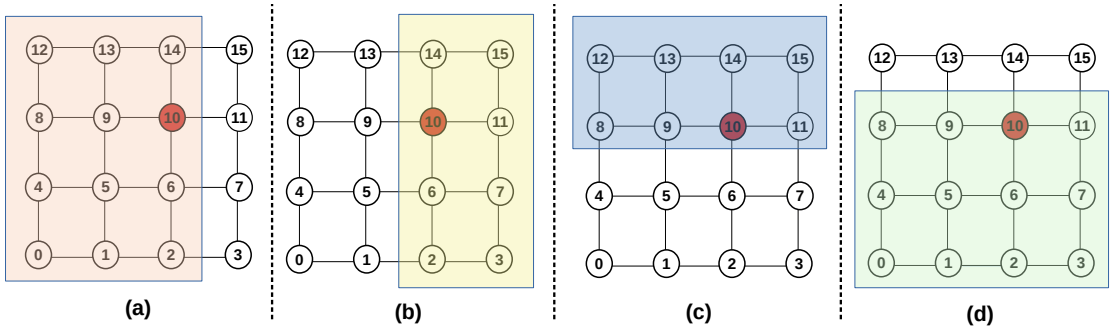


Fig. 5.10 SARON: Dividing the routers of 4×4 mesh NoC (CFS routing) into four SRS for HT router 10: **(a)** SR-W, **(b)** SR-E, **(c)** SR-N and **(d)** SR-S.

Table 5.2 shows the SRS members while using CFS routing. The range of values for the variable m against the E and W ports has now increased. The source ID prediction accuracy by HT at router 10 for XY, DyXY, and CFS routings is displayed in Table 5.3. We observe that the increase in SRS cardinality by CFS routing decreases the prediction probability of the HT. So, 100% source ID prediction is made impossible with SARON's CFS routing, and the packets can remain source anonymous.

Table 5.2 SRS members in an $n \times n$ mesh NoC with CFS routing when the HT is at router location (p,q).

Ports	SRS members	Range (m)
N	$nm, nm+1, ...nm+(n-1)$	$m = q, q+1, ..n-1$
S	$nm, nm+1, ...nm+(n-1)$	$m = q, q-1, ..0$
W	$nm, nm+1, ...nm+p$	$m = 0, 1, ..n-1$
E	$nm+p, nm+(p+1), ...nm+(n-1)$	$m = 0, 1, ..n-1$

In order to make it challenging for the profiling-capable HTs to infer the SD pair of a packet, we propose to have the following three desirable properties in the routing scheme, along with the encryption technique:

1. The cardinality of all SRS should be high enough to incorporate routers from more than one row. It should not be one.

Table 5.3 Source prediction probability in a 4×4 mesh NoC when the HT is at router 10 with location $(2,2)$.

Ports through which packet enters	XY(%)	DyXY(%)	CFS(%)
N	25	25	14.28
S	12.5	12.5	9.09
W	50	12.5	9.09
E	100	25	14.28

2. A packet should not always follow a unique path while communicating between any two tiles in NoC.

3. The SRS against any two ports need not be disjoint sets.

The above analysis shows that the XY routing fails to satisfy these three properties, making it vulnerable to easy SD pair prediction. The DyXY routing can provide path diversity between any pair of communicating nodes (or routers), unless they share the same row or column. If two communicating nodes belong to the same row or column, it behaves exactly like XY routing, thereby providing a single path. Therefore, in this case, the DyXY routing algorithm violates property-2. The XY routing path is a subset of the paths produced by DyXY routing. The CFS algorithm explores path diversity, resulting in packets from a router transmitting over different paths each time between communicating parties, irrespective of whether they share the same row or column. All three properties discussed above are satisfied by the CFS algorithm. As a result, CFS routing can ensure that traffic profiling is prevented. The HT's SD pair prediction accuracy of a packet entering through the given input port of a router is proportional to $(1/P)$, where P is the cardinality of the source router set. As the system scales, P increases for all the ports, thereby decreasing the prediction accuracy. Hence, we prove that SARON offers a better trade-off between performance and anonymity when compared with existing systems like ARNoC and ARPP.

5.8 Experimental Setup and Workloads

We model a 64-core TCMP system with an 8×8 mesh fixed-frequency NoC-based interconnect using the event-driven simulator, gem5 [118]. Other parameters used to configure the multi-core system are shown in Table 2.3 of Section *Evaluation Methodology* of Chapter 2. The proposed HT that performs traffic profiling is modelled using the on-chip interconnection module of gem5, called the garnet framework. The HT circuitry is mounted on the input

buffer of the NoC router. We implement SARON's cryptosystem in the NI of each tile to mitigate the effect of HT. We also implement the CFS routing technique in the route computation stage of the router pipeline.

To evaluate the effectiveness of the proposed SARON, we run a conventional uniform-random synthetic traffic pattern with diverse injection rates from zero load to saturation. We also analyse the performance of SARON using real workloads consisting of SPEC 2017 benchmarks. We make use of the SPEC 2017 benchmarks with workload details as specified in Table 2.4 of Section *Evaluation Methodology* of Chapter 2. Since WL1 consists of only high MPKI benchmarks, it generates the highest number of cache misses and hence the maximum NoC traffic. Similarly, WL2 generates the least number of cache misses and the minimum NoC traffic. The intensity of the NoC traffic load in the other three workloads depends on the percentage of high MPKI benchmarks present in the respective workloads. As a result, the systems' response in terms of packet latency to changing network loads can be analysed. We use a gem5 simulator and a Cadence RTL compiler to collect performance statistics and hardware overhead, respectively.

5.9 Experimental Analysis

We compare the performance of SARON with state-of-the-art anonymous techniques in NoC like ARNoC [51] and ARPP [52]. We explore the following four designs for our comparison and analysis:

a. **NAR:** An NoC system that follows XY routing and uses unencrypted packets without any HT in its routers.

b. **ARPP:** An NoC system affected by HT with the routing of both secure and non-secure packets using ARPP method [52].

c. **ARNoC:** An NoC system affected by HT that employs ARNoC for secure packet communication[51].

d. **SARON** An NoC system affected by HT that utilises the proposed secure routing for packet transmission.

We first study the analysis of CFS routing on various quantitative metrics, followed by a performance analysis of the proposed SARON technique in comparison with other designs mentioned above.

5.9.1 Analysis of Path Diversity of CFS Routing

In comparison to conventional routing algorithms like XY, OE (Odd-Even), and DyXY in mesh NoC, CFS routing offers more distinct paths between any pair of source and destination. So, it hinders adversaries from tracing back the origin of the packet by monitoring the entry port of an NoC router. When we use randomness to increase the number of paths in the Choose stage of CFS routing, the HT logic fails to identify the source ID. To find out the route diversity, we ran an experiment for 4 million cycles (injection rate = 0.1) and took the count of distinct paths that packets followed between 12 source-destination pairs, as shown in Table 5.4. The first column shows the source-destination pairs chosen for our study. The subsequent columns show the distinct paths possible for various routing algorithms. The source-destination pairs are selected by considering all possible directions and relative orientations. We can see that CFS routing offers the highest path (or route) diversity among other routing algorithms for every source-destination pair chosen for our study. In certain cases, it significantly outperforms DyXY routing. Even with adjacent routers, CFS routing can provide more than one path. In DyXY and CFS, both free VC count and randomness are considered for the selection of downstream routers. We do a deeper analysis to examine the path diversity between routers. We know that there are 70 distinct paths between 8 and 36 using CFS routing. Let these paths be numbered from T_1 to T_{70}. For instance, the packet P_1 can choose a random path T_1, P_2 can take another random path T_9, and so on. This selection of routes between routers 8 and 36 is depicted in Fig. 5.11. The 70 unique paths are shown on the Y axis. We can see that, as the simulation progresses (X axis), the packets from 8 to 36 choose one among the 70 paths at random, marked as a cross (x) in the plot. The plot clearly shows the diverse selection of routes and, hence, the effective load distribution between an SD pair.

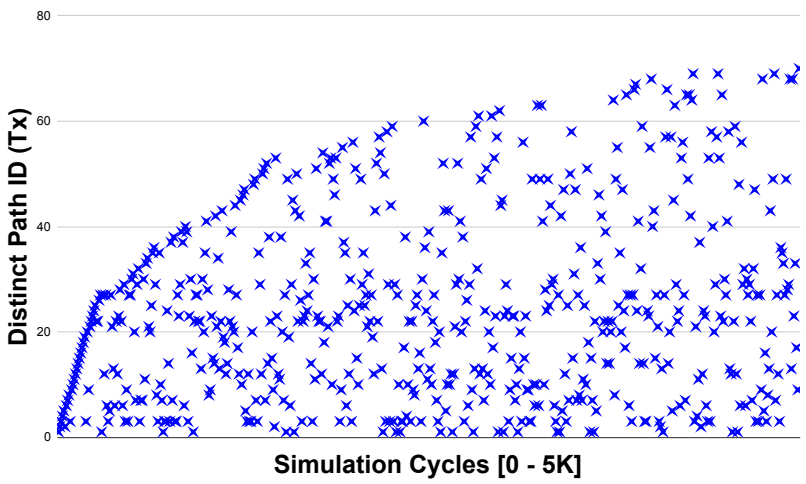


Fig. 5.11 Route selection plot between routers 8 and 36

Table 5.4 Comparison of the count of distinct paths from a source to destination in an 8×8 mesh NoC during simulation

(Source, Destination)	XY	OE	DyXY	CFS
(8,36)	1	10	35	70
(0,63)	1	321	1659	1670
(3,4)	1	1	1	2
(9,12)	1	1	1	7
(50,51)	1	1	1	3
(35,11)	1	1	1	7
(21,13)	1	1	1	3
(49,28)	1	4	20	50
(32,39)	1	1	1	15
(63,7)	1	1	1	8
(45,10)	1	5	35	91
(14,41)	1	35	126	334

5.9.2 Analysis of Hop Count and Packet Latency of CFS Routing

As stated earlier, the CFS routing forwards a packet from its source router to a one-hop away random neighbour (pseudo source) as the first stage of the routing procedure. The pseudo-source can lie on a minimal or non-minimal path. The selection is according to the free VC availability of potential downstream routers. This non-minimal pseudo-source selection causes more hops to be included in the path than the number of hops in the minimal path towards the destination. Hence, the average hop count between a source and destination using CFS routing will be higher than conventional DyXY routing. Fig. 5.12 shows the average hop count for any minimal routing (XY, OE and DyXY) and CFS routing under the uniform-random traffic pattern. We can see that the difference in hop count remains the same, irrespective of injection rate. In our approach, we consciously use a non-minimal strategy (one hop away from the source) as a way to increase the route diversity between source-destination pairs. A slight increase in overall packet latency is inevitable when selecting such a non-minimal neighbour from the source. Fig. 5.13 shows the comparison of routing algorithms in terms of average packet latency using a uniform-random traffic pattern in an 8×8 mesh network. We know that both hop count and congestion increase average packet latency. For lower injection rates, latency is attributed to hop count only. Here we can see that CFS routing creates an average latency hike of 3 to 4 cycles (more or less the same as the router's pipeline latency) due to an additional hop that may come as a result of the Choose step's non-minimal selection. As the injection rate increases, we can see that there

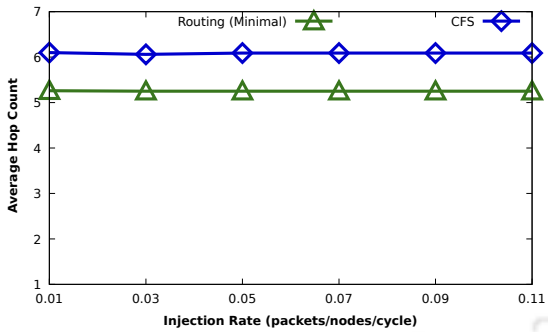


Fig. 5.12 Comparison of average hop count between minimal routing and CFS routing under uniform-random traffic.

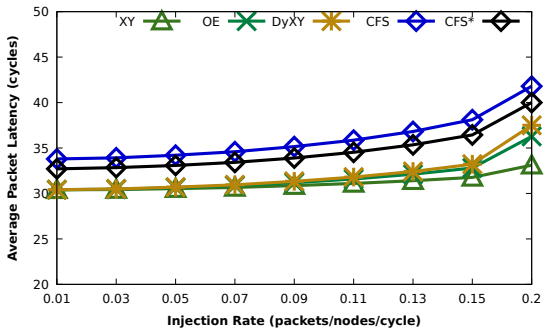


Fig. 5.13 Comparison of average packet latency for various routing algorithms under uniform-random traffic.

is an increase in latency, which is not only dependent on hop count but also impacted by the congestion level in the intermediate routers. For injection rates closer to saturation, the latency increases exponentially. Here, the impact of hop count is negligible.

We also implement a variation of CFS routing called CFS*, which is designed by applying a restriction in the choose stage. In CFS* routing, for a four-neighbour case, we eliminate one of two non-minimal neighbours such that a neighbour is chosen from the remaining three choices (2 minimal neighbours and one non-minimal neighbour) instead of four, which we see in CFS. This lowers the probability of reducing the number of non-minimal choices in the Choose step. The other cases, like three-neighbour and two-neighbour, remain the same as those of CFS routing. From Fig. 5.13, we can see that the latency with the CFS* routing is lower than CFS routing. Since CFS* routing gives a lesser diversity of routes than CFS routing, we cannot fully benefit from it in terms of source anonymity. However, CFS* routing is a better choice than CFS routing when additional latency incurred due to non-minimal paths causes performance degradation. Fig. 5.14 shows the average packet latency of routing algorithms with SPLASH-2 benchmark binaries (Barnes, Fmm, Ocean, Radiosity, and Raytrace) on a 16-core system. As discussed above, with CFS routing, an average latency hike of 3 to 4 cycles is added more when compared with the minimal routing algorithms.

5.9.3 Packet Latency of SARON

Every anonymous system in NoC employing end-to-end encryption measures packet latency to determine how well it performs in a resource-constrained setting. Average packet latency is a crucial quantitative parameter in NoC as it can impact the execution of applications running on the processing cores. Fig. 5.15 displays the average packet latency comparisons of various techniques under different workloads. We can see that, as expected, average

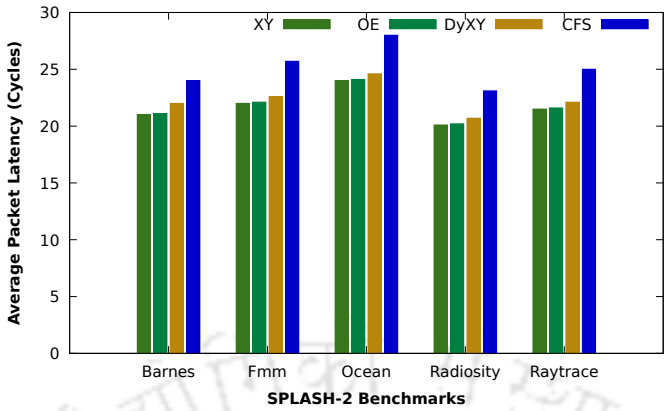


Fig. 5.14 Comparison of average packet latency using routing algorithms under SPLASH-2 benchmark programs.

latency values are highest for WL1 and lowest for WL2 due to the variation in NoC traffic created by the workloads. It is evident in the plot that integrating any form of security architecture into a baseline NoC incurs an unavoidable additional latency. We implement all the security modules of ARNoC, ARPP, and SARON in Verilog to compute the additional latency incurred at the source, destination, and intermediate routers. We then feed these values to gem5 for the computation of average packet latency. ARNoC, ARPP, and SARON show higher latency than NAR across all workloads. Cryptographic computations at the source and destination routers and additional processing at intermediate routers increase the latency of the ARPP technique. ARNoC contains cryptographic computations in all routers, right from source to destination. As discussed before, ARNoC has a route discovery phase using encrypted communications, followed by an encrypted data transfer phase. For a fair comparison with ARPP and SARON, we have implemented the route discovery phase of ARNoC for every 512 cycles and then followed the best path for actual data communication in encrypted format. We have used the amortised average latency overhead of the route discovery phase packets over the average packet latency of data packets.

Additional hop-count due to CFS routing, cryptographic computations at source and destination NIs, and random delays introduced between adjacent packets to prevent jitter analysis all account for the slightly higher average packet latency of SARON when compared to ARPP. When we study the average across all the workloads, SARON shows 3% higher average packet latency than ARPP, while ARNoC experiences 69% higher average packet latency than SARON. Hence, we prove that SARON offers the best trade-off between performance and anonymity.

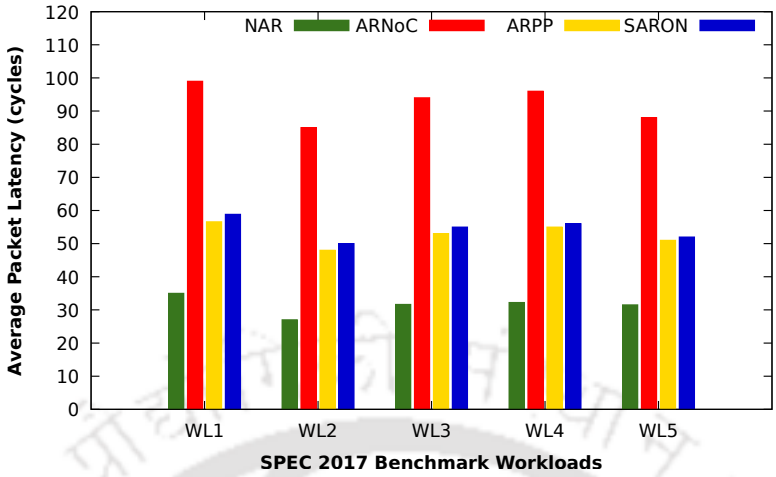


Fig. 5.15 Comparison of average packet latency under various workloads of SPEC 2017 benchmark suite.

5.9.4 Threat Mitigation with SARON

Packet leakage, one of the most prominent security attacks in NoC, enables external attackers to profile applications running on the system [84]. We study the extent of packet leakage by implementing it on a few random routers in an 8×8 mesh NoC. The CA, which acts on behalf of an attacker, sends commands about target routers (SD pairs) to the HT routers for leaking packets. When the HT router receives a packet matching the target routers, the trigger for the HT is enabled. The HT router duplicates the packet and changes the head flit's destination ID field so that the packet eventually reaches the CA. To match with the target routers and leak the packet, it has to break the anonymity first. If anonymity is broken, the SD pair of the packet is revealed. A comparison of the targeted packet leakage count (%) in various architectures under study for different workloads is shown in Fig. 5.16. Since NAR does not employ an encrypted source ID field, HT can have direct access to the source ID field, followed by packet leakage if there is a target match. On average, 16% of packets are leaked out from the HT routers across all the workloads in NAR. Due to the profiling capability of the proposed HT, it could predict the source ID of a few packets correctly in ARPP and ARNoC architectures. This results in leakage of 7.2% and 5.3% of packets, respectively. The SARON leaks only a negligible 0.6% packets across all workloads due to its CFS routing, which prevents the HT's traffic profiling capability. We observe that SARON reduces the leakage rate by 96%, 91% and 88% when compared with NAR, ARPP, and ARNoC, respectively. Since leaked packets are always modified packets, the integrity check of SARON using the lightweight Sip-hash technique at destination NI can block the leaked packets from reaching the CA. The leaked packets get discarded in the destination NI

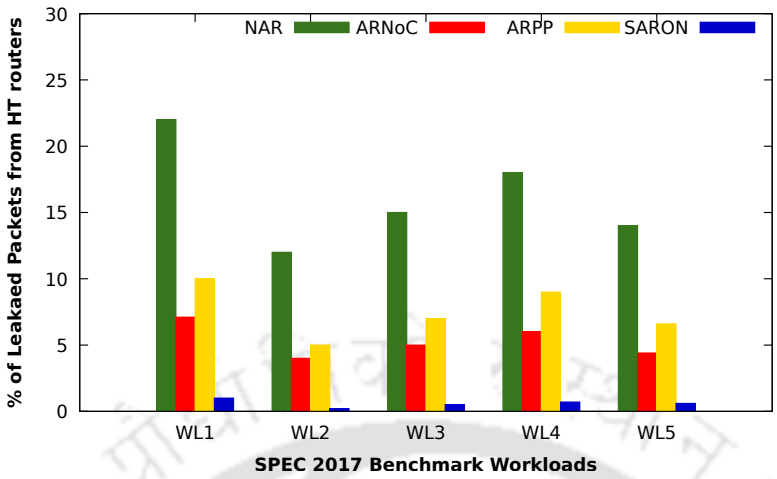


Fig. 5.16 Comparison of packet leakage (%) under various workloads of SPEC 2017 benchmark suite.

itself if there is a MAC mismatch. Thus, SARON does not allow an attacker to capture even a single packet.

5.9.5 Key Strength, Scalability and Throughput

As discussed, the maximum key size is taken to be 32 bits for ECC and RC4 encryption. The security architecture is configurable to enhance the key strength to 128 bits or more. The same architecture utilising high key strength can be employed when there is a need for strong security, regardless of the computational overhead associated with ECC operations. As the system's size increases, only the key-storage space at the NI increases linearly with the number of cores to incorporate that many public keys. However, no change is required in the CFS routing and packet format. So, the proposed method is scalable. The SARON does not inject any additional kind of packet into the NoC. The HT does not cause flooding, congestion, or packet loss in the network. Hence, there is no impact on throughput with respect to HT and the SARON.

5.9.6 Hardware Overhead and Circuit Latency

Since security operations take place at the NI level, key storage also has to be facilitated at NI. The key storage at NI consists of its private key, the public keys of all the NIs, and ECC parameters. A total of 263 bytes of key storage is required at each NI. We design the security module for encryption and CFS routing of SARON using verilog HDL and synthesise it using 90 nm technology in the Cadence RTL compiler. The proposed HT needs 2158.6 μm^2

of area and 68 μW of power. By considering the standard die area of the multi-core chip, the overhead of HT is very negligible, close to 0.001% of the total die area. The area and power overheads associated with SARON in comparison with the NAR *(same as Router+NI in Table 2.6 of Chapter 2)* are 7% and 22%, respectively. Since we employ cryptographic operations for each packet, this area and power overhead are justifiable. Our HDL synthesis shows that the encryption process of SARON at source NI takes 11 cycles, whereas decryption at destination NI takes 12 cycles. These additional overheads in latency are included in the computation of the average packet latency of SARON in Fig. 12.

5.10 Chapter Summary

The anonymous communication in NoC architecture implies that neither a direct content analysis nor traffic profiling can identify the SD pair of a packet. In this chapter, we demonstrated the applicability of secure anonymous transfer of NoC packets in an attempt to defeat the proposed HT that can perform traffic analysis on header-encrypted NoC packets. The proposed method, called SARON, ensured secure origin authenticated end-to-end encryption by combining ECC and lightweight RC4 encryption techniques. The path diversity between any two communicating routers is well explored through an adaptive, deadlock-free CFS routing. Additionally, we introduce a parameter named source router set, which illustrated the crucial significance of circumventing the HT's ability to monitor traffic while utilising CFS routing. With SARON, 96% of the targeted packet leakage was reduced. With negligible overheads, the SARON ensured greater anonymity than the existing state-of-the-art methods in NoC framework.

Chapter 6

Conclusions and Future Work

In this thesis, we have addressed three popular security problems in NoC, such as packet leakage, packet alteration, and the breaking of anonymity. A new lightweight security system is essential in NoC, as it can completely block any packet from reaching the colluding applications, as discussed in Section 4. The proposed method for the same is called Sec-NoC. In addition, we introduce a key exchange method to facilitate the implementation of symmetric ciphers at the network interface. The HTs' ability to perform these deeds can have an impact on the overall performance and credibility of TCMPs. The existing method of routing based on trust is not well suited to a resource-constrained NoC, and it has security vulnerabilities too, as discussed in Section 5. As a result, we propose a novel trust-aware routing technique, called TROP, in NoC. Additionally, the traffic profiling capability of HTs can break the anonymity of state-of-the-art anonymous systems in NoC, as discussed in Section 6. Hence, we propose a new anonymous system called SARON in NoC that can hinder the traffic profiling capability. Overall, we can enhance the security features of NoC's packet communication through the proposed three methods.

6.1 Thesis Summary

Fig. 6.1 summarises three contributions towards designing a secure NoC communication system against various HT attacks. The NoC, which serves as the communication backbone of modern TCMPs, is a popular spot for deploying HT and launching attacks. This is because of the involvement of untrusted third-party IP vendors. We look at three contributions that focus on secure NoC designs: The first (Sec-NoC) contribution proposes a lightweight security system targeted at blocking packet leakage. The second, (TROP), enables NoC packets to follow trusted paths, and the final, (SARON), integrates anonymity into packet

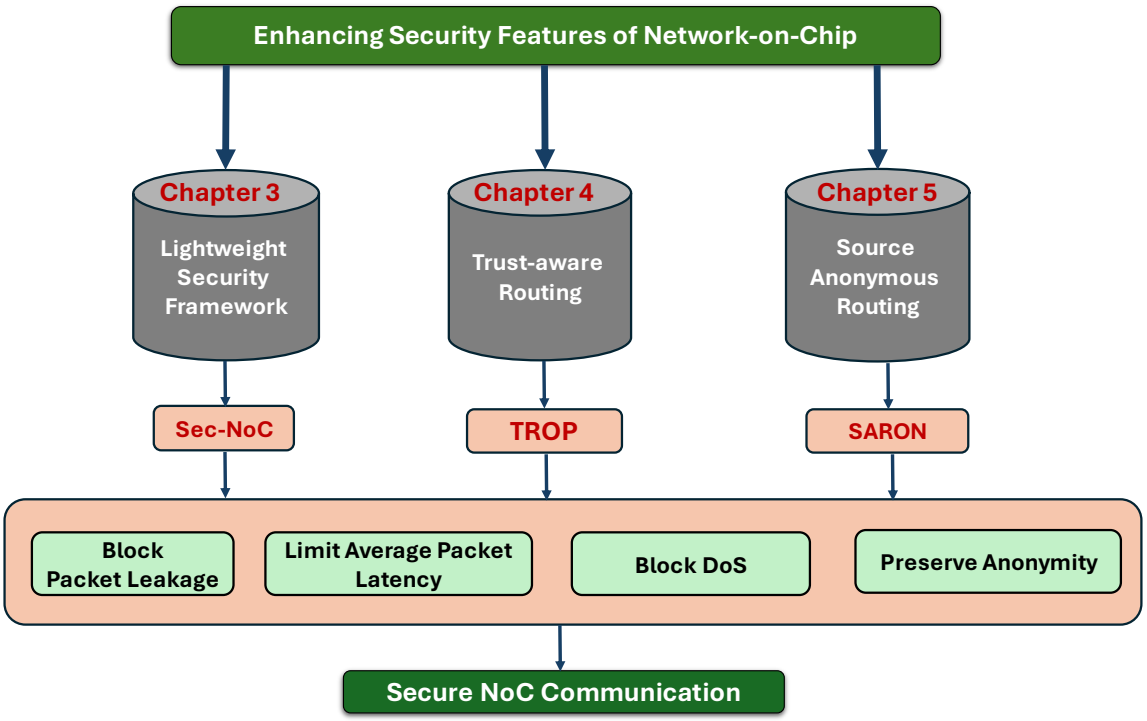


Fig. 6.1 Summary of contributions in the thesis.

transmission to block traffic profiling. Through these contributions, we can control packet leakage, DoS attacks, average packet latency, anonymity, in on-chip interconnects.

One of the most common vulnerabilities in the NoC is packet leakage, which lets attackers extract security keys and get important data from the associated SoC. The leak will have a greater impact and potentially pose a serious threat to national security if the compromised NoC is a component of military equipment. Therefore, it is critical to have lightweight security services as part of NoC communications. The Sec-NoC [57] does that. It assumes a threat model capable of both packet modification and leakage. The Sec-NoC's primary focus is on confidentiality and integrity in the NoC's communication. The lightweight cipher, Prince and a MAC producer, SipHash, are employed with Sec-NoC to ensure these targets. Sec-NoC proposes a secret key exchange method that follows key encapsulation. The metrics like recovery injection rate, transmission error rate, and effective packet utilisation ratio demonstrate the effectiveness of the Sec-NoC's ability to tolerate HT attacks. Sec-NoC outperforms AES-128, Fort-NoC, and P-Sec in average packet latency by 87%, 77%, and 73%, respectively. The Sec-NoC consumes 80% less area and 47% less power than the AES-based NoC. We have shown that around 28.25% of the packets leak from NoC, and the corresponding colluding application can make use of it when there are no security

implementations. With Sec-NoC's integrity check, not even a single packet reaches any malicious application.

An HT's integrity attack is primarily concerned with launching a DoS attack through silent data corruption. Error correction by re-transmission in NoC can lead to DoS attacks. Our research presents TROP [58], a reliable approach to trust-based packet routing that bypasses HT routers and ensures that packets arrive at their destination unaltered. The history of packet re-transmission patterns builds trust between neighbouring NoC routers in TROP, and it works locally in selecting the most trustworthy neighbour at every moment. Some of TROP's main goals are to create a fine-grained model for calculating trust, get rid of the idea of broadcasting trust packets, manage a re-transmission buffer well at the network interface level so that end-to-end re-transmission is possible, lower the number of re-transmissions by using HT bypassing, and conserve energy. While considering the re-transmission count in TROP, on average, there is a 57% reduction when compared with HTWOT and 32% on average with the state-of-the-art method TAR. Under heavy NoC traffic, TROP experiences around 9% less average packet latency than TAR. TROP exhibits an energy savings of 12% over TAR due to a reduced number of re-transmissions and avoiding the traffic created by delegated-trust packets. Although the method incurs negligible area overhead of 4.5% and power overhead of 6%, it provides an opportunistic route for packets to reach their destination unaltered through the minimum HT-affected path. This ensures that applications on the core run smoothly.

It is more difficult for attackers like HT to identify the IP cores (or routers) conversing when packets are sent anonymously. As a result, HT is unable to decode the properties of the applications running on the cores. In an anonymous NoC framework, profiling is defined as identifying the packet's source-destination pair. We propose a threat model with an HT capable of profiling that breaks the existing anonymous frameworks. As a countermeasure to this threat model, we present a technique known as SARON [61], [62], which enables the secure and anonymous transmission of NoC packets, thereby preventing unauthorised application profiling. The SARON comprises a cryptosystem and a new routing scheme called CFS routing. SARON's cryptosystem ensures both end-to-end data encryption and data-origin authentication. The CFS routing acts like an extended DyXY routing method in which the free VC availability parameter decides the next downstream neighbour. It is a deadlock-free routing algorithm. We designed the CFS routing to explore path diversity between any two communicating cores. Unlike the existing routing methods, the CFS explores multiple paths even when two communicating cores are only one hop away. Both the encrypted source ID field in the header and the path diversity play a role in escaping from HT profiling, and as a result, the source-destination pair of the packet becomes untraceable.

SARON has a slightly higher average packet latency than ARPP. This is because more hops are needed for CFS routing, cryptographic calculations at the source and destination NIs, and random delays put in between packets to stop jitter analysis. When we study the average packet latency across all the workloads, SARON shows 3% higher latency than ARPP, while ARNoC experiences 69% higher latency than SARON. Hence, we prove that SARON offers the best trade-off between performance and anonymity. We observe that SARON reduces the targted packet leakage rate in an anonymous system by 96%, 91% and 88% when compared with NAR, ARPP, and ARNoC, respectively. The area overhead associated with SARON in comparison with the NAR is 7%. This increase is reasonable because it involves cryptographic operations to ensure end-to-end data encryption and data-origin authentication.

6.2 Future Research Directions

The future directions include the following works.

- Sec-NoC: As we observed in Sec-NoC, NoC-based SoCs encrypt packets transferred between IP cores to defend against attacks like data leakage and confidentiality breaches. In order to minimise the time required for encrypting the payload of data packets, we wish to incorporate an incremental encryption (IE) approach exclusively for multimedia data. It uses the previously encrypted payload to encrypt current packet payloads. This is due to the unique characteristics of the multimedia data block, which primarily consists of pixel data residing at adjacent memory locations that have more or less similar values. It is likely that consecutive data packets from a core will either have identical data blocks or data with similar pixel values in some regions of the data block. The existing IE for NoC only targets the memory address (payload) of the adjacent control packets. When using conventional encryption techniques, the data packets take a longer time. In the future, we want to explore incremental encryption to significantly reduce data packet encryption time.

- Trust-Aware Routing: As already discussed, the computation of trust values through counter updates from the packets P_0, P_1, P_2....P_i is effectively utilised by the packet P_{i+1} in selecting its trusted, opportunistic neighbour in TROP. Since we ultimately want to explore trust-worthy paths free from HTs, incorporating some machine learning model in the selection of downstream routers gives more accurate HT-free paths. Recently, researchers have developed HT detection based on reinforcement learning (RL). This demonstrates that trust-aware routing can also utilise RL. When the source and destination routers are in the same row or column, TROP follows a single path

regardless of the trust values. This is due to the underlying DyXY routing algorithm. TROP can further be experimented with other adaptive routing methods to handle this scenario.

- Anonymous Routing: As far as NoC communication is concerned, the anonymity set is small compared to internet communications. We defined three properties exclusively to handle HTs capable of flow-based traffic profiling in a 2D mesh-based NoC. There is a need to study other kinds of topologies, such as trees or rings, in terms of preserving anonymity so that we can standardise the necessary and sufficient properties of NoC's routing in an anonymous framework. There is a possibility to explore the effect of collaborative attacks on anonymous communications as well.

References

[1] Abinash Roy, Jingye Xu, and Masud H. Chowdhury. Multi-core processors: A new way forward and challenges. In *Proceedings of IEEE International Conference on Microelectronics*, pages 454–457, 2008.

[2] Gordon E Moore. Cramming more components onto integrated circuits. *Proceedings of the IEEE*, 86(1):82–85, 1998.

[3] Stefanos Kaxiras and Margaret Martonosi. *Computer architecture techniques for power-efficiency*. Synthesis Lectures on Computer Architecture, Springer Nature, 2022.

[4] David W Wall. Limits of instruction-level parallelism. In *Proceedings of IEEE International Conference on Architecture Support for Programming Languages Operating System*, pages 176–188, 1991.

[5] Oyekunle Ayinde Olukotun, Lance Hammond, and James P Laudon. *Chip multiprocessor architecture: techniques to improve throughput and latency*, volume 3. Morgan & Claypool Publishers, 2007.

[6] James Balfour and William J. Dally. Design tradeoffs for tiled CMP on-chip networks. In *Proceedings of ACM International Conference on Supercomputing*, page 390–401, 2006.

[7] Semiconductor Industry Association. 2015 International Technology Roadmap for Semiconductors (ITRS). https://shorturl.at/pwzH7, 2015. [Online; Accessed 28-March-2024].

[8] W.J. Dally and B. Towles. Route packets, not wires: on-chip interconnection networks. In *Proceedings of IEEE Design Automation Conference*, pages 684–689, 2001.

[9] Yuan He, Hiroshi Sasaki, Shinobu Miwa, and Hiroshi Nakamura. Predict-more router: A low latency noc router with more route predictions. In *Proceedings of IEEE International Symposium on Parallel Distributed Processing, Workshops and Phd Forum*, pages 842–850, 2013.

[10] Jongman Kim, Dongkook Park, T. Theocharides, N. Vijaykrishnan, and C.R. Das. A low latency router supporting adaptivity for on-chip interconnects. In *Proceedings of IEEE Design Automation Conference*, pages 559–564, 2005.

[11] R. Mullins, A. West, and S. Moore. Low-latency virtual-channel routers for on-chip networks. In *Proceedings of IEEE International Symposium on Computer Architecture*, pages 188–197, 2004.

[12] Hangsheng Wang, Li-Shiuan Peh, and S. Malik. Power-driven design of router microarchitectures in on-chip networks. In *Proceedings of IEEE/ACM International Symposium on Microarchitecture*, pages 105–116, 2003.

[13] Luciano Bononi, Nicola Concer, Miltos Grammatikakis, Marcello Coppola, and Riccardo Locatelli. NoC Topologies Exploration based on Mapping and Simulation Models. In *Proceedings of IEEE Euromicro Conference on Digital System Design Architectures, Methods and Tools*, pages 543–546, 2007.

[14] Martha Mercaldi Kim, John D. Davis, Mark Oskin, and Todd Austin. Polymorphic on-chip networks. In *Proceedings of IEEE International Symposium on Computer Architecture*, pages 101–112, 2008.

[15] Daniel Sanchez, George Michelogiannakis, and Christos Kozyrakis. An analysis of on-chip interconnection networks for large-scale chip multiprocessors. *ACM Transactions on Architecture and Code Optimization*, 7(1), 2010.

[16] Subodha Charles and Prabhat Mishra. A survey of network-on-chip security attacks and countermeasures. *ACM Computing Surveys*, 54(5), 2021.

[17] Sebastian Tobuschat. *Predictable and runtime-adaptable network-on-chip for mixed-critical real-time systems*. Cuvillier Verlag, 2019.

[18] Arteris Technical report. Optimizing Enterprise-Class SSD Host Controller Design with Arteris FlexNoC Network-On-Chip Interconnect IP. https://shorturl.at/hwFV5, 2016. [Online; Accessed 28-March-2024].

[19] Jeyavijayan Rajendran, Vivekananda Vedula, and Ramesh Karri. Detecting malicious modifications of data in third-party intellectual property cores. In *Proceedings of IEEE/ACM Design Automation Conference*, pages 1–6, 2015.

[20] Onur Mutlu. The rowhammer problem and other issues we may face as memory becomes denser. In *Proceedings of IEEE Design, Automation & Test in Europe Conference & Exhibition*, pages 1116–1121, 2017.

[21] Nael Abu-Ghazaleh, Dmitry Ponomarev, and Dmitry Evtyushkin. How the spectre and meltdown hacks really worked. *IEEE Spectrum*, 56(3):42–49, 2019.

[22] Stephan Van Schaik, Marina Minkin, Andrew Kwong, Daniel Genkin, and Yuval Yarom. CacheOut: Leaking data on Intel CPUs via cache evictions. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 339–354, 2021.

[23] Jo Van Bulck, Daniel Moghimi, Michael Schwarz, Moritz Lippi, Marina Minkin, Daniel Genkin, Yuval Yarom, Berk Sunar, Daniel Gruss, and Frank Piessens. LVI: Hijacking transient execution through microarchitectural load value injection. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 54–72, 2020.

[24] Sujay Deb, Amlan Ganguly, Partha Pratim Pande, Benjamin Belzer, and Deukhyoun Heo. Wireless NoC as Interconnection Backbone for Multicore Chips: Promises and Challenges. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 2(2):228–239, 2012.

[25] Sudeep Pasricha and Nikil Dutt. ORB: An on-chip optical ring bus communication architecture for multi-processor systems-on-chip. In *Proceedings of Asia and South Pacific Design Automation Conference*, pages 789–794, 2008.

[26] Luka B. Daoud, M. El-Sayed Ragab, and Victor Goulart. Processor allocation algorithm based on Frame Combing with Memorization for 2D mesh CMPs. In *Proceedings of IEEE Latin American Symposium on Circuits and Systems*, pages 1–4, 2012.

[27] Aruna Jayasena, Binod Kumar, Subodha Charles, Hasini Witharana, and Prabhat Mishra. Network-on-chip trust validation using security assertions. *Journal of Hardware and Systems Security*, 6(3):79–94, 2022.

[28] Prabhat Mishra and Subodha Charles. *Trustworthy System-on-Chip Design Using Secure on-Chip Communication Architectures*, pages 3–30. Network-on-Chip Security and Privacy, Springer, 2021.

[29] Aashish Phansalkar, Ajay Joshi, Lieven Eeckhout, and Lizy Kurian John. Measuring program similarity: Experiments with SPEC CPU benchmark suites. In *Proceedings of IEEE International Symposium on Performance Analysis of Systems and Software*, pages 10–20, 2005.

[30] Mitali Sinha, Sidhartha Sankar Rout, and Sujay Deb. *DoS Attack Models and Mitigation Frameworks for NoC-Based SoCs*, pages 575–609. Springer, 2023.

[31] Yao Wang and G. Edward Suh. Efficient timing channel protection for on-chip networks. In *Proceedings of IEEE/ACM Sixth International Symposium on Networks-on-Chip*, pages 142–151, 2012.

[32] Martha Johanna Sepúlveda, Jean-Philippe Diguet, Marius Strum, and Guy Gogniat. NoC-Based Protection for SoC Time-Driven Attacks. *IEEE Embedded Systems Letters*, 7(1):7–10, 2015.

[33] Cezar Reinbrecht, Altamiro Susin, Lilian Bossuet, Georg Sigl, and Johanna Sepúlveda. Side channel attack on NoC-based MPSoCs are practical: NoC Prime+Probe attack. In *Proceedings of IEEE Symposium on Integrated Circuits and Systems Design*, pages 1–6, 2016.

[34] Cezar Reinbrecht, Bruno Forlin, and Johanna Sepúlveda. Cache timing attacks on NoC-based MPSoCs. *Microprocessors and Microsystems*, 66:1–9, 2019.

[35] Luka Daoud and Nader Rafla. Analysis of black hole router attack in network-on-chip. In *Proceedings of IEEE International Midwest Symposium on Circuits and Systems*, pages 69–72, 2019.

[36] Luka Daoud. Secure network-on-chip architectures for mpsoc: Overview and challenges. In *Proceedings of IEEE International Midwest Symposium on Circuits and Systems*, pages 542–543, 2018.

[37] Joël Porquet, Alain Greiner, and Christian Schwarz. NoC-MPU: A secure architecture for flexible co-hosting on shared memory MPSoCs. In *Proceedings of IEEE Design, Automation Test in Europe*, pages 1–4, 2011.

[38] Miltos D. Grammatikakis, Kyprianos Papadimitriou, Polydoros Petrakis, Antonis Papagrigoriou, George Kornaros, Ioannis Christoforakis, Othon Tomoutzoglou, George Tsamis, and Marcello Coppola. Security in MPSoCs: A NoC Firewall and an Evaluation Framework. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(8):1344–1357, 2015.

[39] Leandro Fiorin, Gianluca Palermo, and Cristina Silvano. A security monitoring service for NoCs. In *Proceedings of IEEE/ACM International Conference on Hardware/Software Codesign and System Synthesis*, page 197–202, 2008.

[40] Leandro Fiorin, Gianluca Palermo, Slobodan Lukovic, Valerio Catalano, and Cristina Silvano. Secure memory accesses on networks-on-chip. *IEEE Transactions on Computers*, 57(9):1216–1229, 2008.

[41] Adam Wiggins, Simon Winwood, Harvey Tuch, and Gernot Heiser. Legba: Fast hardware support for fine-grained protection. In Amos Omondi and Stanislav Sedukhin, editors, *Advances in Computer Systems Architecture*, pages 320–336. Springer, 2003.

[42] Johanna Sepulveda, Guy Gogniat, Daniel Flórez, Jean-Philippe Diguet, Cesar Zeferino, and Marius Strum. Elastic security zones for NoC-based 3D-MPSoCs. In *Proceedings of IEEE International Conference on Electronics, Circuits and Systems*, pages 506–509, 2014.

[43] Johanna Sepulveda, Daniel Flórez, Vincent Immler, Guy Gogniat, and Georg Sigl. Efficient security zones implementation through hierarchical group key management at NoC-based MPSoCs. *Microprocessors and Microsystems*, 50:164–174, 2017.

[44] Ramon Fernandes, César Marcon, Rodrigo Cataldo, Jarbas Silveira, Georg Sigl, and Johanna Sepúlveda. A security aware routing approach for NoC-based MPSoCs. In *Proceedings of IEEE Symposium on Integrated Circuits and Systems Design*, pages 1–6, 2016.

[45] Siavoosh Payandeh Azad, Michael Tempelmeier, Gert Jervan, and Johanna Sepúlveda. CAESAR-MPSoC: Dynamic and efficient mpsoc security zones. In *Proceedings of IEEE Computer Society Annual Symposium on VLSI*, pages 477–482, 2019.

[46] K. Sajeesh and Hemangee K. Kapoor. An authenticated encryption based security framework for noc architectures. In *Proceedings of IEEE International Symposium on Electronic System Design*, pages 134–139, 2011.

[47] Hansika Weerasena, Subodha Charles, and Prabhat Mishra. Lightweight encryption using chaffing and winnowing with all-or-nothing transform for network-on-chip architectures. In *Proceedings of IEEE International Symposium on Hardware Oriented Security and Trust*, pages 170–180, 2021.

[48] Kapoor Hemangee k, Bhoopal Rao G, Arshi Sharique, and Trivedi Gaurav. A Security Framework for NoC Using Authenticated Encryption and Session Keys. *Circuits, Systems, and Signal Processing*, 32:2605–2622, 2013.

[49] Julian Haase, Sebastian Jaster, Elke Franz, and Diana Göhringer. Secure communication protocol for network-on-chip with authenticated encryption and recovery mechanism. In *Proceedings of IEEE International Conference on Application-specific Systems, Architectures and Processors*, pages 156–160, 2022.

[50] Dean Michael Ancajas, Koushik Chakraborty, and Sanghamitra Roy. Fort-NoCs: Mitigating the threat of a compromised noc. In *Proceedings of ACM/IEEE Design Automation Conference*, pages 1–6, 2014.

[51] Subodha Charles, Megan Logan, and Prabhat Mishra. Lightweight anonymous routing in NoC based SoCs. In *Proceedings of IEEE Design, Automation & Test in Europe Conference & Exhibition*, pages 334–337, 2020.

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

[53] Ramon Fernandes, César Marcon, Rodrigo Cataldo, and Johanna Sepúlveda. Using smart routing for secure and dependable NoC-based MPSoCs. *IEEE/ACM Transactions on Networking*, 28(3):1158–1171, 2020.

[54] Johanna Sepúlveda, Andreas Zankl, Daniel Flórez, and Georg Sigl. Towards Protected MPSoC Communication for Information Protection against a Malicious NoC. *Procedia Computer Science*, 108:1103–1112, 2017.

[55] Travis Boraten and Avinash Karanth Kodi. Packet security with path sensitization for NoCs. In *Proceedings of IEEE Design, Automation & Test in Europe Conference & Exhibition*, pages 1136–1139, 2016.

[56] Subodha Charles and Prabhat Mishra. Lightweight and Trust-Aware Routing in NoC-Based SoCs. In *Proceedings of IEEE Computer Society Annual Symposium on VLSI*, pages 160–167, 2020.

[57] Syam Sankar, Ruchika Gupta, John Jose, and Sukumar Nandi. Sec-NoC: A Lightweight Secure Communication System for On-Chip Interconnects. *IEEE Embedded Systems Letters*, 16(2):214–217, 2023.

[58] Syam Sankar, Ruchika Gupta, John Jose, and Sukumar Nandi. TROP: TRust-aware OPportunistic Routing in NoC with Hardware Trojans. *ACM Transactions on Design Automation of Electronic Systems*, 29(2), 2024.

[59] Syam Sankar, Lissiyas Antony, Ruchika Gupta, John Jose, and Sukumar Nandi. Exploring trustable paths in network-on-chip for low-slack packets. In *Proceedings of International SoC Design Conference (ISOCC)*, pages 85–86, 2023.

[60] Arm Developer. CoreLink CI-700 . https://developer.arm.com/Processors/CoreLink%20CI-700, 2021. [Online; Accessed 23-March-2024].

[61] Syam Sankar, John Jose, Ruchika Gupta, and Sukumar Nandi. Enhancing Anonymity in NoC Communication to Counter Traffic Profiling by Hardware Trojans. In *Proceedings of IEEE International Symposium on Embedded Multicore/Many-core Systems-on-Chip*, pages 560–567, 2023.

[62] Syam Sankar, Ruchika Gupta, John Jose, and Sukumar Nandi. SARON: Secure Anonymous ROuting for NoC Communications. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,*, Under Minor Revision, 2024.

[63] Luca P Carloni. From latency-insensitive design to communication-based system-level design. *Proceedings of the IEEE*, 103(11):2133–2151, 2015.

[64] Davide Giri, Paolo Mantovani, and Luca P. Carloni. NoC-Based Support of Heterogeneous Cache-Coherence Models for Accelerators. In *Proceedings of IEEE/ACM International Symposium on Networks-on-Chip*, pages 1–8, 2018.

[65] Bill Dally. Enabling technology for on-chip interconnection networks. In *Proccedings of IEEE International Symposium on Networks-on-Chip*, pages 3–3, 2007.

[66] Luca Benini and Giovanni De Micheli. Networks on chips: A new SoC paradigm. *computer*, 35(1):70–78, 2002.

[67] Giovanni De Micheli and Luca Benini. Networks on chips: 15 years later. *Computer*, 50(5):10–11, 2017.

[68] Geoffrey Blake, Ronald G. Dreslinski, and Trevor Mudge. A survey of multicore processors. *IEEE Signal Processing Magazine*, 26(6):26–37, 2009.

[69] Avinash Sodani. Knights landing (KNL): 2nd Generation Intel® Xeon Phi processor. In *Proceedings of IEEE Hot Chips Symposium*, pages 1–24, 2015.

[70] William J Dally and Charles L Seitz. The torus routing chip. *Distributed computing*, 1:187–196, 1986.

[71] Natalie D Enright Jerger, Tushar Krishna, Li-Shiuan Peh, and Margaret Martonosi. *On-chip networks*, volume 12. Springer, 2017.

[72] William J Dally et al. Virtual-channel flow control. *IEEE Transactions on Parallel and Distributed systems*, 3(2):194–205, 1992.

[73] Yatin Hoskote, Sriram Vangal, Arvind Singh, Nitin Borkar, and Shekhar Borkar. A 5-GHz Mesh Interconnect for a Teraflops Processor. *IEEE Micro*, 27(5):51–61, 2007.

[74] Robert Mullins, Andrew West, and Simon Moore. Low-latency virtual-channel routers for on-chip networks. *ACM SIGARCH Computer Architecture News*, 32(2):188, 2004.

[75] Tushar Krishna. *Enabling dedicated single-cycle connections over a shared network-on-chip*. PhD thesis, Massachusetts Institute of Technology, 2014.

[76] Jose Duato, Sudhakar Yalamanchili, and Lionel Ni. *Interconnection networks*. Morgan Kaufmann, 2003.

[77] William James Dally and Brian Patrick Towles. *Principles and practices of interconnection networks*. Elsevier, 2004.

[78] Tiago D. Perez and Samuel Pagliarini. Hardware Trojan Insertion in Finalized Layouts: From Methodology to a Silicon Demonstration. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 42(7):2094–2107, 2023.

[79] Rajat Subhra Chakraborty, Seetharam Narasimhan, and Swarup Bhunia. Hardware trojan: Threats and emerging solutions. In *Proceedings of IEEE International High Level Design Validation and Test Workshop*, pages 166–171, 2009.

[80] Mohammad Tehranipoor, Hassan Salmani, Xuehui Zhang, Michel Wang, Ramesh Karri, Jeyavijayan Rajendran, and Kurt Rosenfeld. Trustworthy Hardware: Trojan Detection and Design-for-Trust Challenges. *Computer*, 44(7):66–74, 2011.

[81] Georg T. Becker, Francesco Regazzoni, Christof Paar, and Wayne P. Burleson. Stealthy dopant-level hardware trojans. In Guido Bertoni and Jean-Sébastien Coron, editors, *Cryptographic Hardware and Embedded Systems*, pages 197–214. Springer, 2013.

[82] Kaiyuan Yang, Matthew Hicks, Qing Dong, Todd Austin, and Dennis Sylvester. A2: Analog malicious hardware. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 18–37, 2016.

[83] Lang Lin, Markus Kasper, Tim Güneysu, Christof Paar, and Wayne Burleson. Trojan side-channels: Lightweight hardware trojans through side-channel engineering. In *Cryptographic Hardware and Embedded Systems*, pages 382–395. Springer, 2009.

[84] Yuriy Shiyanovskii, F Wolff, Aravind Rajendran, C Papachristou, D Weyer, and W Clay. Process reliability based trojans through NBTI and HCI effects. In *Proceedings of IEEE Conference on Adaptive Hardware and Systems*, pages 215–222, 2010.

[85] Timothy Trippel, Kang G. Shin, Kevin B. Bush, and Matthew Hicks. ICAS: an Extensible Framework for Estimating the Susceptibility of IC Layouts to Additive Trojans. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 1742–1759, 2020.

[86] Mohammad Tehranipoor and Farinaz Koushanfar. A survey of hardware trojan taxonomy and detection. *IEEE Design Test of Computers*, 27(1):10–25, 2010.

[87] Ramesh Karri, Jeyavijayan Rajendran, Kurt Rosenfeld, and Mohammad Tehranipoor. Trustworthy Hardware: Identifying and Classifying Hardware Trojans. *Computer*, 43 (10):39–46, 2010.

[88] Sally Adee. The hunt for the kill switch. *IEEE Spectrum*, 45(5):34–39, 2008.

[89] David E. Sanger and Thom Shanker. N.S.A. Devises Radio Pathway Into Computers. https://shorturl.at/fsAJ2, 2014. [Online; Accessed 28-January-2024].

[90] Jordan Robertson and Michael Riley. The Big Hack: How China Used a Tiny Chip to Infiltrate U.S. Companies. https://shorturl.at/dqG69, 2018. [Online; Accessed 20-January-2024].

[91] Virinchi Roy Surabhi, Prashanth Krishnamurthy, Hussam Amrouch, Kanad Basu, Jörg Henkel, Ramesh Karri, and Farshad Khorrami. Hardware trojan detection using controlled circuit aging. *IEEE Access*, 8:77415–77434, 2020.

[92] Zhixin Pan and Prabhat Mishra. Automated test generation for hardware trojan detection using reinforcement learning. In *Proceedings of Asia and South Pacific Design Automation Conference*, pages 408–413, 2021.

[93] Vasudev Gohil, Satwik Patnaik, Hao Guo, Dileep Kalathil, and Jeyavijayan (JV) Rajendran. DETERRENT: detecting trojans using reinforcement learning. In *Proceedings of ACM/IEEE Design Automation Conference*, page 697–702, 2022.

[94] Vasudev Gohil, Hao Guo, Satwik Patnaik, and Jeyavijayan Rajendran. ATTRITION: Attacking Static Hardware Trojan Detection Techniques Using Reinforcement Learning. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, page 1275–1289, 2022.

[95] Rajesh JS, Koushik Chakraborty, and Sanghamitra Roy. *Hardware Trojan Attacks in SoC and NoC*, pages 55–74. Springer, 2018.

[96] Venkata Yaswanth Raparti and Sudeep Pasricha. Lightweight mitigation of hardware trojan attacks in noc-based manycore computing. In *Proceedings of ACM/IEEE Design Automation Conference*, pages 1–6, 2019.

[97] Bruno Oliveira, Rafael Reusch, Henrique Medina, and Fernando Moraes. Evaluating the cost to cipher the NoC communication. In *Proceedings of IEEE Latin American Symposium on Circuits Systems*, pages 1–4, 2018.

[98] Michel A. Kinsy, Shreeya Khadka, Mihailo Isakov, and Anam Farrukh. Hermes: Secure heterogeneous multicore architecture design. In *Proceedings of IEEE International Symposium on Hardware Oriented Security and Trust*, pages 14–20, 2017.

[99] C.H. Gebotys and R.J. Gebotys. A framework for security on NoC technologies. In *Proceedings of IEEE Computer Society Annual Symposium on VLSI*, pages 113–117, 2003.

[100] Manoj Kumar J.Y.V., Ayas Kanta Swain, Sudeendra Kumar, Sauvagya Ranjan Sahoo, and Kamalakanta Mahapatra. Run time mitigation of performance degradation hardware trojan attacks in network on chip. In *Proceedings of IEEE Computer Society Annual Symposium on VLSI*, pages 738–743, 2018.

[101] Julian Haase, Sebastian Jaster, Elke Franz, and Diana Göhringer. Secure communication protocol for network-on-chip with authenticated encryption and mechanism. In *Proceedings of IEEE International Conference on Application-specific Systems, Architectures and Processors*, pages 156–160, 2022.

[102] Sergei Skorobogatov. *Physical Attacks and Tamper Resistance*, pages 143–173. Springer, 2012.

[103] Elena Dubrova, Mats Näslund, and Göran Selander. Secure and efficient lbist for feedback shift register-based cryptographic systems. In *Proceedings of IEEE European Test Symposium*, pages 1–6, 2014.

[104] Mihir Bellare, Oded Goldreich, and Shafi Goldwasser. Incremental cryptography: The case of hashing and signing. In Yvo G. Desmedt, editor, *Advances in Cryptology*, pages 216–233. Springer, 1994.

[105] Mubashir Hussain, Amin Malekpour, Hui Guo, and Sri Parameswaran. EETD: An Energy Efficient Design for Runtime Hardware Trojan Detection in Untrusted Network-on-Chip. In *Proceedings of IEEE Computer Society Annual Symposium on VLSI*, pages 345–350, 2018.

[106] Amey Kulkarni, Youngok Pino, and Tinoosh Mohsenin. Svm-based real-time hardware trojan detection for many-core platform. In *Proceedings of International Symposium on Quality Electronic Design*, pages 362–367, 2016.

[107] Ke Wang, Hao Zheng, and Ahmed Louri. TSA-NoC: Learning-Based Threat Detection and Mitigation for Secure Network-on-Chip Architecture. *IEEE Micro*, 40(5):56–63, 2020.

[108] Manoj Kumar J.Y.V., Ayas Kanta Swain, Sudeendra Kumar, Sauvagya Ranjan Sahoo, and Kamalakanta Mahapatra. Run time mitigation of performance degradation hardware trojan attacks in network on chip. In *Proceedings of IEEE Computer Society Annual Symposium on VLSI*, pages 738–743, 2018.

[109] Sadia Moriam, Elke Franz, Paul Walther, Akash Kumar, Thorsten Strufe, and Gerhard Fettweis. Protecting communication in many-core systems against active attackers. In *Proceedings of ACM Great Lakes Symposium on VLSI*, page 45–50, 2018.

[110] Qiaoyan Yu and Jonathan Frey. Exploiting error control approaches for hardware trojans on network-on-chip links. In *Proceedings of IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems*, pages 266–271, 2013.

[111] Qiang Liu, Wenqing Ji, Qi Chen, and Terrence Mak. IP Protection of Mesh NoCs Using Square Spiral Routing. *IEEE Transactions on Very Large Scale Integration Systems*, 24(4):1560–1573, 2016.

[112] Prasad Nagabhushanamgari, Vikash Sehwag, Indrajit Chakrabarti, and Santanu Chattopadhyay. Embedding delay-based physical unclonable functions in networks-on-chip. *IET Circuits, Devices & Systems*, 15(1):27–41, 2021.

[113] Jonathan Frey and Qiaoyan Yu. A hardened network-on-chip design using runtime hardware trojan mitigation methods. *Integration*, 56:15–31, 2017.

[114] Mohammad Humam Khan, Ruchika Gupta, Vedika J. Kulkarni, John Jose, and Sukumar Nandi. Hardware trojan mitigation for securing on-chip networks from dead flit attacks. In *Proceedings of IFIP/IEEE International Conference on Very Large Scale Integration*, pages 1–6, 2022.

[115] Vedika J. Kulkarni, R. Manju, Ruchika Gupta, John Jose, and Sukumar Nandi. Packet header attack by hardware trojan in noc based tcmp and its impact analysis. In *Proceedings of IEEE/ACM International Symposium on Networks-on-Chip*, pages 21–28, 2021.

[116] Mitali Sinha, Setu Gupta, Sidhartha Sankar Rout, and Sujay Deb. Sniffer: A Machine Learning Approach for DoS Attack Localization in NoC-Based SoCs. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 11(2):278–291, 2021.

[117] Rajesh JS, Dean Michael Ancajas, Koushik Chakraborty, and Sanghamitra Roy. Runtime detection of a bandwidth denial attack from a rogue network-on-chip. In *Proceedings of ACM International Symposium on Networks-on-Chip*, 2015.

[118] Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R Hower, Tushar Krishna, Somayeh Sardashti, et al. The gem5 simulator. *ACM SIGARCH computer architecture news*, 39(2):1–7, 2011.

[119] James Jeffers, James Reinders, and Avinash Sodani. *Intel Xeon Phi processor high performance programming: knights landing edition*. Morgan Kaufmann, 2016.

[120] Simon Hammond, Courtenay Vaughan, and Clay Hughes. Evaluating the intel skylake xeon processor for hpc workloads. In *Proceedings of IEEE International Conference on High Performance Computing & Simulation*, pages 342–349, 2018.

[121] Niket Agarwal, Tushar Krishna, Li-Shiuan Peh, and Niraj K Jha. GARNET: A detailed on-chip network model inside a full-system simulator. In *Proceedings of IEEE international Symposium on Performance Analysis of Systems and Software*, pages 33–42, 2009.

[122] Tushar Krishna and Srikant Bharadwaj. Interconnect modeling for homogeneous and heterogeneous multiprocessors. In *Network-on-Chip Security and Privacy*, pages 31–54. Springer, 2021.

[123] Ankur Limaye and Tosiron Adegbija. A workload characterization of the spec cpu2017 benchmark suite. In *Proceedings of IEEE International Symposium on Performance Analysis of Systems and Software*, pages 149–158, 2018.

[124] Michel A. Kinsy, Michael Pellauer, and Srinivas Devadas. Heracles: Fully synthesizable parameterized mips-based multicore system. In *International Conference on Field Programmable Logic and Applications*, pages 356–362, 2011.

[125] M Meraj Ahmed, Abhijitt Dhavlle, Naseef Mansoor, Sai Manoj Pudukotai Dinakarrao, Kanad Basu, and Amlan Ganguly. What Can a Remote Access Hardware Trojan do to a Network-on-Chip? In *Proceedings of IEEE International Symposium on Circuits and Systems*, pages 1–5, 2021.

[126] Hansika Weerasena and Prabhat Mishra. Security of electrical, optical, and wireless on-chip interconnects: A survey. *ACM Transactions on Design Automation of Electronic Systems*, 29(2), 2024.

[127] Junfeng Fan, Kazuo Sakiyama, and Ingrid Verbauwhede. Elliptic curve cryptography on embedded multicore systems. *Design Automation for Embedded Systems*, 12: 231–242, 2008.

[128] George Hatzivasilis, Konstantinos Fysarakis, Ioannis Papaefstathiou, and Charalampos Manifavas. A review of lightweight block ciphers. *Journal of cryptographic Engineering*, 8:141–184, 2018.

[129] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon v1. 2: Lightweight authenticated encryption and hashing. *Journal of Cryptology*, 34: 1–42, 2021.

[130] Julia Borghoff, Anne Canteaut, Tim Güneysu, Elif Bilge Kavun, Miroslav Knezevic, Lars R Knudsen, Gregor Leander, Ventzislav Nikov, Christof Paar, Christian Rechberger, et al. PRINCE–a low-latency block cipher for pervasive computing applications. In *Proceedings of Advances in Cryptology–ASIACRYPT*, pages 208–225. Springer, 2012.

[131] Kitae Jeong, Yuseop Lee, Jaechul Sung, and Seokhie Hong. Improved differential fault analysis on PRESENT-80/128. *International Journal of Computer Mathematics*, 90(12):2553–2563, 2013.

[132] Simon Heron. Advanced encryption standard (AES). *Network Security*, 2009(12): 8–12, 2009.

[133] Fang-Yu Rao. On the security of a variant of elgamal encryption scheme. *IEEE Transactions on Dependable and Secure Computing*, 16(4):725–728, 2019.

[134] Jean-Philippe Aumasson and Daniel J. Bernstein. SipHash: A Fast Short-Input PRF. In Steven Galbraith and Mridul Nandi, editors, *Proceedings of Progress in Cryptology - INDOCRYPT*, pages 489–508. Springer, 2012.

[135] Aaron M Johnson, Paul Syverson, Roger Dingledine, and Nick Mathewson. Trust-based anonymous communication: Adversary models and routing algorithms. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 175–186, 2011.

[136] Yonglin Ren and Azzedine Boukerche. Modeling and managing the trust for wireless and mobile ad hoc networks. In *Proceedings of IEEE International Conference on Communications*, pages 2129–2133, 2008.

[137] Tameem Eissa, Shukor Abdul Razak, Rashid Hafeez Khokhar, and Normalia Samian. Trust-based routing mechanism in manet: Design and implementation. *Mobile Networks and Applications*, 18:666–677, 2013.

[138] Guangjie Han, Jinfang Jiang, Lei Shu, Jianwei Niu, and Han-Chieh Chao. Management and applications of trust in wireless sensor networks: A survey. *Journal of Computer and System Sciences*, 80(3):602–617, 2014.

[139] Theodore Zahariadis, Helen C Leligou, Panagiotis Trakadas, and Stamatis Voliotis. Trust management in wireless sensor networks. *European Transactions on Telecommunications*, 21(4):386–395, 2010.

[140] Tigist Abera, N Asokan, Lucas Davi, Farinaz Koushanfar, Andrew Paverd, Ahmad-Reza Sadeghi, and Gene Tsudik. Things, trouble, trust: on building trust in IoT systems. In *Proceedings of IEEE Annual Design Automation Conference*, pages 1–6, 2016.

[141] Eberle A. Rambo, Alexander Tschiene, Jonas Diemer, Leonie Ahrendts, and Rolf Ernst. Failure analysis of a network-on-chip for real-time mixed-critical systems. In *Proceedings of Design, Automation Test in Europe Conference Exhibition*, pages 1–4, 2014.

[142] S. Murali, T. Theocharides, N. Vijaykrishnan, M.J. Irwin, L. Benini, and G. De Micheli. Analysis of error recovery schemes for networks on chips. *IEEE Design Test of Computers*, 22(5):434–442, 2005.

[143] Praveen Bhojwani, Rohit Singhal, Gwan Choi, and Rabi Mahapatra. Forward error correction for on-chip interconnection networks. In *Unique Chips and Systems*, pages 341–354. CRC Press, 2018.

[144] Jean-Philippe Aumasson and Daniel J. Bernstein. Siphash: A fast short-input prf. In Steven Galbraith and Mridul Nandi, editors, *Progress in Cryptology - INDOCRYPT*, pages 489–508. Springer, 2012.

[145] Mubashir Hussain and Hui Guo. Packet Leak Detection on Hardware-Trojan Infected NoCs for MPSoC Systems. In *Proceedings of ACM International Conference on Cryptography, Security and Privacy*, page 85–90, 2017.

[146] Haiying Shen and Lianyu Zhao. ALERT: an anonymous location-based efficient routing protocol in MANETs. *IEEE Transactions on Mobile Computing*, 12(6): 1079–1093, 2012.

[147] Na Wang, Junsong Fu, Jian Li, and Bharat K Bhargava. Source-location privacy protection based on anonymity cloud in wireless sensor networks. *IEEE Transactions on Information Forensics and Security*, 15:100–114, 2019.

[148] Pandi Vijayakumar, Mohammad S Obaidat, Maria Azees, SK Hafizul Islam, and Neeraj Kumar. Efficient and secure anonymous authentication with location privacy for IoT-based WBANs. *IEEE Transactions on Industrial Informatics*, 16(4):2603–2611, 2019.

[149] Clay Shields and Brian Neil Levine. A protocol for anonymous communication over the internet. In *Proceedings of ACM Conference on Computer and Communications Security*, pages 33–42, 2000.

[150] Ramzi A Haraty, Maram Assi, and Imad Rahal. A Systematic Review of Anonymous Communication Systems. *ICEIS (2)*, pages 211–220, 2017.

[151] Stefan Rass, Raphael Wigoutschnigg, and Peter Schartner. Doubly-anonymous crowds: Using secret-sharing to achieve sender-and receiver-anonymity. *J. Wirel. Mob. Networks Ubiquitous Comput. Dependable Appl.*, 2(4):27–41, 2011.

[152] Andrei Serjantov and George Danezis. Towards an information theoretic metric for anonymity. In *Proceeding of International Workshop on Privacy Enhancing Technologies*, pages 41–53. Springer, 2002.

[153] Travis Boraten and Avinash Karanth Kodi. Mitigation of denial of service attack with hardware trojans in noc architectures. In *Proceedings of IEEE International Parallel and Distributed Processing Symposium*, pages 1091–1100, 2016.

[154] Akshay BP, Ganesh KM, Thippeswamy DR, Vishnu S Bhat, Anitha Vijayakumar, Ananda YR, and John Jose. Implementation of a novel fault tolerant routing technique for mesh network on chip. In *Proceedings of International Symposium on VLSI Design and Test*, pages 495–506. Springer, 2019.

[155] Matthew Edman and Bülent Yener. On anonymity in an electronic society: A survey of anonymous communication systems. *ACM Computing Surveys*, 42(1), 2009.

[156] Yuankun Xue and Paul Bogdan. User cooperation network coding approach for noc performance improvement. In *Proceedings of the International Symposium on Networks-on-Chip*, 2015.

[157] Dong-won Park, Nam Su Chang, Sangyub Lee, and Seokhie Hong. Fast Implementation of NIST P-256 Elliptic Curve Cryptography on 8-Bit AVR Processor. *Applied Sciences*, 10(24), 2020.

[158] W Timothy Polk, Donna F Dodson, William E Burr, Hildegard Ferraiolo, and David Cooper. Cryptographic algorithms and key sizes for personal identity verification. *NIST Special Publication*, 800:78–4, 2015.

[159] Mohammad Humam Khan, Ruchika Gupta, John Jose, and Sukumar Nandi. Dead flit attack on NoC by hardware trojan and its impact analysis. In *Proceedings of the International Workshop on Network on Chip Architectures*, page 10–15, 2021.

[160] Ming Li, Qing-An Zeng, and Wen-Ben Jone. DyXY - a proximity congestion-aware deadlock-free dynamic routing method for network on chip. In *Proceedings of ACM/IEEE Design Automation Conference*, pages 849–852, 2006.

[161] Abhijit Das, Sarath Babu, John Jose, Sangeetha Jose, and Maurizio Palesi. Critical packet prioritisation by slack-aware re-routing in on-chip networks. In *Proceedings of IEEE/ACM International Symposium on Networks-on-Chip*, pages 1–8, 2018.

[162] Jean Paul Degabriele, Jérôme Govinden, Felix Günther, and Kenneth G Paterson. The security of ChaCha20-Poly1305 in the multi-user setting. In *Proceedings of ACM SIGSAC Conference on Computer and Communications Security*, pages 1981–2003, 2021.

List of Publications

From Thesis

J1 **Syam Sankar**, Ruchika Gupta, John Jose, and Sukumar Nandi, "TROP: TRust-aware OPportunistic Routing in NoC with Hardware Trojans", *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, Volume 29, Issue 2, pp. 1-25, March 2024.

J2 **Syam Sankar**, Ruchika Gupta, John Jose, and Sukumar Nandi, "Sec-NoC: A Lightweight Secure Communication System for On-Chip Interconnects", *IEEE Embedded Systems Letters (ESL)*, Volume 16, Issue 2, pp. 214-217, June 2024.

J3 **Syam Sankar**, Ruchika Gupta, John Jose, and Sukumar Nandi, "SARON: Secure Anonymous ROuting for NoC Communications", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, Under Revision - Awaiting decision.

C1 **Syam Sankar**, John Jose, Ruchika Gupta, and Sukumar Nandi, "Enhancing Anonymity in NoC Communication to Counter Traffic Profiling by Hardware Trojans", *In Proceedings of 16th IEEE International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC), Singapore*, pp. 560-567, 2023.

C2 **Syam Sankar**, Lissiyas Antony, Ruchika Gupta, John Jose, and Sukumar Nandi, "Exploring Trustable Paths in Network-on-Chip for Low-Slack Packets", *In Proceedings of 20th IEEE International SoC Design Conference (ISOCC), South Korea*, pp. 85-86, 2023.

PhD Forum / Student Research Forum-Presentations

1 **Syam Sankar** and John Jose, "Securing Network on Chip using Trust-Aware Routing and Anonymous Packet Transmission", *ESWEEK Ph.D Forum-2023, Germany*, Poster Presented.

2 **Syam Sankar** and John Jose, "Securing Network on Chip using Trust-Aware Routing and Anonymous Packet Transmission", *DATE Ph.D Forum-2024, Spain*, Submission Accepted Only.

3 **Syam Sankar** and John Jose, "TROP: TRust-aware OPportunistic Routing in NoC with Hardware Trojans", *IEEE GCON-2023, IIT Guwahati-Student Research Forum (SRF)*, Poster Presented. -**BEST SRF Poster Award**

3 **Syam Sankar** and John Jose, "Enhancing Security Features of Network-on-Chip Using Lightweight Cryptosystem, Trust-Aware Routing, and Anonymous Communication", *VLSID 2025, Bengaluru, Student Research Forum (SRF)*, Poster Presented.

Outside Thesis

J1 Ruchika Gupta, Vedika J. Kulkarni, **Syam Sankar**, John Jose, and Sukumar Nandi, "A Dynamic Re-configurable Cryptographic Solution for Packet Header Attacks on Network-on-Chip", *ACM Transactions on Embedded Computing Systems (TECS)*, Under Major Revision.

C1 Manju Rajan, Arya Phadke, **Syam Sankar**, and John Jose "Handling Packet Blocking Attack in NoC using Traffic Snooping", *In Proceedings of IEEE International Conference on Omni-layer Intelligent Systems (COINS), Kings College London, UK*,

Vita

Syam Sankar joined the PhD programme in the Department of Computer Science and Engineering at the Indian Institute of Technology, Guwahati (IITG), India, in July 2021. He was affiliated with the Multicore ARchitecture and Systems (MARS) Lab in the Department of Computer Science and Engineering. He was a recipient of the prestigious Prime Minister's Research Fellowship (PMRF) in the December 2021 cycle. He was the first scholar to receive PMRF in the Department of Computer Science and Engineering. He is also an Assistant Professor in the Department of Computer Science and Engineering at NSS College of Engineering, Palakkad, Kerala. He was on deputation to pursue PhD programme at IITG. He received his Master of Technology in Computer and Information Science from the Cochin University of Science and Technology (CUSAT), Kerala, India, in 2015. Prior to that, he received his Bachelor of Technology in Computer Science and Engineering from CUSAT in 2013. His PhD work focuses on improving the security features of NoC communication. Both the ESWEEK 2023 PhD Forum and the DATE 2024 PhD Forum have accepted his thesis work. His current research interests also include optimising the performance of multi-core architectures and other security issues in TCMP.

January 2025