A Survey on Security Mechanisms for NoC-based Many-Core SoCs

Luciano Lores Caimi[2], Rafael Faccenda[1], Fernando Gehm Moraes[1],

[1]PUCRS – School of Technology - Av. Ipiranga 6681, Porto Alegre, Brazil, USA
[2]UFFS, Federal University of Fronteira Sul, Av. Fernando Machado 108, Chapecó, Brazil
fernando.moraes@pucrs.br

Abstract— **The adoption of many-cores systems introduces the concern for data protection as a critical design requirement due to the resource sharing and the simultaneous executions of several applications on the platform. A secure application that processes sensitive data may have its security harmed by a malicious process. The literature contains several proposals to protect many-cores against attacks, focusing on the protection of the application execution or the access to shared memories. However, there is a gap to be fulfilled: a solution covering the entire application lifetime, including its admission, execution, and peripheral's access. This survey discusses three security-related issues: the secure admission of applications, the prevention of resource sharing during their execution, and the safe access to external devices. This survey concludes with an evaluation of the studied methods, pointing out directions and research opportunities.**

Index Terms— **NoC-based Many-core System; Security; Application Admission; Secure Execution; Secure Zones; Threats.**

I. Introduction

Many-core SoCs (System on Chip) are platforms that provide high connectivity and massive parallelism for running a wide variety of applications. The use of many-cores systems is continuously increasing due to the growing number of integrated processing cores into one single chip and the diversity of fields to where they have been applied to. A many-core SoC contains PEs (Processing Element) and IPs (Intellectual Property) modules interconnected by complex communication infrastructures, such as hierarchical buses or NoC (Network on Chip), that must be able to handle the high communication demands [1].

In the context of this survey, the term $MCSoC$ refers to NoC-based many-core SoCs. $MCSoCs$ are becoming the solution to meet the high-performance demand of embedded systems while maintaining the power consumption constraints during its execution. Examples of modern architectures with a large number of processors interconnected by NoC includes the Mellanox family TILE-Gx72 (72 cores) [2], Intel Knights Landing [3] and Oracle M8 (32 cores) [4], Kalray array (256 cores) [5], KiloCore chip (1,000 cores) [6], and Esperanto with 1,100 RISC-V cores [7]. Recently, even complex architectures are following the $MCSoC$ trend, such as Intel Xeon i9 (18 x86-processors) [8].

As the adoption and complexity of $MCSoCs$ increases, data protection's concern appears as a new design requirement [9]. A $MCSoC$ may be employed in scenarios where availability is a critical factor and downtimes must be minimized. Simultaneously, the system may also need to handle sensitive information; thus, it is necessary to protect this data from unauthorized access.

According to [10], not only data protection, unauthorized access and availability are concerns on $MCSoC$ design. The following seven *security principles* are generally accepted as the foundation of a good security solution being the first three principles mandatory features:

- Confidentiality: the property of non-disclosure of information to unauthorized processes, entities or users;
- Availability: the protection of assets from DoS (Denial-of-Service) threats that might impact any of the system's resources availability;
- Integrity: the prevention of modification or destruction of an asset by an unauthorized entity or user;
- Authentication: the process of establishment and validation of a claimed identity;
- Authorization: the process of determining whether a validated entity is allowed to access a secured resource based on attributes, predicates or context;
- Auditing: the property of logging sufficient system activities to reconstruct events;
- Nonrepudiation: the prevention of any participant denying his role in the interaction once it is completed.

$MCSoCs$, beyond their inherent scalability, provide massive parallelism and high performance to the users. In such systems, several applications execute simultaneously, sharing computation (processors) and communication (routers and links) resources. However, this *resource sharing* also leads to security and trust problems, requiring the design of solutions that prevent malicious entities from exploring vulnerabilities and breaking any of the security principles. Furthermore, those malicious entities can damage the system (hardware and Operating System) or the application at different moments of the application's lifetime, compromising their admission and execution through the resource sharing feature of the $MCSoC$ [11–13].

The application's lifetime encompasses three main phases: (a) before the execution we need to admit the application to the $MCSoC$, mapping the tasks and reserving resources for execution; (b) during the application execution the computation resources run the algorithms and use communication resources to inter-task cooperation and peripherals access; (c) finally, at the end of the execution, the resources release must be provided to enable its use by another application.

Digital Object Identifier 10.29292/jics.v16i2.485

The execution of an application with security constraints comprises at least three assumptions. The first one is the secure admission of the application, to guarantee the object code integrity. The second assumption regards the application execution in an runtime environment protected from attacks. The third assumption is related to the protection of the communication with peripherals and shared memories (Figure 1).

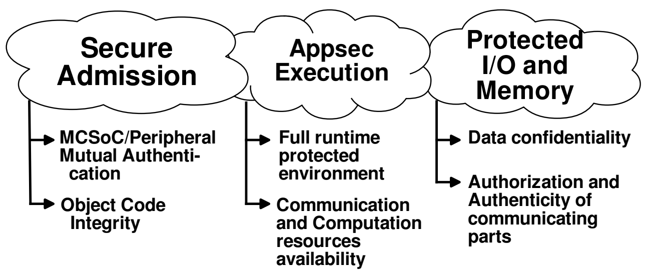

Secure Admission
Appsec Execution
Protected I/O and Memory
MCSoC/Peripheral Mutual Authentication
Object Code Integrity
Full runtime protected environment
Communication and Computation resources availability
Data confidentiality
Authorization and Authenticity of communicating parts

Fig. 1 Security constrains related to the secure application execution.

The *application admission* corresponds to the object code transfer from an off-chip entity to the $MCSoC$. With respect to security in this process, the many-core must trust on the entity transmitting the application and the integrity of the application must be verified to avoid the insertion of malicious code. Examples of attacks in this phase are Trojan Horses, man-in-the-middle and spoofing. While solutions to these issues exist for the Internet, computer networks, and software [14–16], few works are still applied to the $MCSoC$.

At *execution time*, a malicious attacker may have access to sensitive computation or communication data. Therefore, a secure application that processes sensitive data will have its security harmed by a malicious process. Examples of attacks that occur in this phase include DoS, timing attack, side-channel attack and information leakage (snooping) [17–19]. The adoption of firewalls, encryption mechanisms, and resource isolation through secure zones are common strategies to deal with those security threats.

Relative to the *communication with external devices*, unauthorized access to instructions and data in shared memory and peripherals can compromise the applications' execution, resulting in attacks of information tampering or information leakage. Examples of such attacks include DoS and timing attack [20, 21].

The $MCSoC$ research field is broad, with several research groups working in this area. Concerns related to NoC security is not a recent thread, with publications dating from the beginning of the 2000's, such as [22–25].

Table I details how this survey is organized. Emphasis is given to mechanisms used to protect the system on the different lifetime phases, considering the positioning of the mechanisms: at the NoC level; at the $MCSoC$ level; in other systems or research areas.

Section II. presents the threat model, discussing the $MCSoC$'s vulnerabilities for each security principle. Then, still in this context, details attacks that can be carried-out by exploring those vulnerabilities.

During the application admission phase, authentication mechanisms are applied to the entities in order to ensure the integrity of the application's object code (Section III.). With respect to the application's execution, the review focus on the protection of the communication resources, the computation resources, and works that considers the protection of both (Section IV.). The review of proposed mechanisms to protect the access to peripherals (mainly shared memory) is presented in Section V.. Each related work is summarized according to the following structure: (*i*) the attacks employed to obtain some advantage; (*ii*) the threats mitigation methods; (*iii*) the operation of the method; (*iv*) the cost regarding the area, power or latency of the proposed mechanisms.

Section VI. concludes this survey, presenting a discussion, alongside the issues and gaps found in the reviewed works.

Table I.: Security mechanisms proposals to distinct application lifetime phase with respective positioning level

App. lifetime phase ↓	NoC	$MCSoC$	other area
Admission **Section III.**	–	–	[16] Zero Knowledge Proof
	[26] MAC	–	[27] ECDH protocol
Execution **Section IV.**	[28] Packet Certification	–	–
	[29] Auditing and Firewall	[30] Secure Zones and Sym. encryption	–
	[31] Routing Scheme	[32] Sym. and Asym. Encryption	–
	[33] Symmetric Encryption	[34] Symmetric Encryption	–
I/O Access **Section V.**	[20] Firewall	–	–
	[21] Routing Scheme	–	–

II. Threat Model

The resource sharing of $MCSoCs$ components introduces vulnerabilities to the applications running on it. These vulnerabilities compromise the security principles throughout the applications lifetime, by example:

- *Integrity*: a malicious entity can change the source code of the tasks, at the application admission, inserting Trojan Horses or backdoors causing a suspicious behavior of the application during its execution.
- *Confidentiality*: unauthorized access to the data writing or reading. With different applications sharing the $MCSoCs$, a malicious application can be loaded and executed by a given processor, accessing the memory to retrieve or leak critical data through malicious PEs or peripherals.
- *Availability*: disruption of the system by overloading resources. A malicious application generating packets with a high injection rate can produce this attack, overloading the communication infrastructure.
- *Authentication*: before the application admission, the $MCSoC$ must have confidence that the entity that want to run an application in the many-core system proofs its identity.
- *Authorization*: the deployment of an malicious application without authorization enables innumerable threats

in the system compromising its integrity, availability and integrity.

In the $MCSoCs$ context, it is possible to explore such vulnerabilities with attacks that compromise the system by using:

- *Denial-of-Service - DoS*: disruption of the system by overloading resources, compromising its availability. A malicious application task generating packets with a high injection rate can produce this attack, overloading the communication infrastructure.
- *Distributed Denial-of-Service - DDoS*: similar to DoS, uses multiples tasks to attack and disrupt the system by overloading resources, compromising its availability. A malicious application running in distinct PEs can coordinate an attack to a specific router overloading its communication capacity.
- *Timing attack*: explores the communication collision between the sensitive traffic and the attacker traffic. The latency interference induced by malicious traffic can provide to the attacker some information about the timing, frequency, and volume of the secure communication.
- *Hardware Trojans*: a malicious modification of the system's hardware (e.g., inserted into the NoC) aiming to sniff and leak sensitive data.
- *Spoofing*: a malicious application successfully falsifies its identity to obtain unauthorized privileges.
- *Hijacking*: an attempt to alter the system configuration to execute a set of abnormal tasks along with normal system operation (e.g. during the load of the operating system or an application).
- *Man-in-the-Middle - MitM*: an attack where the attacker secretly relays and alters the communication between the external entity and the $MCSoC$, in such way that each one believes they are directly communicating with each other. Enabling the attacker to send malicious data or obtain secret information of the $MCSoC$.
- *Trojan Horse and backdoor*: the tampering of the task's source code during the admission of the application can insert malicious code that, during the application execution, can compromise the availability of the $MCSoC$ and the confidentiality of the data.

III. Protecting the application admission

The application admission corresponds to the object code transfer from an off-chip entity to the $MCSoC$. With respect to security in this process, each actor (external entity and $MCSoC$) must confirm the other part's identity, and the integrity of the application must be verified to avoid the tampering of the application's object code. Solutions to these issues exist for the Internet, computer networks, and software using techniques such as ZKP (Zero Knowledge Proof) [16], DH (Diffie-Hellman) protocol based on ECC (Elliptic Curve Cryptography) [27] and MAC [26]. Caimi et al. [35] present a solution to the application admission in $MCSoCs$.

(i) Zero Knowledge Proof protocol

In [16], the Authors present a secure lightweight and energy efficient authentication scheme for WBAN (Wireless Body Area Network) called BANZKP. The scheme is based on a ZKP protocol and a commitment scheme to authenticate the sensor nodes in the WBAN network. They used ZKP to confirm the identity of the sensor nodes while the commitment scheme deals with replay attacks. According to the authors, after the authentication success, an encryption mechanism (not presented in the paper) provides the message privacy protection. Evaluation uses the Omnet++ simulator. Authors claim that the proposed method reduces the memory consumption by 56.13% and energy consumption by 94.11% when compared with the alternative protocol TiniZKP.

(ii) ECDH protocol

He et al. [27] present an authentication scheme for session initiation protocol based on ECDH applied to multimedia services. The proposed scheme consists of three phases: system setup; registration; and authentication. At the setup phase, the system generates a point (P) over an elliptic curve using a large prime number, an integer (K_s) as the secret key, then calculate the P_{pub} = K_s x P as a public key (over the finite field F_P and modular arithmetic over P), and publish these parameters, except K_s. At registration phase, the user becomes a new legal user sending the username and password, and the server computes two secret values, the first based on the user password and another based on K_s value. Then the server calculates and stores a password auditor based on these two values. At the authentication phase, the user sends a request using the username and a value (R_1), calculated with P_{pub}; the server responds with a challenge using (R_1); the user validate the server's challenge and, if it holds, compute a response that involves the user password, the R_1, the server's challenge, and the username. Finally, the server validates the answer to accept the user's request. After the mutual authentication, a shared session key is calculated in both sides using common values. The paper presents a security and performance analysis based on the number of elliptic curve scalar multiplications, modular multiplications, modular inversions, and one-way hash functions operations used on the scheme.

(iii) Message Authentication Code (MAC)

Sepúlveda et al. [26] propose a runtime mechanism based on MAC and PUF (Physical Unclonable Function) to provide memory integrity and authentication. The MAC uses the SipHash algorithm. The proposed mechanism prevents code injection and memory modification attacks, including spoofing, reallocation and replay attacks to the off-chip components. The mechanism is divided into three stages: key generation; MAC initialization and application installation; and operation. At the key generation stage, one key for each application is derived utilizing a challenge PUF and a random number. These two are also utilized to compute the helper data, which is part of the generation and reconstruction of the key. Application identifier, PUF challenge, and helper data are utilized by the Authentication Controller module (Figure 2) to trigger the regeneration of the application-specific key which is required for MAC initialization during application installation as well as during the SoC operation.

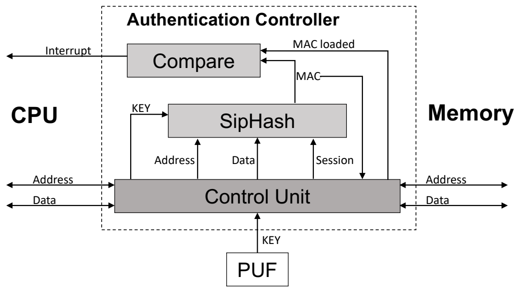


Fig. 2 The Authentication Controller module (adapted from [26]).

At the MAC Initialization and Application Installation stage, before the normal system operation, the MAC for all applications that are going to be executed are computed and stored in the off-chip memory. During SoC operation phase, code/data may be migrated to another IP core or wrote back to the main memory. In such situations, MACs are computed in the Authentication Controller and stored in the off-chip memory. During a read access, the loaded data is used to recompute the MAC, which is then compared with the MAC previously stored in memory. For an authentic memory line the computed MAC matches to the MAC stored in memory. The Authors implement the mechanism using Xilinx Nexys4 FPGA (Field Programmable Gate Array) and the evaluation FPGA resource utilization shows the following: overhead of the modules when compared with the baseline system: +132% of FFs (Flip-Flops) and +135% of LUTs to the entyre secure PUF module; +19% of FFs (Flip-Flops) and +17% of LUTs to the authentication Controller module. Performance evaluation shows a low impact on the application degradation due to the mechanism: up to 25% (write intensive Qsort application).

(iv) ECDH and MAC

Caimi et al. [35] propose a solution to authenticate external entities responsible by deploying secure applications (App_{sec}) to $MCSoC$ using an ECDH (Ellipt Curves Diffie-Hellman) protocol. At the App_{sec} admission, a protocol using a MAC (based on the SipHash algorithm) verifies the object code integrity of each application's task. Only after all App_{sec} tasks pass through the integrity validation the application start.

The entity's authentication evaluation shows an overhead of 1.68 seconds using an ARM processor running at 500 MHz. The object code integrity verification using SipHash costs 32K clock cycles per KB.

IV. Protecting the application execution

The literature presents a diversity of mechanisms used to protect communication, computation and memory accesses in $MCSoCs$. Mechanisms protecting the communication include: (*i*) firewalls; (*ii*) secure zones; (*iii*) routing schemes; (*iv*) temporal network partitioning; (*v*) cryptography; (*vi*) packet certification. To protect computation, the main mechanism used is spatial and/or logical isolation. The mechanism protecting memory accesses include routing schemes and firewalls.

A. *Protecting communication*

Hardware Trojans (HT) or malicious processes are threats that can be utilized to access sensitive information or break the communication subsystem security. The most reported attacks are DoS [17, 19, 36], HT [28, 29], timing side-channel attack [31, 37], and attacks to confidentiality and integrity [29, 38].

We describe in this Section works that use different mechanisms to specifically protect the communication subsystem. However, in such cases, the computation is still exposed due to resource sharing that is not directly addressed in the proposed mechanisms.

(i) Firewalls

In the on-chip communication scope, a firewall is a hardware barrier placed at the communication structure ports to control the input and output of an element. This mechanism is, in general, composed of a table to store the recognized trusted sources and a controller, which allows the certified traffic and blocks unauthorized traffic.

Rajesh et al. [29] propose a runtime latency auditor for NoCs, called RLAN, to dynamically monitor the on-chip resources availability and properly filter the malicious traffic. The goal of the proposed method is to prevent HT and mitigate attacks to the availability of the NoC. According to the Authors, RLAN is a non-invasive technique that can work without any modification with third part IP NoCs. The method is implemented at the NI (Network Interface), as shown in Figure 3.

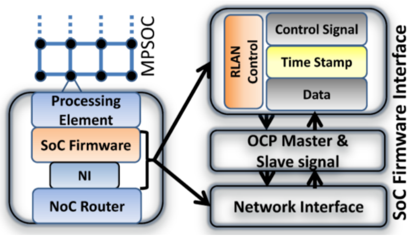
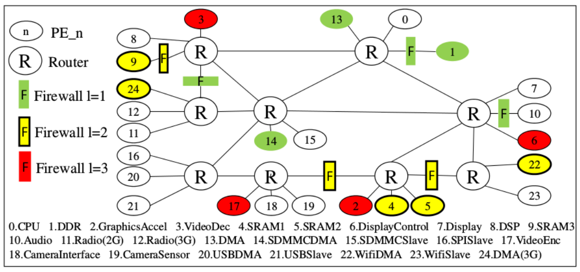


Fig. 3 Block diagram of RLAN's network interface [29].

The principle adopted by the RLAN design is that packets traversing routes with significant overlap (spatial similarity) around the same time (temporal similarity) have comparable latencies. The method includes two major steps: (a) all packets are tagged with a timestamp to enable latency computation; (b) creation of Source/Destination traffic in RLAN. The second step creates a control traffic to compute the reference latency, using it to detect attacks by HTs or malicious processes. The authors use BooKSim 2.0 Simulator to evaluate the performance of the method, and a 45nm TSMC standard cell library to evaluate power and area. Results show that RLAN incurs an overhead of 12.73% in area, 9.84% in power and 5.4% in terms of network latency when compared to the baseline NI.

Hu et al. [17] propose a three-level firewall to provide access control, authentication, and availability of the communication system, preventing information leakage and DoS attack. The proposed method makes a design time analysis of the traffic and the NoC architecture [39] applying Integer Linear Programming (ILP) to select the levels and position of the firewalls: (a) between a PE and a router or; (b) between routers. Figure 4 illustrate the method in a smartphone application.


n PE_n
R Router
F Firewall l=1
F Firewall l=2
F Firewall l=3
0.CPU 1.DDR 2.GraphicsAccel 3.VideoDec 4.SRAM1 5.SRAM2 6.DisplayControl 7.Display 8.DSP 9.SRAM3
10.Audio 11.Radio(2G) 12.Radio(3G) 13.DMA 14.SDMMCDMA 15.SDMMCSlave 16.SPISlave 17.VideoEnc
18.CameraInterface 19.CameraSensor 20.USBDMA 21.USBSlave 22.WifiDMA 23.WifiSlave 24.DMA(3G)
Fig. 4: Example of optimized NoC topology for a smartphone application [17].

The Authors' goal is to reduce the communication overhead required for security information in packets' headers. The authors do not present the area, power, or performance costs. Results show a 30% to 63% overhead reduction in header size of packets when the firewalls are positioned between routers over a standard solution, i.e., firewalls connected to the NIs.

Azad et al. [40] propose a firewall to guarantee the platform's integrity and confidentiality. The firewall is placed at the NI and has two tables: (*i*) *initiator table*, which checks if the source has permission to send messages, and (*ii*) *target table*, which verifies if the message is allowed to enter the target unit. Another important aspect of this proposal is that a Security Manager configures the firewalls, which send configuration messages protected with MAC. There is an area increase of 139% in the secure NI with a firewall with 15 entries. The critical path delay of the baseline NI increases by 6.46% from 2.32ns to 2.47ns by adding the firewall. Integration of SipHash (MAC algorithm) for reconfiguration packets imposes an NI overhead of 17,105 μm^2 (TSMC 40 nm technology).

Another proposal of Azad et al. [41], named CAESAR-MCSoC, utilizes a group of firewalls to set an isolated cluster of nodes in the platform, building a Secure Zone with them. However, as the Firewalls are configurable, the focus is to promote a secure configuration protocol with encrypted and authenticated configuration packets. Implementation puts two encryption and authentication modules (CAESAR-AEGIS and CAESAR-ASCON) integrated into the Network Interfaces (NIs). The evaluations show an area overhead of 277.6% when NI uses the CAESAR-AEGIS, and 18% when uses CAESAR-ASCON (TSMC 40 nm technology). The delay in the critical path increases 17,8% with CAESAR-AEGIS and no additional delay with CAESAR-ASCON.

(ii) Routing Scheme

Sepúlveda et al. [36] present a runtime method to prevent timing side-channel attacks and information leakage. The work proposes two mechanisms: adaptive routing and random arbitration. The proposed method assumes that a malicious task in the path of a memory access may extract sensitive data from the communication flow's temporal behavior. To prevent the temporal behavior extraction, the first mechanism implements a random arbitration in the routers to remove the temporal correlation of malicious injected traffic and memory access. The second mechanism is the adaptive West-First routing method, which allows, in some situations, to make turns to escape from blocking conditions. Thus, the secure traffic deviates from the malicious traffic in the path automatically to get a free one. Results show that random arbitration and adaptive routing maintain an average throughput of 0.41 and 0.6 flits/cycle, respectively, to one secure traffic flow with several malicious injection rates. The area overhead due to the random arbitration and adaptive routing is 11% and 5%, respectively. The power overhead due to the random arbitration and adaptive routing is 9% and 8%, respectively.

Charles et al. [42] propose a lightweight Anonymous Routing (AR). This method inserts extra layers of protection that hide the source and target fields of the message and utilizes a three-way handshake protocol to establish the communication path (Route Initiation, Route Accept and Route Confirmation). As a result, the message exchange becomes untraceable for devices that do not access the key. AR results present 70% (69% on average) delay increase and 34% (28% on average) increase in execution time, both compared to the No-AR implementation

Indrusiak et al. [43] propose the implementation of route randomization in a many-core as a protection mechanism against SCA. In this case, varying the routes taken by sensitive traffic prevents the collision with malicious traffic provoked by an SCA attacker, making the SCA information extraction considerably harder since the timing measures are not precise. Results show distinct latency overhead according to the routing algorithm and the packet priority, presenting high dependency on traffic scenario, difficulting a performance evaluation.

(iii) Secure Zones - Routing Scheme

Fernandes et al. [31] propose a design time method that enables the creation of Secure Zones (SZ) based on the routing algorithm to mitigate DoS and timing side channel attacks. The authors extend the Segment-based Routing (SBR) to security purposes creating the SBR Security Zone Awareness (SBR-SZA) that enables the creation of SZs. After running the SBR-SZA, the Region-based Routing Algorithm (RBR) creates routing restrictions avoiding shared paths between different applications and deadlock free paths.

Figure 5 shows two cases of segment computation for the same six routers of the $MCSoC$. Depending on the NoC segments computed by SBR, the communication path between S and D can be either PIZ (Partial intra-zone communication), as shown in Figure 5(a); or FIZ (Full intra-zone communication), as shown in Figure 5(b).

The evaluation method uses two synthetic scenarios and the Numerical Aerodynamic Simulation (NAS) benchmark [44] to measure the routing tables size and the communica-

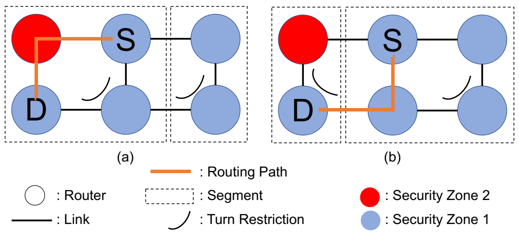


Fig. 5: Example of two SBR segment computations: a) the path among S and D goes through an insecure element due to a routing restriction; b) a set of restrictions in the segments enables a secure path between S and D. [31]

tion latency. Results showed up to 16.56% overhead in the routing tables size.

(iv) Secure Zones - Encryption

Sharma et al. [34] proposes an encryption mechanism for zone-to-zone secure communication. The authors present a runtime protocol (PF-ID-2PAKA) that generates private/public par keys for each IP core of the SoC. The secure zones are created dynamically having an anchor node responsible by the secure communication with other zones. The protocol enables dynamically creation of session keys used to encrypt the message flow between the anchor nodes, protecting the communication.

(v) Temporal Network Partitioning

Temporal network partitioning (TNP) employs explicit flow separation to avoid interference of low-priority flows in high priority flows. The goal of the approach is to mitigate DoS, timing side-channel attacks and information leakage [37, 38].

In [37, 45], the authors propose a design time method to create domains of noninterference to prevent DoS and timing attacks. Noninterference means that packet injection from one domain can never have any effect on the packets delivered from other domains. The domains are implemented using virtual channels, and the noninterference is obtained through bounded priority arbitration, called surf scheduling at each router port. In this schedule, a packet waits until it can be forwarded in one dimension (e.g. X-direction) and then does not experience any wait at any downstream router in this dimension. After finishing the first dimension, the packet might experience another wait until it can be forwarded to the next dimension. The authors call this *schedule surf scheduling* because a packet is like a surfer who waits to ride a wave to some location and then waits to ride another wave.

The proposed SurfNoC was implemented in BookSim 2.0, a cycle accurate simulator. The latency evaluation shows that the overhead of surf scheduling is almost independent of network size (average number of hops), leading to a constant overhead of 19 cycles (except for 16 nodes) because the packet wait time depends only on the number of dimensions and domains. The drawback of the proposed method is that increasing the number of domains also increases the number of virtual channels, increasing the router area an power consumption.

Wang et al. [38] propose a design time priority-based arbitration and a static limit mechanism to provide protection against information leakage and DoS. The idea is to assign high-priority to low-security traffic, in such way that its behavior is not affected by high-security traffic. With this scheme, when flows from two different security levels compete for the router traversal, the low-security flow always wins due to the arbitration police, avoiding a malicious task to infer timing information over the high-security flow. Virtual channels are statically allocated to each security domain to remove interference in buffers. To prevent a possible DoS attack due to this unfair arbitration scheme, the authors include an additional mechanism that monitors and limits the amount of the low-security traffic regardless the amount of high-security flows. The proposed method evaluation uses the Darsim Simulator. As expected, results show that the method increases the performance in low-security flow and decreases the high-security flow.

(vi) Cryptography

Ancajas et al. [28] present a runtime method to protect information leakage from HTs. The authors propose the Fort-NoC, a three-layer security mechanism. These security measures are introduced in the NI of the NoC. The first layer is the Data Scrambling (DS), which makes the HT activation harder. This is done with XOR cipher encryption [46] to encrypt the data before sending it to the NoC. The second layer is the Packet Certification (PC) that attach an encrypted tag at the end of the packet before injecting it into the NoC. This is obtained creating a random lookup table dynamically at the boot-time that creates a 16-bit unique identifier for each node in the system. Based on the destination node of a packet, each data packet embeds a tag containing the translated identifier of the destination node from the lookup table. The third layer is the Node Obfuscation (NObf) that decouples the source and destination nodes of a communication to increase side-channel resilience. The NObf is obtained with task migration. The evaluation of the area shows an overhead of 0.34% and 9.57% to PC and DS respectively. The power consumption evaluation shows no overhead to PC technique and overhead of 5.8% to DS technique. According to the authors, the performance evaluation shows an overhead of 5.9%.

Silva and Zeferino [47] propose the use of an AES block and a KDC (Key Distribution Center), adding authenticity and confidentiality in the message flow of the SoCIN NoC. The KDC share different master keys with each node (the authors do not explain how this is executed). During the operation, when a node A wants to send a sensitive message to node B, the node A send a session key request to the KDC. This request inform the communicating nodes (A and B) an is encrypted using his master key of node A. The KDC decrypt the request and generate (using a LFSR) a session key. The KDC sends the session key generated to node A encrypted with its master key and another message with the session key to node B encrypted with the note B's master key. At this point the two communicating nodes (A and B) have a session key to encrypt the message at source node A and decrypt message at target node B. Figure 6 illustrate the process.

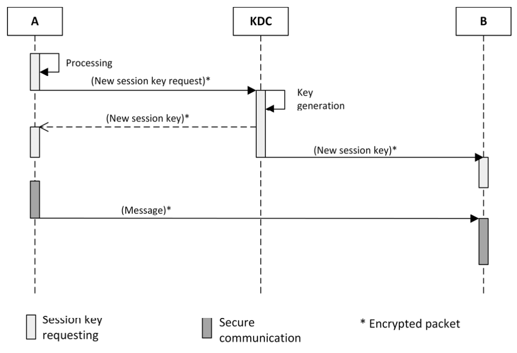


Fig. 6 Establishment of a secure session in [47].

In the paper, it is not clear if each message exchange needs a new key session or the same key session is used by all messages from node A to node B. The evaluation shows that the communication using the security mechanism is from 7 to 17.6 times slower if the secure session was already established or not, respectively. The hardware cost evaluation using Xilinx ISE synthesis toll shows the additional costs in LUTs, FFs and BRAMs equal 233.5%, 188.6%, and 687.5%, respectively.

Oliveira et al. [33] propose an architecture that includes a firewall capable of filtering incoming and outgoing NoC traffic, an AES cipher block to encrypt the NoC flow and, an auxiliary NoC that use a Hamiltonian path to configure the firewall rules and distribute the keys. Figure 7 presents the interfaces connected to the firewall. The router and the NI signals are the same. Instead of changing the interfaces, state machines in the firewall manage the flow control signals. The firewall may encrypt or not the packets according to an identifier in the packet.

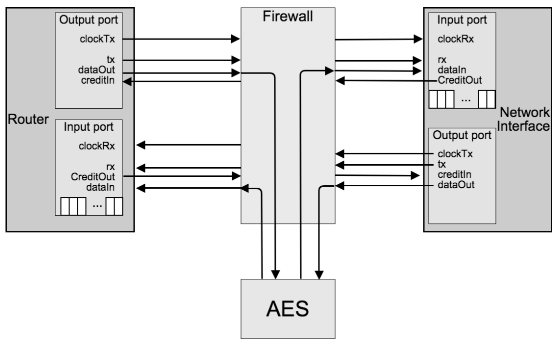


Fig. 7 Representation of the Firewall connections [33].

The firewall plus the AES module increases the router area by 193.7% and latency increases (*i*) in the best-case scenario (disturbing traffic in the path of the ciphered traffic) by 126.3%; (*ii*) in the worst-case scenario (contention due to the simultaneous need of encrypt and decrypt packets by the same AES block) by 395.92%.

Santana et al. [48] extend the aforementioned work [33], by adopting four security countermeasures: spatial isolation of applications, a dedicated network to send sensitive data, filters to block malicious traffic, lightweight cryptography. This proposal is decoupled from the $MCSoC$ architecture, protecting the system from attacks, such as Denial of Service and man-in-the-middle, ensuring confidentiality and integrity to applications. Performance evaluation present distinct behaviors, according to the application profile. For a synthetic application, the average latency increased by 13.63% and 46.51% without and with the AES encryption, respectively, compared to the baseline $MCSoC$. The overhead in the average latency of a real application (MPEG) is 2.55%.

In Kinsy et al. [12], the authors introduced Hermes, a secure multicore computing architecture framework. The proposed scheme claims to prevent DoS, virtual channel and physical memory attacks by creating a virtualization layer that isolates computing threads based on system and user-defined trust levels and security policies.

Hermes achieves both hardware and software views of secure processing by grouping processors into physical zones called wards and virtual logical zones called islands. The wards have different secure level defined at design time and each one have a anchor node responsible by its key management. The environment protects the physical memory using an MMU (Memory Management Unit) with access restrictions based on two tables that provide a firewall behavior to the MMU. Hermes achieve communication protection using several mechanisms: (*i*) DH protocol distributes public keys to anchor nodes of the physical zones, with join and leave operations to distribute the keys to the logical zones; (*ii*) a AES module in the NI to encrypt the traffic flow when required; (*iii*) a key manager in the NI to select an appropriate key during the operation; (*iv*) MAC and hash modules to generate session keys. Beyond those mechanisms, the environment provides a routing algorithm to prohibit or limit the traversal of zones by non-member generated traffic. According to the Authors, the hardware overhead to fully implement the security features of Hermes architecture is 17%. The performance results on the SPLASH-2 benchmarks presents an overhead of 1% to 9% across all the benchmarks when compared to the non-secure baseline architecture.

(vii) Packet Validation

Boraten et al. [49] propose a runtime packet-security (P-Sec) method, including a packet validation technique to protect compromised NoC architectures from fault injection side channel attacks (potentially DoS) and HTs by merging two error detection schemes, namely algebraic manipulation detection (AMD) and cyclic redundancy check (CRC) codes. According to the authors, in normal operating environments (not under attack) CRC is capable of detecting faults in packets since the fault rates are low. For cores sending sensitive data over the network, P-Sec is turned ON, switching CRC to AMD mode, protecting sensitive packets from fault injection attacks.

Figure 8 shows, at the top, the changing between normal operation (CRC) and secure operation (AMD). In the same Figure, at the middle, the logical arrangement of two modules and, at the bottom, the packet structures in the distinct operation modes. In the AMD technique the number of redundant bits is a function of the message length and the size of a random number internally used.

The work uses CRC messages with 204 bits length and CRC of 32 bits length; AMD messages with 204 bits

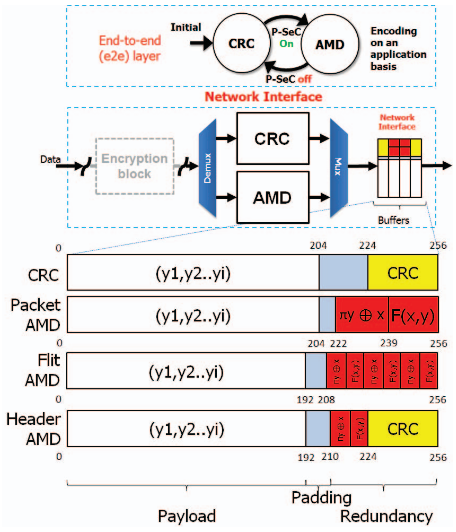


Fig. 8: Overview of P-sec architectural blocks and structure of packets [49].

length, random number of 17 bits and redundancy of 17 bits (204,17,17). The evaluation implements four methods (AMD, CRC-32, JTEC-QED and SECDED). Compared to SECDED, the smallest method, the power consumption is 2.68x and 8.36x greater to CRC-32 and AMD, respectively. The evaluated area, in comparison with SECDEC is 3.5x and 16.1x greater to CRC-32 and AMD, respectively. Performance results indicate P-Sec reduces overhead compared to Fort-NoCs.

B. *Protecting Computation*

The computation protection encompass mechanisms to avoid the processors sharing between distinct applications or deeply embedded into the processor micro-architecture mechanisms. The review shows that logical and spatial isolation are adopted [19, 50]. When just processors or clusters isolation is used, the communication system remain exposed to attacks. Solutions based on processor micro-architecture mechanisms such as obfuscated instruction execution [51] or PUF-based authentication architecture [52] are not addressed in this survey.

Real et al. [53] and their previous works [11, 19] propose a logical and spatial isolation of sensitive applications through the dynamic creation of SZs to mitigate DoS and cache SCA attacks at runtime. The architecture uses the MPSoCSim [54], a Mesh NoC where each router is connected to a cluster with four processors (with local memory), one shared memory and one shared bus (see Figure 9).

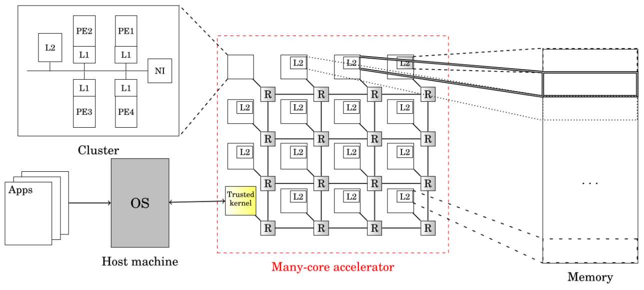


Fig. 9 Overview of architecture [53].

Only cluster resources are isolated by the SZ. If a task needs to communicate with a task in another cluster the message is sent through an insecure channel.

The evaluation focus on the creation of secure zones with different deployment strategies (number of clusters in the SZ) and execution scenarios (number of isolated application and their priorities). According to the chosen deployment and execution strategies, either the isolated applications performance or the non-isolated applications performance can be penalized. The performance overhead of the proposed mechanisms increases with the number of required secure zones. The worst-case shows an execution-time overhead up to 35.86% over the baseline.

ARM processors provide the ARM TrustZone (ATZ) [50], a hardware support for the creation at runtime of Trusted Execution Environments (TEEs), isolating the applications in the same processor. This feature creates two virtual processors and two Memory Management Units (MMU), allowing to execute a secure and a non-secure application simultaneously. However, at any instant, only a single domain in the system is secured. TEE allows the secure partition of shared memory to control the accesses to avoid data extraction and change (mitigating confidentiality and integrity attacks). Nevertheless, in multicore and many-core architectures, applications running on different processors share resources such as the communication infrastructure (NoC, buses) and memory. Thereby, with TEE, applications running on different processors are not protected from each other since sharing the communication infrastructure leads to possible leakage of information.

C. *Protecting Computation and Communication*

This section includes works that protect both computation and communication simultaneously [30, 32]. These works use firewalls or encryption mechanisms along with isolation.

(i) Secure Zones - Partition and encryption

Isakovic et al. [30] obtain computation and communication protection using spatial isolation with encryption mechanisms. The proposal is an architectural partitioning of the $MCSoC$ resources at design time to provide availability, confidentiality and integrity. This goal is reached by adopting security components like a *secure microkernel* and a *secure channel* infrastructure that includes cryptography and firewalls. The Authors also propose to migrate the security functions from application components to the security components. Furthermore, Spatial Isolation of applications and secure channels (encryption) is used to obtain a secure environment for applications. The proposed method uses the ACROSS MPSoC architecture [55], but does not detail the implementation methods and how the protocols work. Figure 10 shows the block diagram of architecture (TISS means Trusted Interface Subsystem). The authors also show a use-case of the proposal on a Engine Control Unit (ECU) but do not present results regarding area, power consumption or latency.

(ii) Secure Zones - Spatial isolation and encryption

Sepúlveda et al. [32] also protect computation and communication resources using spatial isolation with encryption

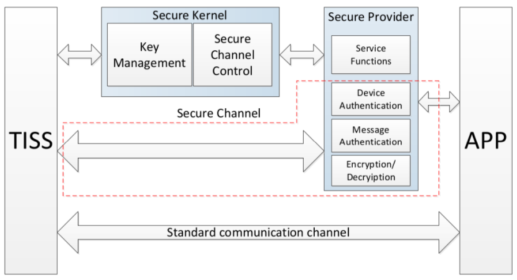


Fig. 10 Block diagram of Secure Communication Architecture [30].

mechanisms.The authors propose an NoC-based architecture that implements runtime disrupted SZs using three cryptographic techniques: Hierarchical Diffie-Hellman, Hierarchical Tree-based Diffie-Hellman and mapping key predistribution scheme. The method prevents attacks to availability, confidentiality, and integrity of the system. The architecture adopts two NoCs: (a) data NoC, used by the application data; (b) service NoC used to exchange the security control packets (key exchange, firewall rules, etc.). After mapping the application, one of the key agreement protocol is executed between the mapped PEs using the service NoC. The encryption/decryption is obtained XORing the message with the shared key. Figure 11 presents an example with one disrupt SZ, two malicious routers and one infected IP.

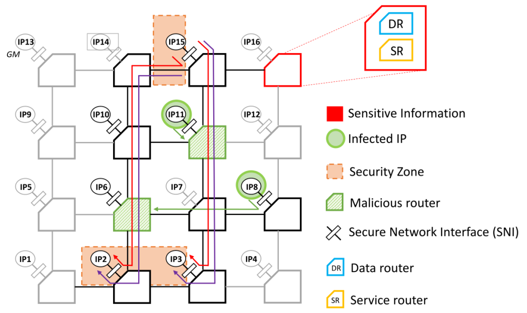


Fig. 11: Security zones at MPSoCs interconnected through a two-level NoC (Service and data NoC). [32]

The architecture was modeled in SystemC-TLM and RTL-VHDL, and the evaluation uses the SHOC simulation environment. Despite the use of cryptographic primitives like Diffie-Hellmann, the authors show an area overhead of 20% and a power consumption overhead of 12.7% over a baseline implementation. The latency evaluation overhead shows a dependency with the number PEs in the SZ, between 30% and 55% over the baseline implementation.

(iii) Opaque Secure Zones

Caimi et al. [56] propose the method Opaque Secure Zones (OSZ), enabling temporal and spatial isolation of applications, preventing the communication and computation resource sharing. The execution mechanism includes: *(i)* OSZ shape definition and positioning; *(ii)* wrapper activation; *(iii)* retransmission of lost packets in and out the OSZ boundaries; and *(iv)* launch application. Figure 12 illustrates a simplified view of the approach. In Figure 12(a), the $MCSoC$ contains one application in execution, App_1. Next, the manager processor maps an App_2, activating the wrappers at the boundary of the OSZ. At this moment (Figure12(b)), the App_1 traffic is blocked by the OSZ. Figure 12(c) shows the App_2 executing in the OSZ, and the App_1 traffic circumventing the region.

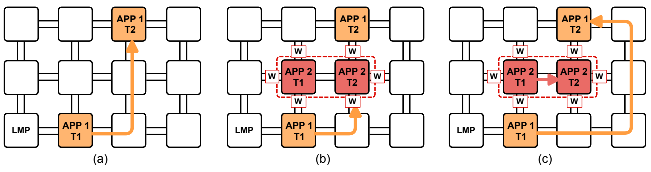


Fig. 12 Secure zone and dynamic reconfiguration of routing paths.

This work [56] stands out from related works because it covers all phases required to execute an application with security constraints using runtime mechanisms to tackle these issues: application admission, execution, and peripheral access. The protocol authenticates trustworthy entities and creates a shared key. The authenticated entities are enabled to deploy applications on the $MCSoC$, with an attached MAC (Message Authentication Code), which ensures both integrity and authentication at the same time. Reserving and isolating computation and communication resources inside a region of the $MCSoC$ guarantees the secure execution. During the application execution, the protection to the peripherals access is obtained with a lightweight encryption mechanism and packet integrity checking. After the application execution, the resources are released, and the memory is erased to avoid information leakage.

The performance evaluation considers the three phases. The admission phase has the overheads presented in section Section III..iv, where the authentication of external entities occurs once by the device. The object code's integrity verification impacts an extra time of 32K clock cycles per KB of the object code. Due to the isolation mechanism of OSZ, the execution phase does not have overhead in inter-task communication or task computation. About peripheral access, the communication overhead is up to 15% and depends on the protocol to enable the I/O message to pass through the OSZ and the encryption mechanism.

(iv) SDN (Software-Defined Networking

Most works adopt isolation in continuous regions, using *SZs*. Ruaro et al. [57] propose an alternative to *SZs*, which is the SDN-based management to implement the communication isolation at runtime. The method adopts two constraints. The first one is the spatial isolation at the computation level: secure tasks can only share a PE with tasks belonging to the same application. The second one is the spatial isolation at the communication level: map tasks in regions where the SDN subnets utilization is low to increase the probability of all applications' tasks receiving a dedicated connection. The communication isolation is supported by the SDN paradigm, which establishes exclusive paths for secure applications (circuit switching). Results show that the SDN-based approach presents a negligible latency to admit and execute a secure application, with a reduced hardware cost and higher computational resources utilization compared to *SZs*.

(v) Obfuscation

Reinbrecht et al. [58] present Guard-NoC, a platform protected against Side-Channel Attacks, specifically Timing Attacks. The protection is based on the *Obfuscation Module* (Figure 13), placed between the Network Interface and the local IC input. This module uses two processes to prevent timing attacks: blinding and masking. The blinding strategy implementation changes the response time of the ICs to have a constant value. Masking is applied to insert delays on the responses, operating as a noise source. Both strategies are effective against the attempts to read or access time measurements of that local IC. In addition to this obfuscation mechanism, the Guard-NoC protects the system's communication via *Switching Mechanism*. This NoC has dual switching, the packet switching is reserved to secure communications. Consequently, the circuit switching is destined to common packets transmission. The separation of secure and common traffic prevents the attackers from injecting malicious traffic directed to collide against secure traffic and infer timing values about a protected communication.

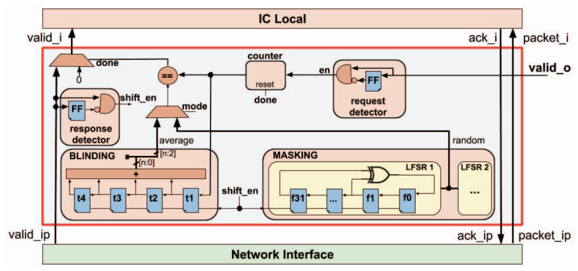


Fig. 13 Obfuscation Module [58].

The evaluation shows a performance degradation of 12.6% due to the blinding mechanism (worst-case) and 26.9% due to the masking mechanism. The Guard-Noc router presents an area and power overhead of 16% and 18%, respectively.

V. Protecting the Memory Access

When the $MCSoC$ architecture has shared memories, it is necessary to protect the memory accesses. The protection must prevent unauthorized tasks to read or write sensitive memory blocks and avoid information leakage in the communication subsystem.

(i) Routing Scheme

Reinbrecht et al. [21] and their previous works [59, 60] propose the improvement of the Prime + Probe (P + P) attack introduced by Osvik [61], expanding it to the communication structure of the $MCSoC$ (i.e. the NoC). The (P + P) attack is a timing side-channel attack that exploits the information leakage through communication timing behavior. The attack is executed over the interaction between the shared cache and the victim IP that is reading the AES secret key.

The proposed attack has two variations: (i) (P + P) firecracker; (ii) (P + P) arrow. The authors shows results to both attacks. In the (P + P) firecracker technique, 14 cryptographic tasks were needed to complete the attack, properly recovering 12 of 16 bytes from AES key. The (P + P) arrow technique performed 256 encryptions in the attack and recovered 9 of 16 bytes from AES key.

To mitigate the (P + P) attacks, the authors propose the Gossip NoC. The Gossip NoC combines two strategies to protect the $MCSoC$ against timing side-channel attacks: (*i*) detection, which includes a bandwidth monitoring and a gossip message generation in the presence of an abnormal behavior that enables the second strategy; (*ii*) protection, triggered when any gossip message is received and which is able to modify the route of the packet (XY routing algorithm to the YX).

The work Reinbrecht et al. [21] shows that the Gossip router increases the logic area of the NoC about 21%. The unprotected NoC represent 35% of the $MCSoC$, became 42.5%, meaning that the effective logic overhead in the system was 7.5%. When calculating the same impact for power, the authors obtain 16.2% of power overhead over the baseline router and 1.18% of power overhead in the entire system.

Another cache collision attack in NoC-based SoC is presented in [13]. The authors show the attack efficacy to fully recover an AES key. The paper does not propose a mechanism to mitigate the attack in the work.

(ii) Firewall

Grammatikakis et al. [62] and their previous works [20, 63, 64] propose a source-side firewall at the NI which, by checking the physical address against a set of rules, rejects untrusted CPU requests to the on-chip memory. The Authors claim that the proposed method protects against DDoS and threats related to data leakage, confidentiality, integrity, and availability. The firewall architecture has three modules: (a) the operating mode controller (OMC), that accepts, decodes and dispatches NoC firewall commands; (b) the segment-level rule-checking (SLRC), that processes incoming memory accesses and configuration commands; (c) the interrupt unit (INTU) that accepts interrupt requests from the OMC and SLRC modules besides reporting interrupt contexts to the CPU.

The evaluation environment uses the STNoC, a ring-based NoC topology. First, a time-annotated RTL description was implemented, then a set of GEM5 simulations are executed. Results show a reduction on power consumption by 30% due to the fact that malicious requests are prevented from entering the network, thus fewer packets are released, resulting in lower network traffic and smaller queues and activities. For the same reason, the delay of the packets to transverse the STNoC was decreased by 20.47%.

VI. State-of-the-Art Discussion

Proposals to provide mutual authentication of entities (users, sensors, peripherals) and application deploy exists in other computing areas (wireless sensor networks, WBAN, multimedia services, cloud computing, etc.), however, according to the literature review, these issues have scarce attention in $MCSoC$ research area.

The recent work Sepúlveda et al. [26] deals partially this issue presenting a MAC mechanism based on PUF to protect the memory from data forgery. The application admission protection is not directly addressed since that the paper doesn't discuss the authentication of the entities and the secure transfer of the object code to the memory, focusing on the protection during the memory access provided by the MAC scheme, that includes the tasks object code.

Table II. State-of-the-art summary.

Proposal	Protection Comput.	Protection Comm.	Method	Prevent - Provide	Design Time	Runtime
Sepúlveda (2018) [26]	No	Yes	PUF and MAC	Access control; Data integrity and authenticity	–	Key regeneration MAC verification
Rajesh (2015) [29]	No	Yes	Firewall	Access control; Confidentiality	–	Traffic monitoring and rules activation
Hu (2015) [17]	No	Yes	Firewall	Access control; Confidentiality	Application mapping; Firewall positioning and rules configuration	–
Sepúlveda (2015) [65]	No	Yes	Routing Scheme Random arbitration	Timming SCA ; and DoS prevention	–	arbitration obfuscation traffic monitoring and routing change
Fernandes (2016) [31]	No	Yes	Secure Zone: table-routing algorithm (RBR/SBR)	Timing SCA and DoS attacks mitigation;	Runs RBR and SBR algorithms to mapping tasks, calculate paths and router tables	–
Wassel (2014) [37]	No	Yes	Temporal Network Partition	Timing SCA and DoS attacks mitigation;	Task mapping and configuration of static schedule arbitration	–
Ancajas (2014) [28]	No	Yes	Encryption, Authentication Obsfucation	Hardware trojan attack; Confidentiality and authentication	–	Encryption, table-based authentication, taks migration
Boraten (2016) [49]	No	Yes	Packet validation	Hardware trojan attack; Data integrity	–	Change operation mode; generate and verify coded packets
Silva (2017) [47]	No	Yes	Encryption	Data Confidentiality; Authentication	–	Key distribution; Cipher and decipher messages
Kinsy (2017) [12]	No	Yes	Firewall; Encryption	Memory access control; Data leakage and integrity; DoS	Secure wards arrangement	Key distribution; Cipher and decipher messages
Oliveira (2018) [33]	No	Yes	Firewall; Encryption	Data Confidentiality	–	Key distribution; Cipher and decipher messages
Real (2018) [53]	Yes	No	Secure Zone: spatial and temporal isolation	Data integrity; Confidentiality inside the Cluster	Application Priority levels	Application mapping; SZ creation;
ARM (2008) [50]	Yes	No	Secure Zone: spatial and logical isolation	Access control; Data integrity;	Application development using ATZ API	Switch to TTE mode and police execution
Isakovic (2013) [30]	Yes	Yes	Secure Zone: Spatial isolation and encryption	Access control; Authentication; Data integrity	Secure Kernel (keys and secure channel management) and secure mechanisms (protocols, algorithms) provided	Secure channels creation; key exchange; Firewall rules execution
Sepúlveda (2017) [32]	Yes	Yes	Discontinuous Secure Zone: cryptography and firewalls	Access control; Authentication; Confidentiality	–	Key exchange; cipher and decipher messages
Caimi (2019) [56]	Yes	Yes	ECDH; Opaque Secure Zones; Rerouting; Symmetric Encryption	DoS; Spoofing; Data integrity; Authentication; Timming attacks; MitM	Cluster size	Entities authentication; Application admission; Opaque Secure Zones; Rerouting; Peripheral data encryption
Ruaro (2020) [57]	Yes	Yes	Spatial Isolation SDN (circuit switching)	DoS; Spoofing; Data integrity; Authentication	Multiple-Physical NoC (MPN)	Entities authentication Application admission; SDN establishment
Reinbrecht (2017) [21]	No	Yes	Routing Scheme	Timing SCA;	–	Find baseline communication level; Communication monitoring; Change routing police
Grammatikakis (2015) [62]	No	Yes	Firewall	DDoS; Data leakage;	–	Firewall rules configuration and execution

The state-of-the art showed that during the application execution phase, most works related to the security of $MCSoCs$ protect only the communication subsystem. The main mechanisms used are firewalls, temporal network partition, routing schemes and secure zones. Table II summarizes the characteristics of state-of-the-art proposals. The first column presents the primary author, year and reference paper of proposed mechanism. The second and third columns indicate if the proposed mechanism protects the communication and/or computation. The fourth column presents the protection mechanism. The fifth column shows which attack or malicious behavior is mitigated by the proposed mechanism. The sixth and seventh columns summarize what happens at design-time and runtime for each proposed mechanism. Table II also has three blocks, each one presenting works related to the protection of different application lifetime phase (admission, execution, peripheral access).

Several works [17, 31, 37] adopt design time methods. Methods deployed at design time enable the adoption of sophisticated and robust algorithms to provide solutions to the security problem since they do not have limitations related to the computation time of the heuristics. However, design time methods are not applicable in dynamic workload scenarios. Thus, these methods are limited to scenarios where the workload is known beforehand, without any change during the life cycle of the system. In the review, only [37] addresses this issue suggesting that their proposed method can be used in aerospace or medical devices. According to the authors, these fields require high performance, security and have static workloads.

The most common and intuitive approach to protect communication refers to encryption mechanisms. The review shows the use the AES modules incorporated to the NI, ciphering and deciphering the message flow in works such as [12, 33, 47] or, less robust encryption modules using XOR logic gates [28, 32]. This approach provides data confidentiality but still expose the traffic to DoS and timing SCA attacks. Firewall and TNP (Temporal Network Partition) try mitigate this issues.

The use of firewalls [20, 27, 29] ensure access control to the communication system, avoiding DoS attacks and minimizing the possibility of data extraction by a malicious process. Wassel et al. [37] and Wang et al. [38] use TNP to provide temporal and logical traffic isolation avoiding the interference on secure flows, enabling communication availability and timing SCA attacks protection.

The works offering protection to computation [19, 50] or protecting computation and communication simultaneously [30, 32] adopts temporal, logical or spacial isolation as main mechanism.

The isolation enables the creation of secure zones. According to review, secure zones are defined at design time [31] or at runtime [30, 32]. The techniques to create secure zones include encryption [32], the routing algorithm [31], firewalls [12] or, spatial isolation [53]. The proposed mechanisms found in the literature implement secure zones providing logical [32, 34] or spatial isolation [31, 53], although still sharing the communication resources, remaining vulnerable to attacks such as, DoS, timing SCA, HT or data eavesdropping, depending on the mechanism used. The most comprehensive work [56] protects simultaneously computation and communication, using opaque secure zones.

In the reviewed works, just Isakovic et al. [30] discuss the needs of an explicit partitioning in the prevent resources. According to the Authors the application level don't need to implement mitigating resources directly. These mechanisms must be implemented at hardware and microkernel level to be used by the application level.

The protection of the application communication with peripherals and shared memories avoids unauthorized access to instructions and data, which may also compromise the resources availability, due to DoS attacks, cache-based SCA attacks or tampering. Mechanisms to mitigate these threats employ techniques such as firewalls [20], routing scheme [21] and MAC [26].

Table III. Cost overhead of the proposals.

Proposal	Overhead			
	area	power	latency	other
Sepúlveda (2018) [26]	152%	–	–	performance: 1% to 9%
Ancajas (2014) [28]	9.9%	5.8%	5.8%	–
Rajesh (2015) [29]	12.7%	9.8%	5.4%	–
Sepúlveda (2015) [36]	11.0%	9.0%	–	–
Fernandes (2016) [31]	–	–	–	routing tables: 15.56%
Silva (2017) [47]	233.0%	–	700% to 1760%	–
Kinsky (2017) [12]	17.0%	–	9.0%	–
Real (2018) [53]	–	–	–	execution-time: 35.8%
Sepúlveda (2017) [32]	20.0%	12.7%	30% to 55%	–
Oliveira (2018) [33]	193%	–	126.3% to 395.9%	–
Grammatikakis (2015) [20]	–	-30%	-20.5%	–
Reinbrecht (2017) [21]	21.0%	16.2%	–	–

Table III presents the overhead of distinct proposals (first column) related to the area (second column), power consumption (third column) and latency (fourth column). The fifth column shows another overhead parameter eventually presented by the schemes.

Even though all proposed mechanisms use $MCSoC$, according to [66] a comparative cost analysis is hard to make. This is the case for the performance, area, and power consumption evaluation over distinct platforms and the comparison only against a baseline architecture of one's own work. Also, the baseline system adopted for each work and the workload varies. As an example, [32] uses encryption mechanisms to protect the communication and a dedicated service NoC to key exchange and firewall rules configuration, presenting area overhead of 20%, i.e. less than [21], which use a router logic to generate and detect gossip messages, with 21% of area overhead. These works do not detail the baseline system, making the comparison difficult and imprecise.

The area overheads are within the range of 9.9% up to 233% of the correspondent baseline implementation. In general, methods that protect communication using AES modules present higher overhead, as Silva et al. [47] (233%) and Oliveira et al. (193%). An initial expectation is that adopting encryption methods would imply in highest area and power costs. However, the reviewed proposed mechanisms [28, 32], use only simple XOR-based methods to encrypt the communication. While such strategies imply good results regarding area and power consumption, the papers do not discuss how secure or how strong is this encryption technique.

The power consumption is strongly related to the workload and the method used to mitigate the threats. The evaluation in Grammatikakis et al. [62] show a power consumption reduction of 30% when compared to the system under attack. With the proposed method, several packets are dropped by the firewall in the NI, saving power resources. However, the impact of the firewall with normal operation is not evaluated. Other works that evaluate the impact of the proposed methods in power consumption shows an increase between 5.8% with the packet certification presented in [28] and 16.2% with the gossip method proposed in [21].

A. Final Remarks

As presented in Section I., we argue that the security concerns to run a sensitive application must deal with the application admission, the computational and communications aspects of their execution, and the memory and I/O access. The state-of-the-art review shows that few works present a systemic solution that includes all these aspects.

Most works consider only one of these aspects, limited to the application execution (computation or communication protection) and the memory access. The concern about memory access is considered from the communication point of view. Proposals regarding the application admission and the access to peripherals are scarce in the $MCSoC$ research field. A low-cost protocol for secure application admission and communication with external devices targeting $MCSoCs$ is still an open research problem.

Two recent works present systemic solutions, opaque secure zones (OSZ) [56] and the use of SDN to define paths through circuit-switching [57].

The OSZ proposal is concerned with the three stages of execution, presenting a proposal with lightweight encryption for communication with peripherals. Future works for OSZ include: (i) provide an open standard IO interface, such as AMBA or Wishbone, at the borders of the $MCSoC$, enabling the isolation of internal NoC signals to the peripheral. This feature increases the $MCSoC$ security once the internal signaling used is hidden, the attacks started on malicious peripherals are hindered; (ii) implement a peripheral request manager with arbitration functionality, enabling simultaneous requests to devices, including the retransmission of messages.

The SDN approach does not require the use of continuous regions. Spatial isolation is controlled by mapping, and isolation of data flows controlled by circuit switching. Future work for SDN include: (i) protection of the packet switching network to prevent DoS attacks, since this subnet is not protected; (ii) definition of a method for safely communication with peripherals, since it occurs through packet switching given the limitations of the number of SDN subnets.

Acknowledgements

This work was financed in part by CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico) – grant 309605/2020-2; and CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior), Finance Code 001.

References

[1] K. Popovici, F. Rousseau, A. A. Jerraya, and M. Wolf, *Embedded Software Design and Programming of Multiprocessor System-on-Chip: Simulink and System C Case Studies*. Springer Publishing Company, Incorporated, 290p, 2010.

[2] Mellanox Tecnhlogies, “TILE-Gx72 Processor Overview,” Nov 2018, source: http://www.mellanox.com, Nov. 2018.

[3] A. Sodani, R. Gramunt, J. Corbal, H. S. Kim, K. Vinod, S. Chinthamani, S. Hutsell, R. Agarwal, and Y. C. Liu, “Knights Landing: Second-Generation Intel Xeon Phi Product,” *IEEE Micro*, vol. 36, no. 2, pp. 34–46, 2016.

[4] Oracle, “Oracle’s SPARC T8 and SPARC M8 Server Architecture,” Oracle Corporation, Tech. Rep., 2017.

[5] B. D. D. Dinechin, D. V. Amstel, M. Poulhiès, and G. Lager, “Time-critical computing on a single-chip massively parallel processor,” in *Design, Automation Test in Europe Conference (DATE)*, 2014, pp. 1–6.

[6] B. Bohnenstiehl, A. Stillmaker, J. Pimentel, T. Andreas, B. Liu, A. Tran, E. Adeagbo, and B. Bass, “A 5.8 pJ/Op 115 billion ops/sec, to 1.78 trillion ops/sec 32nm 1000-processor array,” in *IEEE Symposium on VLSI Circuits (VLSIC)*, 2016, pp. 1–2.

[7] O. Peckham, “Esperanto Unveils ML Chip with Nearly 1,100 RISC-V Cores,” DEC 2020, source: https://www.hpcwire.com/2020/12/08/esperanto-unveils-ml-chip-with-nearly-1100-risc-v-cores.

[8] Intel, “Intel Core i9-7980XE Extreme Edition Processor,” Nov 2018, source: https://www.intel.com/content/www/us/en/products/processors/core/x-series/i9-7980xe.html.

[9] S. Baron, M. S. Wangham, and C. A. Zeferino, “Security mechanisms to improve the availability of a Network-on-Chip,” in *IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*, 2013, pp. 609–612.

[10] J. Ramachandran, *Designing Security Architecture Solutions*. John Wiley & Sons, Inc., 483p, 2002.

[11] M. M. Real, V. Migliore, V. Lapotre, and G. Gogniat, “ALMOS Many-Core Operating System Extension with New Secure-Enable Mechanisms for Dynamic Creation of Secure Zones,” in *Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP)*, 2016, pp. 820–824.

[12] M. A. Kinsy, S. Khadka, M. Isakov, and A. Farrukh, “Hermes: Secure heterogeneous multicore architecture design,” in *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2017, pp. 14–20.

[13] C. Reinbrecht, B. Forlin, A. Zankl, and J. Sepúlveda, “Earthquake — A NoC-based optimized differential cache-collision attack for MP-SoCs,” in *Design, Automation Test in Europe Conference (DATE)*, 2018, pp. 648–653.

[14] O. Hanka and H. Wippel, “Secure deployment of application-tailored protocols in future networks,” in *International Conference on the Network of the Future (NoF)*, 2011, pp. 10–14.

[15] C. Kuntze, N.; Rudolph, “Secure deployment of SmartGrid equipment,” in *IEEE Power Energy Society General Meeting (PESGM)*, 2013, pp. 1–5.

[16] N. Khernane, M. Potop-Butucaru, and C. Chaudet, “BANZKP: a Secure Authentication Scheme Using Zero Knowledge Proof for WBANs,” in *International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, 2016, pp. 307–315.

[17] Y. Hu, D. Müller-Gritschneder, M. J. Sepulveda, G. Gogniat, and U. Schlichtmann, “Automatic ILP-based Firewall Insertion for Secure Application-Specific Networks-on-Chip,” in *Workshop on Interconnection Network Architectures: On-Chip, Multi-Chip (INA-OCMC)*, 2015, pp. 9–12.

[18] J. Sepúlveda, D. Flórez, and G. Gogniat, “Reconfigurable security architecture for disrupted protection zones in NoC-based MPSoCs,” in *Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*, 2015, pp. 1–8.

[19] M. M. Real, P. Wehner, V. Migliore, V. Lapotre, D. Göhringert, and G. Gogniat, "Dynamic spatially isolated secure zones for NoC-based many-core accelerators," in *Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*, 2016, pp. 1–6.

[20] M. D. Grammatikakis, P. Petrakis, A. Papagrigoriou, G. Kornaros, and M. Coppola, "High-level security services based on a hardware NoC Firewall module," in *Workshop on Intelligent Solutions in Embedded Systems (WISES)*, 2015, pp. 73–78.

[21] C. Reinbrecht, A. Susin, L. Bossuet, G. Sigl, and J. Sepúlveda, "Timing attack on NoC-based systems: Prime+Probe attack and NoC-based protection," *Microprocessors and Microsystems (MICPRO)*, vol. 52, no. C, pp. 556–565, 2017.

[22] C. H. Gebotys and R. J. Gebotys, "A framework for security on NoC technologies," in *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2003, pp. 113–117.

[23] A. Avizienis, J. C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 1, pp. 11–33, 2004.

[24] J. Coburn, S. Ravi, A. Raghunathan, and S. Chakradhar, "SECA: Security-enhanced Communication Architecture," in *IEEE International Conference on Compilers, Architectures and Synthesis for Embedded Systems (CASES)*, 2005, pp. 78–89.

[25] S. Evain and J. P. Diguet, "From NoC security analysis to design solutions," in *IEEE Workshop on Signal Processing Systems Design and Implementation (SiPS)*, 2005, pp. 166–171.

[26] J. Sepúlveda, F. Willgerodt, and M. Pehl, "SEPUFSoC: Using PUFs for Memory Integrity and Authentication in Multi-Processors System-on-Chip," in *Great Lakes Symposium on VLSI (GLSVLSI)*, 2018, pp. 39–44.

[27] D. He, J. Chen, and Y. Chen, "A secure mutual authentication scheme for session initiation protocol using elliptic curve cryptography," *Security and Communication Networks*, vol. 5, no. 12, pp. 1423–1429, 2012.

[28] D. M. Ancajas, K. Chakraborty, and S. Roy, "Fort-NoCs: Mitigating the threat of a compromised NoC," in *ACM/IEEE Design Automation Conference (DAC)*, 2014, pp. 1–6.

[29] J. Rajesh, D. M. Ancajas, K. Chakraborty, and S. Roy, "Runtime Detection of a Bandwidth Denial Attack from a Rogue Network-on-Chip," in *IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*, 2015, pp. 8:1–8:8.

[30] H. Isakovic and A. Wasicek, "Secure channels in an integrated MP-SoC architecture," in *Industrial Electronics Society (IECON)*, 2013, pp. 4488–4493.

[31] R. Fernandes, C. Marcon, R. Cataldo, J. Silveira, G. Sigl, and J. Sepúlveda, "A security aware routing approach for NoC-based MPSoCs," in *Symposium on Integrated Circuits and Systems Design (SBCCI)*, 2016, pp. 1–6.

[32] J. Sepúlveda, D. Flórez, V. Immler, G. G., and G. Sigl, "Efficient security zones implementation through hierarchical group key management at NoC-based MPSoCs," *Microprocessors and Microsystems (MICPRO)*, vol. 50, pp. 164 – 174, 2017.

[33] B. Oliveira, R. Reusch, H. Medina, and F. G. Moraes, "Evaluating the Cost to Cipher the NoC Communication," in *IEEE Latin American Symposium on Circuits and Systems (LASCAS)*, 2018, pp. 1–4.

[34] G. Sharma, . S. Ellinidou, R. Anand, V. Kuchta, O. Markowitch, and J. Dricot, "Secure Communication on NoC based MPSoC," in *International Conference on Security and Privacy in Communication Networks (SecureComm)*, 2018, p. 12.

[35] L. L. Caimi, V. Fochi, and F. G. Moraes, "Secure Admission of Applications in Many-Cores," in *IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, 2018, pp. 761–764.

[36] J. Sepúlveda, J. P. Diguet, M. Strum, and G. Gogniat, "NoC-Based Protection for SoC Time-Driven Attacks," *IEEE Embedded Systems Letters (ESL)*, vol. 7, no. 1, pp. 7–10, 2015.

[37] H. M. G. Wassel, Y. Gao, J. K. Oberg, T. Huffmire, R. Kastner, F. T. Chong, and T. Sherwood, "Networks on Chip with Provable Security Properties," *IEEE Micro*, vol. 34, no. 3, pp. 57–68, 2014.

[38] Y. Wang and G. E. Suh, "Efficient Timing Channel Protection for On-Chip Networks," in *IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*, 2012, pp. 142–151.

[39] NaNoC, "The NaNoC project," Nov 2018, source: http://www.nanoc-project.eu/.

[40] S. Azad, B. Niazmand, G. Jervan, and J. Sepulveda, "Enabling Secure MPSoC Dynamic Operation through Protected Communication," in *ICECS*, 2019, pp. 481–484.

[41] S. Payandeh Azad, G. Jervan, M. Tempelmeier, and J. Sepulveda, "CAESAR-MPSoC: Dynamic and Efficient MPSoC Security Zones," in *ISVLSI*, 2019, pp. 477–482.

[42] S. Charles, M. Logan, and P. Mishra, "Lightweight Anonymous Routing in NoC based SoCs," in *DATE*, 2020, pp. 334–337.

[43] L. Indrusiak, J. Harbin, C. Reinbrecht, and J. Sepúlveda, "Side-channel protected MPSoC through secure real-time networks-on-chip," *Microprocessors and Microsystems*, vol. 68, pp. 34–46, 2019.

[44] NASA, "Numerical Aerodynamic Simulation - NAS," Nov 2015, source: http://www.nas.nasa.gov/publications/npb.htm, Nov. 2018.

[45] H. M. G. Wassel, Y. Gao, J. K. Oberg, T. Huffmire, R. Kastner, F. T. Chong, and T. Sherwood, "SurfNoC: A Low Latency and Provably Non-interfering Approach to Secure Networks-on-chip," in *International Symposium on Computer Architecture (ISCA)*, 2013, pp. 583–594.

[46] R. Churchhouse, *Codes and Ciphers: Julius Caesar, the Enigma, and the Internet*. Cambridge University Press, 240p, 2002.

[47] M. R. Silva and C. A. Zeferino, "Confidentiality and Authenticity in a Platform Based on Network-on-Chip," in *Brazilian Symposium on Computing Systems Engineering (SBESC)*, 2017, pp. 225–230.

[48] A. C. Sant'Ana, H. Medina, and F. G. Moraes, "Security Vulnerabilities and Countermeasures in MPSoCs," *IEEE Design Test*, vol. *preprint*, pp. 1–7, 2021. [Online]. Available: https://doi.org/10.1109/MDAT.2021.3049710

[49] T. Boraten and A. K. Kodi, "Packet security with path sensitization for NoCs," in *Design, Automation Test in Europe Conference (DATE)*, 2016, pp. 1136–1139.

[50] ARM, "ARM Security Technology: Building a Secure System using TrustZone Technology," Nov 2018, source: http://infocenter.arm.com, Nov. 2018.

[51] C. W. Fletcher, M. Dijk, and S. Devadas, "A Secure Processor Architecture for Encrypted Computation on Untrusted Programs," in *ACM Workshop on Scalable Trusted Computing (WSTC)*, 2012, pp. 3–8.

[52] C. Hoffman, M. Cortes, D. F. Aranha, and G. Araujo, "Computer security by hardware-intrinsic authentication," in *Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, 2015, pp. 143–152.

[53] M. M. Real, P. Wehner, V. Lapotre, D. Göhringer, and G. Gogniat, "Application Deployment Strategies for Spatial Isolation on Many-Core Accelerators," *ACM Transaction on Embedded Computing Systems*, vol. 17, no. 2, pp. 55:1–55:31, 2018.

[54] P. Wehner, J. Rettkowski, T. Kleinschmidt, and D. Göhringer, "MPSoCSim: An extended OVP simulator for modeling and evaluation of Network-on-Chip based heterogeneous MPSoCs," in *Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS)*, 2015, pp. 390–395.

[55] C. E. Salloum, M. Elshuber, O. Höftberger, H. Isakovic, and A. Wasicek, "The ACROSS MPSoC – A new generation of multi-core processors designed for safety–critical embedded systems," *Microprocessors and Microsystems (MICPRO)*, vol. 37, no. 8, pp. 1020 – 1032, 2013.

[56] L. L. Caimi and F. G. Moraes, "Security in Many-Core SoCs Leveraged by Opaque Secure Zones," in *ISVLSI*, 2019, pp. 471–476. [Online]. Available: https://doi.org/10.1109/ISVLSI.2019.00091

[57] M. Ruaro, L. L. Caimi, and F. G. Moraes, "SDN-Based Secure Application Admission and Execution for Many-Cores," *IEEE Access*, vol. 8, pp. 177 296–177 306, 2020.

[58] C. Reinbrecht, A. Aljuffri, S. Hamdioui, M. Taouil, B. Forlin, and J. Sepulveda, "Guard-NoC: A protection against side-channel attacks for MPSoCs," in *ISVLSI*, 2020, pp. 536–541.

[59] C. Reinbrecht, A. Susin, L. Bossuet, and J. Sepúlveda, "Gossip NoC - Avoiding Timing Side-Channel Attacks through Traffic Management," in *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2016, pp. 601–606.

[60] C. Reinbrecht, A. Susin, L. Bossuet, G. Sigl, and J. Sepúlveda, "Side channel attack on NoC-based MPSoCs are practical: NoC Prime+Probe attack," in *Symposium on Integrated Circuits and Systems Design (SBCCI)*, 2016, pp. 1–6.

[61] D. A. Osvik, A. Shamir, and E. Tromer, "Cache Attacks and Countermeasures: The Case of AES," in *Topics in Cryptology*, 2006, pp. 1–20.

[62] M. D. Grammatikakis, K. Papadimitriou, P. Petrakis, A. Papagrigoriou, G. Kornaros, I. Christoforakis, O. Tomoutzoglou, G. Tsamis, and M. Coppola, "Security in MPSoCs: A NoC Firewall and an Evaluation Framework," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 8, pp. 1344–1357, 2015.

[63] K. Papadimitriou, P. Petrakis, M. D. Grammatikakis, and M. Coppola, "Security Enhancements for building saturation-free, low-power NoC-based MPSoCs," in *IEEE Conference on Communications and Network Security (CNS)*, 2015, pp. 594–600.

[64] M. D. Grammatikakis, K. Papadimitriou, P. Petrakis, A. Papagrigoriou, G. Kornaros, I. Christoforakis, and M. Coppola, "Security Effectiveness and a Hardware Firewall for MPSoCs," in *IEEE High Performance Computing and Communications (HPCC)*, 2014, pp. 1032–1039.

[65] J. Sepúlveda, G. Gogniat, D. Flórez, J. P. Diguet, R. Pires, and M. Strum, "TSV protection: Towards secure 3D-MPSoC," in *IEEE Latin American Symposium on Circuits Systems (LASCAS)*, 2015, pp. 1–4.

[66] T. Bjerregaard and S. Mahadevan, "A Survey of Research and Practices of Network-on-chip," *ACM Computing Surveys (CSUR)*, vol. 38, no. 1, pp. 1–51, 2006.