Towards NoC Protection of HT-Greyhole Attack

Soultana Ellinidou*[1][0000−0001−6843−9253], Gaurav Sharma[1], Olivier Markowitch[1], Jean-Michel Dricot[1], and Guy Gogniat[2]

[1] Cybersecurity Research Center, Université Libre de Bruxelles, Brussels, Belgium
{soultana.ellinidou, gsharma, olivier.markowitch, jdricot}@ulb.ac.be
[2] Lab-STICC, Université de Bretagne Sud, Lorient, France
guy.gogniat@univ-ubs.fr

Abstract. As the number of processing cores is increasing dramatically, the communication among them is of high importance. Network-on-Chip (NoC) has direct access to all resources and information within a System-on-Chip (SoC) by rendering it appealing to attackers. In this paper a novel Hardware Trojan (HT) assisted Denial of Service (DoS) attack, called Greyhole attack is introduced. The HT-Greyhole attack targets the routers within NoC by causing performance decrease and packet loss increase. However, during an HT-Greyhole attack certain packets, which are arriving towards the router, are dropped which makes it hard to detect. In this paper, we design a detection and defense method, against HT-Greyhole attack, which is based on Software Defined Network-on-Chip (SDNoC) architecture. The results demonstrate that by using the proposed defense method the packet loss decreases by 76% under Transpose traffic, 77% under BitReverse and 50% under Uniform traffic.

Keywords: Hardware Trojans · Network-on-Chip · Greyhole attack · HT DoS attacks · SDNoC

1 Introduction

Malicious hardware modifications at different stages of its life cycle create major security concerns in the field of electronics. The Hardware Trojan (HT) attacks emerged as a major security threat for Intellectual Properties (IPs) blocks, Integrated Circuits (ICs), Printed Circuit Boards (PCBs), and System-on-Chip (SoC). Specifically, these attacks introduce a malicious modification of a circuit during the design or fabrication process in an untrusted design house or foundry, in which untrusted people, design tools, or components are involved [1]. Such modifications can lead to abnormal functional behavior of a system, degrade performance and provide covert channels or backdoors by which an attacker can leak sensitive information.

Since the number of processors and cores on a single chip is increasing, the interconnection between them becomes significant. A key challenge is to provide secure and reliable communication in the SoC, even in the case of an untrusted

* Corresponding Author: soultana.ellinidou@ulb.ac.be.

Network-on-Chip (NoC) IP inserted into it. Since the NoC has direct access to all communication resources and information flow within the SoC, attackers have a strong motivation to exploit its possible vulnerabilities. In recent literature, a vast number of HT attacks, which are mainly focusing on NoC, have been introduced [2, 3, 4]. As far as the hardware methods for the detection and defense of the HT-attacks targeting NoC, it is observed that most of them are employed in the Network Interfaces (NIs) [2], which connect the IP cores and routers, some of them on the links between routers [5] and very few on the routers [3].

Following the literature, the common assumption is that a NoC is supplied to a SoC integrator and there is a possibility that it is already compromised with a HT [6]. In order to activate the HT, a malicious circuit has already been inserted during the design time of the IP block and a malicious program can activate it later at runtime. The possible attacks due to infected NoC IP block are:

- **Snooping**: In this case, illegal monitoring is performed by an untrusted router within the path, which tracks the number of packets between source and destination IP cores.
- **Corruption of the data**: A malicious router can modify the content of the incoming flits and the route of the packets.
- **Spoofing**: A malicious router copies and replays packets, which may lead to the malfunction or eviction of sensitive data.
- **Denial of service (DoS)**: The denial or distributed denial of service can make the resources unavailable to legitimate IPs.

In this paper, we consider a specific HT assisted DoS attack, called the Greyhole attack, which targets the routers of a NoC within a SoC. The Greyhole attack is a well known attack from Wireless Sensor Networks (WSN) [7, 8]. In case of a Greyhole attack, a malicious router blocks certain packets from its neighboring routers instead of forwarding them. Hence, critical packets, that are forwarded to a Greyhole router, are captured and can not arrive to their destinations. In order to detect and mitigate a malicious router within a network, some of the security mechanisms encountered in the literature are: data partitioning, key management, key generation, localization and trust management [8].

However, despite the large amount of research contributions in WSN about the Greyhole attack, this attack has not been introduced in the field of electronics and more specifically in NoC context. Hence our main contributions are summarized as follows:

- the description and activation of an HT-Greyhole attack in NoC context,
- the exploration of Software Define Network on Chip (SDNoC) as a potential solution for NoC protection,
- a security management mechanism relying on SDNoC, as key proposal in order to identify malicious routers,
- depending on the position of affected routers, a route exclusion approach is presented in order to mitigate the impact of the attack.

SDNoC provides secure paths in presence of untrusted routers and assures that the packets will be successfully delivered to their destination.

The rest of the paper is organized as follows: In Section 2 we discuss the related work. Thereafter, in Section 3 the SDNoC concept is presented. In Section 4 the launching of HT-Greyhole attack is introduced, followed by the detection strategy and the defense approach, described in Section 5 and Section 6 respectively. The evaluation results are discussed in Section 7, followed by conclusion and future work in Section 8.

2 Related work

There is no existing literature on HT-Greyhole attacks, however since Greyhole attacks are variants of Blackhole attacks, the related literature of HT-Blackhole attacks is presented. In 2018 the HT-Blackhole attack targeting the NoCs was introduced [9], the authors investigate not only the Blackhole but also the Sinkhole attack in the context of NoC. Specifically, they focus on the effects of the attack by measuring the packet loss rate, considering the number of HTs and their distribution in NoC. They provide a theoretical detection method, where a global manager injects detection request packets to randomly selected routers in order to find the suspicious one. Though the main disadvantage of the detection method is that it can only detect HTs which are always on trigger mode. A defense method is also presented, where each router keeps a record of neighbors, which is updated by the global manager and needs to be checked by the router itself before taking routing decisions.

Afterwards, in [10], an analysis of the HT-Blackhole attack considering the area and power overhead of the malicious router was presented. Precisely, the authors presented the influence of the number of HT-Blackhole routers along with their distribution in the NoC. Another contribution by the same authors is presented in [11], where they proposed a secure protocol with runtime detection and protection of HT-Blackhole attack. The proposed secure protocol protects the system from HT-Blackhole attacks, but it increases dramatically the overhead due to the large number of acknowledgment (ACK) packets that need to be exchanged between the routers for each data packet transmission from a source to a destination router.

3 SDNoC

The launching part can individually be performed on different NoC architectures. However, for our defense and detection approach we target a NoC alternative, called SDNoC [12, 13, 14]. The SDNoC concept is inspired by the well known Software Defined Networking (SDN) technology [15]. The main idea of SDNoC is to minimize router complexity by exporting the routing logic to a centralized controller which has a general view of the network and can take routing decisions efficiently. According to the authors of [12, 14], SDNoC could possibly be adaptable for SoCs thanks to its advantages: 1) it reduces the hardware complexity, 2)

it has high re-usability and 3) it has flexible management of communication policies. However, there are also some challenges that should be taken into account, in particular the high overhead for path selection in software against hardware based approaches and the centralized controller can be a single point of failure.

The only difference between SDNoC and NoC is that the SDNoC manages routing in an adaptive manner with the help of a centralized controller. In Fig. 1, an SDNoC architecture is depicted, which consists of 16 routers and a centralized controller. The routers are interconnected with the neighboring routers through links and with a Processing Element (PE) through the NI. The centralized controller is interconnected with the routers through direct links, as depicted in Fig. 1. Furthermore, it sends configuration packets and manages the routing of the packets in an efficient manner.

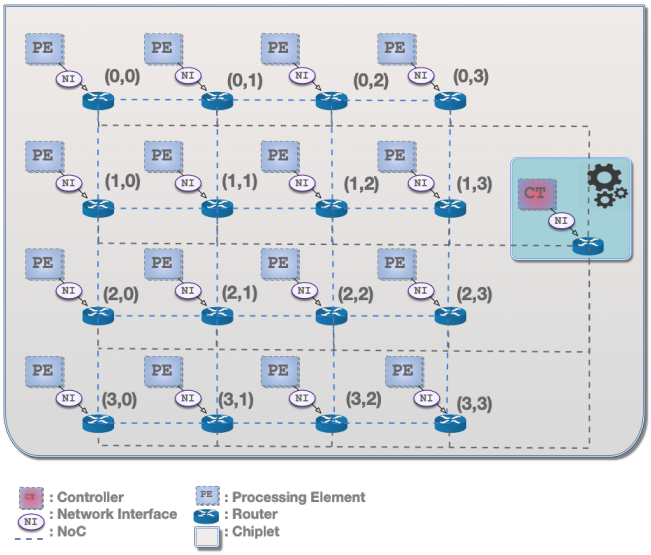


Fig. 1: Software Defined Network-on-Chip (SDNoC) architecture

4 Launching of HT-Greyhole Attack

HTs can be inserted into the pipeline of a Virtual Channel (*VC*) router according to [16] and at each input port of a router. The main HT is placed on the *VC Allocator* and the other HTs are synchronized with it through a control signal [9]. A HT structure consists of three modules: the Trigger, the Configuration and the Greyhole function module. The placement of the HT-Greyhole in a NoC router is shown in Fig. 2, where specifically a malicious HT-Greyhole router architecture is illustrated. The router consists of 5 input/output ports, 5 *VC*s, 5

buffers with a counter (C), a *VC allocator*, a *Crossbar switch*, a *Switch allocator*, a *TrustTable* and *Flow Tables*. The five ports correspond to the four cardinal directions and the local direction which connects the router with the PE through the NI. The router employs a pipelined design with speculative path selection to improve performance. The router consists of a two-stage, pipelined architecture. The first stage is responsible for routing and the second stage is responsible for crossbar traversal. In this work, the functionality of the router is described with respect to a 2D mesh interconnect. A HT is placed in each input port: South, North, East, West, Local, which are synchronized with the main HT-Greyhole which is placed on the *VC allocator*. More details for the HT insertion in one port in a single cycle VC-based router can be found in [17].

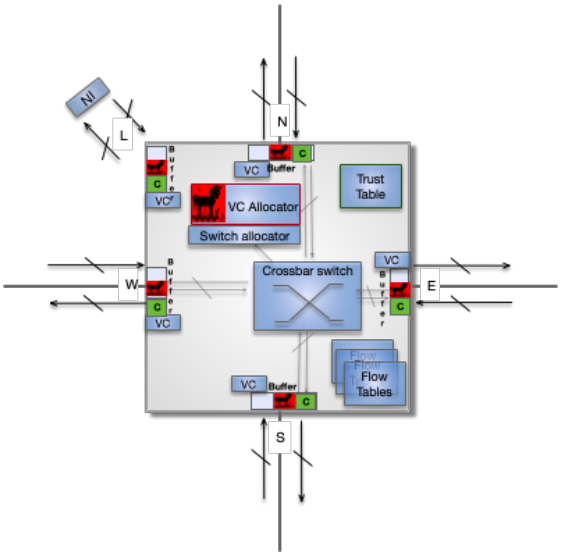


Fig. 2: HT-Greyhole router.

Before an attack is launched, a configuration packet should be sent to the target router by an attacker through a malicious program running on the given IP core connected to the router. The configuration packet, which is depicted in Fig. 3, consists of the following fields:

- **Config cmd**: is the field of a packet that consists of a specific bit pattern (e.g. 00110101), which states as a HT configuration packet.
- **Trigger**: has 2 modes: Always Activated (AA) and Destination Based (DB). An AA trigger HT is always active, while a DB trigger is activated only when the destination ID of an incoming packet is identical with the target ID of the configuration packet.
- **Packet Type**: declares the type of packet which is either signal or data packet.

– **Activation Signal:** could be on or off depending on the activation of the HT.
– **Target ID**: specifies the target address for the DB trigger.
– **Interceptor ID**: in case that an HT is launched, every data packet Destination ID will be replaced with the Interceptor ID.

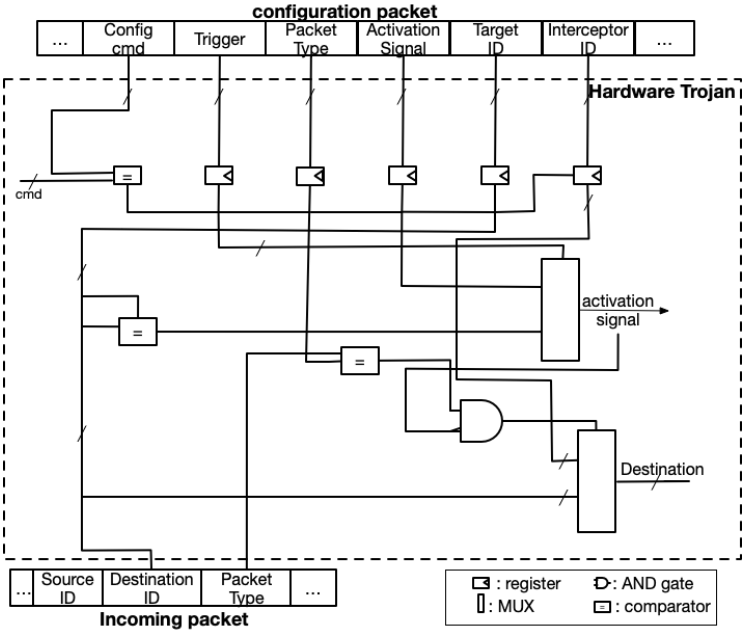


Fig. 3: HT design on circuit level.

After the configuration packet has been delivered to the target router, the HT configuration information will be saved in a set of registers (Fig. 3). When a HT has been configured, it can be activated by the trigger module. More precisely the steps for launching an attack are the following:

– **Step 1**: An attacker sends a configuration packet through a malicious program to the target router.
– **Step 2**: The HT, placed in the target router, receives the configuration packet and updates its configuration information.
– **Step 3**: The trigger module chooses its mode based on the trigger field data stored in the registers.
– **Step 4**: An activation signal is generated by trigger, by taking into account the trigger mode. As for AA mode an activation signal is generated all the time, while for the DB mode the activation signal is set to on when the

destination ID of the incoming packet matches with the Target ID of the register.
- **Step 5**: The attacker specifies in the configuration packet the type of the packet that needs to be dropped. If the type of the packet matches with the type of the incoming packet then we move to the next step (in our scenario the signal packets are normally processed and the data packets are dropped).
- **Step 6**: Launch the attack according to the signal and the packet type.
- **Step 7**: If the Packet Type is *data* then the Destination ID of the incoming packet will be replaced with the Interceptor ID. If the Packet Type is *signal*, the Destination ID will not be modified.

5 Detection

HT assisted DoS attacks are hard to be detected due to their low silicon footprint, small power and area consumption but also due to their conditional activation during the run time. Specifically in the HT attack presented in [9], the area and the power consumption are 0.07% and 0.02% of a NoC router and in [10] the malicious router area and power increase are 1.98% and 0.74%, respectively.

Our detection strategy has been designed in order to specifically detect a HT-Greyhole attack in the context of SDNoC. Based on our architecture, each router has a counter in each port (Fig. 2), which is incremented every time that a new packet is imported and it is decremented every time that a packet is exported. The results are saved in the *TrustTable*, which includes all the values for each port. The routers are responsible to periodically send the *TrustTable* along with their *RouterID*, to the controller. The controller calculates and chooses the routes for each individual source and destination by storing them in the table *Routes*. The value k indicates the 4 different directions north, east, south, west.

As soon as the controller receives a *TrustTable*, it uses the Algorithm 1 in order to find out which routers are considered as suspects. In the algorithm, the controller checks if any input of the *TrustTable* is less than a threshold value (tv), which value can be chosen depending on traffic pattern or buffer holding capacity. More details about tv value can be found in the Section 7.

Since a malicious router can modify its *TrustTable* and pretend that it is non-malicious, the only option is to be detected through their neighbors. Hence the algorithm searches the previous hop (*neighbor*) of the given *RouterID* and afterwards it clarifies if the direction of the *neighbor* matches with the direction of the port value of *RouterID* within the *TrustTable*. If so then the neighbor is considered as suspect.

Since the controller calculates the table *Suspect* of the given *RouterID*, it will also check the tables *Suspect* of the other *RouterID*'s. If a suspect appears at least in two different *Suspect* tables, because each router could have at least two neighbor routers, the suspect router will be considered as malicious.

The detection method is less costly in terms of overhead and complexity since the control links between routers and controller are utilized and the only router side operation is to calculate a *TrustTable*, which includes the values of the 4 counters (4-bit each), and to send it through the control links to the controller.

Algorithm 1 Detection Algorithm

```
Data: Routes[ ][ ], TrustTable[ ][ ], RouterID, a=0
for k=1:4 do
    if TrustTable[k][2] < tv then
        for j=1:Routes.rows() do
            for t=1:Routes.column() do
                if Routes[j][t] == RouterID then
                    neighbor == Routes[j-1] [t];
                     if TrustTable[k][1] == direction.neighbor() then
                        a=a+1;
                         Suspect[a]=neighbor;
                    end
                end
            end
        end
    end
end
```

6 Defense

As the proposed detection strategy has already identified the malicious routers and their positions, a route exclusion approach is presented in order to mitigate the attack. The controller executes the defense approach which consists of following three phases:

- **Route Exploration Phase:** Given a source and a destination the controller computes a set of admissible routes based on Odd-Even (OE) routing algorithm [18] and it stores them in a table. OE is a turn model routing algorithm, lightweight and deadlock-free. Among the existing turn model routing algorithms OE tends to provide better performance and higher adaptiveness than the others.
- **Untrusted Paths Phase:** From the detection algorithm, the controller already has a list with the malicious routers. Hence, in this phase, it has as input the set of admissible routes from the previous phase, which are checked if they include any malicious router. The routes that include a malicious router are marked as untrusted and the rest of the routes as trusted.
- **Selection Phase:** The inputs in this phase are all the trusted routes from a given source to a destination. In the classic OE routing algorithm, a random route is chosen among the admissible ones. However, in our case the controller chooses the least congested route among the admissible ones by calculating the link load (l_i) of the routes. The l_i corresponds to the number of flits per second that are passing through the link. The designed formula in order to avoid the highly-loaded links and routers within the route is depicted in (1).

$$S = \sum_{i=0}^{L_f} l_i. \quad (1)$$

Where S is the computed score for each admissible trusted route and L_f is the number of links along the route.
Among the scores of the different routes, the route with smallest score value is selected and indeed it represents the least congested route. Note that the controller's knowledge concerning the data network state (via the link load) is gathered by immediate inputs from different routers and their *Trust-Table* computations. Nevertheless for the initial route computation there is no available score, hence a random route is chosen among the admissible ones, offered by OE.

7 Evaluation

In order to perform an attack but also to evaluate our detection and defense strategy, simulations were performed with Garnet2.0 [19], which is a NoC model implementation within the gem5 simulator. The traffic generated by the processing cores is based on the traffic injection rate (tir), which is expressed as the average number of packets injected by the cores into the network per clock cycle ($0 < tir \leq 1$). Each processing core will generate a packet every $1/tir$ clock cycles on average, but the actual time at which the packets are transmitted is random. It should be noted that for each scenario we perform 40 iterations, of which the average value of throughput and packet loss is calculated.

An 8×8 topology is simulated, by taking into account that it contains 1, 3 and 6 HT-Greyhole routers. Furthermore, three different traffic scenarios have been evaluated: Transpose, BitReverse and Uniform.

As far as the evaluation of the detection strategy is concerned, binary classification is used. Binary or binomial classification is the task of classifying the elements of a given set into two groups (predicting which group each one belongs to) on the basis of a classification rule [20]. Considering a two-class prediction problem, where the outcomes are labeled either as Positive (P) or Negative (N). In this case there are four possible outcomes from a binary classifier. If the outcome from a prediction is P and the actual value is also P, then it is called a True Positive (TP); however if the actual value is N then it is said to be a False Positive (FP). Conversely, a True Negative (TN) has occurred when both the prediction outcome and the actual value are N and False Negative (FN) is when the prediction outcome is N, while the actual value is P. These four counts constitute a confusion matrix shown in Table 1.

Table 2 presents the measurements, that we took into account, for binary classification based on the values of confusion matrix. In the context of confusion matrix, Receiver Operating Characteristic (ROC) is a graphical representation plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. ROC graphs are two-dimensional graphs in

Table 1: Confusion Matrix

		True Condition	
Predicted Condition	***Total Population***	**Positive (P)**	**Negative (N)**
	Positive (P)	True Positive	False Positive
	Negative (N)	False Negative	True Negative

which True Positive Rate (TPR) is plotted on the y axis and False Positive Rate (FPR) is plotted on the x axis. When using normalized units, the Area Under the Curve (AUC) is equal to the probability that a classifier will rank a randomly chosen P instance higher than a randomly chosen N one [21]. The Area Under Curve (AUC) measure gives a better view about the algorithms capability of distinguishing between classes. The higher AUC, the better model is at predicting the positive and negative values.

Table 2: Measurements for binary classification

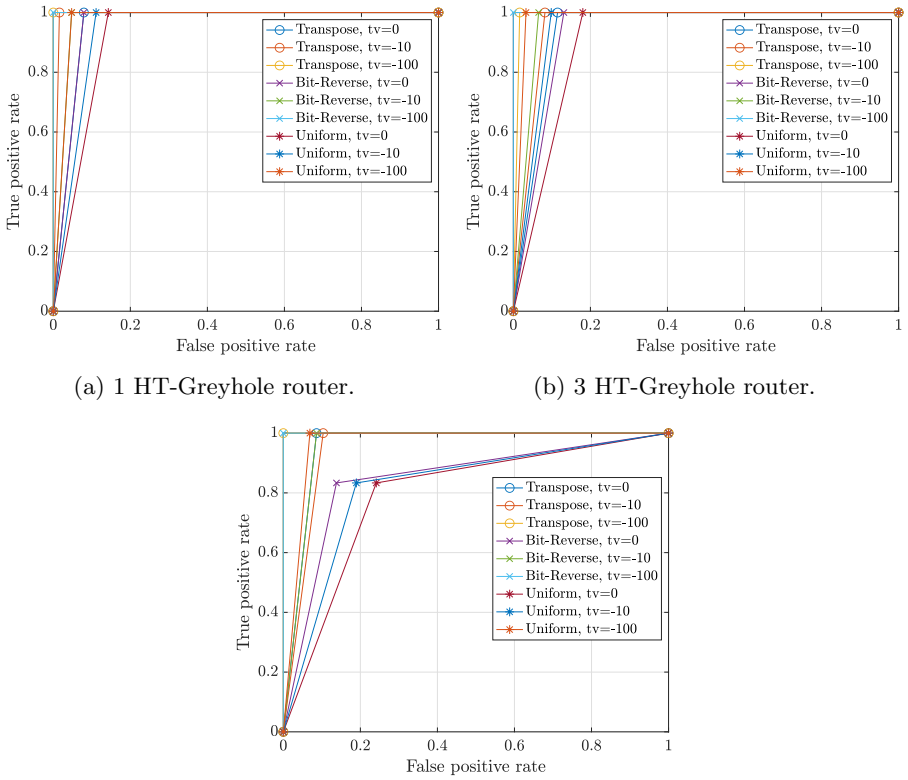
Measurement	**Abbr**	**Formula**	**Explanation**
Sensitivity, Recall or True Positive Rate	TPR	$\frac{TP}{TP+FN}$	Effectiveness of a classifier to identify positive labels.
Miss rate or False Negative Rate	FNR	$\frac{FN}{FN+TP}$	Probability of identifying positive labels as negative.
Accuracy	ACC	$\frac{TP+TN}{TP+TN+FP+FN}$	Overall effectiveness of a classifier.
Area Under the Curve	AUC	$\frac{1}{2}(\frac{TP}{TP+FN} + \frac{TN}{TN+FP})$	Classifier's ability to avoid false classification.

By using binary classification, we identify 27 different scenarios of an 8x8 topology, where we took into account, different traffics (Transpose, BitReverse, Uniform), different number of HT (1, 3, 6) and different value of tv (0, -10, -100) of the detection algorithm. For our scenario a malicious node (HT-Greyhole router), which considered as negative and non-malicious node, which considered as positive, in that setting:

- **TP**: Non-malicious node correctly identified as non-malicious.
- **FP**: Malicious node incorrectly identified as non-malicious.
- **TN**: Malicious node correctly identified as malicious.
- **FN**: Non-Malicious node incorrectly identified as malicious.

Fig. 4 represents the ROC curves of the different scenarios by taking into account the number of HT-Greyhole routers within the network, under different traffic scenarios and different tv values. From the graphs, it is obvious that the ROC curves for the $tv = -100$ tend to be ideal for all scenarios, hence our

algorithm is able to better distinguish between positive and negative values for this threshold value. As fas as the Accuracy (ACC) of the algorithm is concerned, for $tv = 0$ the ACC is between 73.4% and 92.2%, for $tv = -10$ the ACC was between 81.2% and 98.4% and for $tv = -100$ the ACC is between 95.2% and 100%. However it worths to be mentioned that for some test cases for $tv = 0$ and $tv = -10$, we noticed FN values, hence the FPR will be higher. As far as the AUC value is concerned for the $tv = -100$, it is between 0.965 and 1, which means that in some test cases it is perfect (AUC=1) and in other cases tend to be perfect ($0.95 < AUC < 1$).



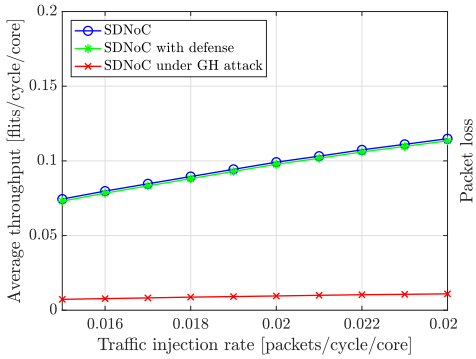
(a) 1 HT-Greyhole router.

(b) 3 HT-Greyhole router.

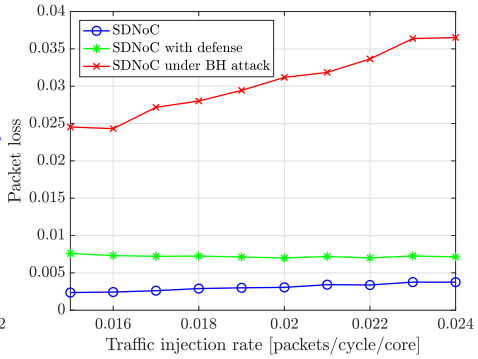
(c) 6 HT-Greyhole router

Fig. 4: Roc curve diagrams for 1, 3, 6 HT-Greyhole routers with tv=0, -10, -100 and for Transpose, BitReverse, Uniform traffic.

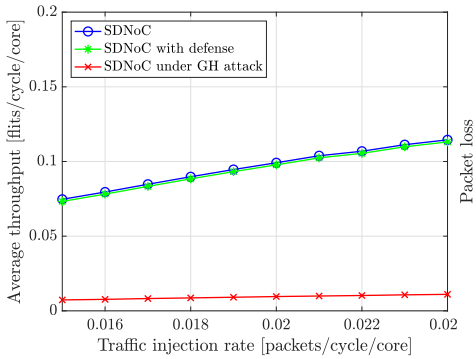
Fig. 5 depicts a scenario of 1 HT-Greyhole router. More precisely, on Fig. 5a, 5c, 5e the average throughput under different injection rates (0.015-0.024) is presented for Transpose, BitReverse and Uniform traffic respectively. In Fig. 5b, 5d, 5f the packet loss rate is shown under different injection rates. From the fig-



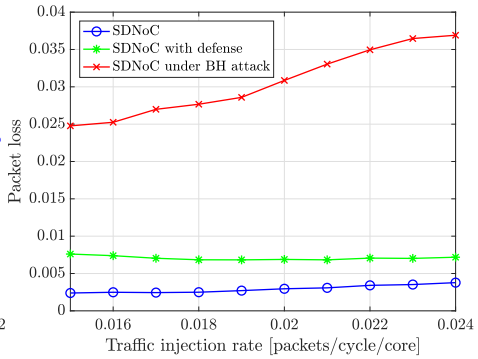
(a) Throughput under Transpose Traffic.



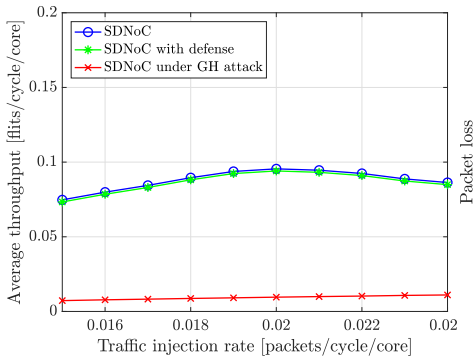
(b) Packet loss under Transpose Traffic.



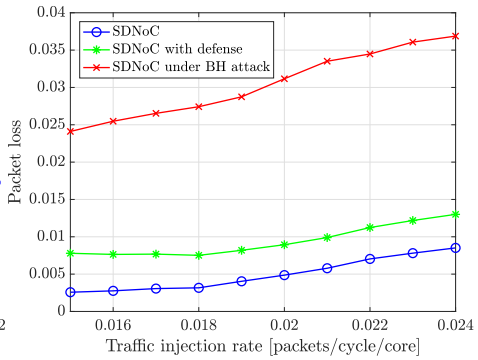
(c) Throughput under BitReverse Traffic.



(d) Packet loss under BitReverse Traffic.



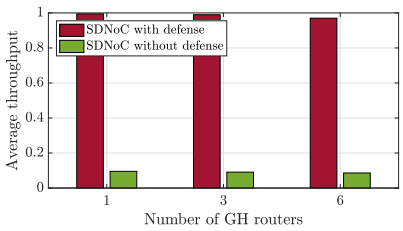
(e) Throughput under Uniform Traffic.



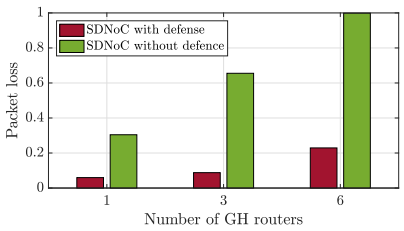
(f) Packet loss under Uniform Traffic.

Fig. 5: Throughput and packet loss graphs for 1 HT-Greyhole router under Uniform, Transpose and BitReverse traffic scenarios.

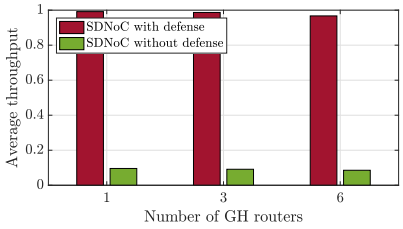
ures it is obvious that there is an increase of the packet loss rate and a decrease of the throughput of the SDNoC when the network is under attack compared to when the network works normally. Furthermore, when our defense part is employed on SDNoC, it is noticeable that the throughput values of SDNoC with defense and the throughput values of normal SDNoC tend to be identical. Precisely, under the higher injection rate we have an increase of 3% under Uniform, Transpose and BitReverse traffics of the overall packet loss rate between SDNoC and SDNoC under GH attack. As far as the average throughput, it is decreased by 8% under Uniform traffic, 10% under Transpose and BitReverse traffic. Thus the detection of this attack is a very difficult process.



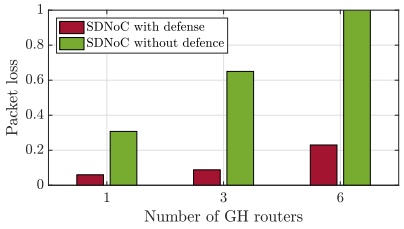
(a) Throughput under Transpose Traffic.



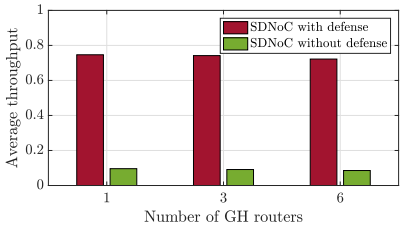
(b) Packet loss under Transpose Traffic.



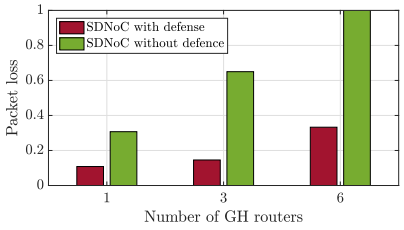
(c) Throughput under BitReverse Traffic.



(d) Packet loss under BitReverse Traffic.



(e) Throughput under Uniform Traffic.



(f) Packet loss under Uniform Traffic.

Fig. 6: Throughput and packet loss graphs for 1, 3, 6 HT-Greyhole routers.

In Fig. 6, three different scenarios are presented in each graph. In the first scenario, only one HT-Greyhole router is considered, second scenario consid-

ers three HT-Greyhole routers and in third instance, there are six HT-Greyhole routers. Fig. 6a 6c 6e depict the *normalized* average throughput under Transpose BitReverse and Uniform traffic respectively. Fig. 6b, 6f, 6d show the *normalized* packet loss rate. By taking into account these results, the packet loss improvement is shown in Table 3. As far as the throughput is concerned, by applying our defense method it improved between 63-66% for Uniform traffic and 88-89% for Transpose and BitReverse traffics.

Table 3: Packet loss improvement with defense method.

# HT-Greyhole Router	**1**	**3**	**6**
Transpose Traffic	27.3%	56.8%	76%
BitReverse Traffic	27.6%	56.2%	72%
Uniform Traffic	23.6%	50.5%	66%

8 Conclusion and Future Work

The HT-Greyhole DoS attack targeting NoC can possibly cause network performance decrease and higher packet loss. In this paper the attack within SDNoC context is introduced and a detection method and a defense method have been designed and evaluated. Through the evaluation of the detection algorithm by using binary classification, we explore the different possibilities of threshold values by finding the most accurate. Afterwards by taking into account the performance results, it is obvious that the packet loss increase and throughput decrease are not significant (3%-10%) enough in order to detect an HT-Greyhole router, due to its stealthy behavior. Hence, the need of an alternate detection method able to detect malicious routers and a defense method which allows the normal function of the systems is mandatory. By applying our defense method, the interconnection system continues to function normally by improving the overall packet loss 23.6%-77% and the average throughput 63%-89%. As a future work, more measurements in the context of power and area consumption of this attack could be considered together with the time of HT-Greyhole router detection and its affect on the system.

References

1. Swarup Bhunia and M Tehranipoor. *The Hardware Trojan War*. Springer, 2018.
2. Dean Michael Ancajas, Koushik Chakraborty, and Sanghamitra Roy. Fort-nocs: Mitigating the threat of a compromised noc. In *Proceedings of the 51st Annual Design Automation Conference*, pages 1–6. ACM, 2014.
3. Jonathan Frey and Qiaoyan Yu. Exploiting state obfuscation to detect hardware trojans in noc network interfaces. In *2015 IEEE 58th International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 1–4. IEEE, 2015.

4. Mubashir Hussain and Hui Guo. Packet leak detection on hardware-trojan infected nocs for mpsoc systems. In *Proceedings of the 2017 International Conference on Cryptography, Security and Privacy*, pages 85–90. ACM, 2017.
5. Travis Boraten and Avinash Karanth Kodi. Mitigation of denial of service attack with hardware trojans in noc architectures. In *2016 IEEE international parallel and distributed processing symposium (IPDPS)*, pages 1091–1100. IEEE, 2016.
6. JS Rajesh, Koushik Chakraborty, and Sanghamitra Roy. Hardware trojan attacks in soc and noc. In *The Hardware Trojan War*, pages 55–74. Springer, 2018.
7. Meenakshi Tripathi, Manoj Singh Gaur, and Vijay Laxmi. Comparing the impact of black hole and gray hole attack on leach in wsn. *Procedia Computer Science*, 19:1101–1107, 2013.
8. David Martins and Hervé Guyennet. Wireless sensor network attacks and security mechanisms: A short survey. In *2010 13th International Conference on Network-Based Information Systems*, pages 313–320. IEEE, 2010.
9. Li Zhang, Xiaohang Wang, Yingtao Jiang, Mei Yang, Terrence Mak, and Amit Kumar Singh. Effectiveness of ht-assisted sinkhole and blackhole denial of service attacks targeting mesh networks-on-chip. *Journal of Systems Architecture*, 89:84–94, 2018.
10. Luka Daoud and Nader Rafla. Analysis of black hole router attack in network-on-chip. In *2019 IEEE 62nd International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 69–72. IEEE, 2019.
11. Luka Daoud and Nader Rafla. Detection and prevention protocol for black hole attack in network-on-chip. In *Proceedings of the 13th IEEE/ACM International Symposium on Networks-on-Chip*, page 22. ACM, 2019.
12. Soultana Ellinidou, Gaurav Sharma, Sotirios Kontogiannis, Olivier Markowitch, Jean-Michel Dricot, and Guy Gogniat. Microlet: A new sdnoc-based communication protocol for chiplet-based systems. In *2019 22nd Euromicro Conference on Digital System Design (DSD)*, pages 61–68. IEEE, 2019.
13. Konstantin Berestizshevsky, Guy Even, Yaniv Fais, and Jonatan Ostrometzky. Sdnoc: Software defined network on a chip. *Microprocessors and Microsystems*, 50:138–153, 2017.
14. Liu Cong, Wang Wen, and Wang Zhiying. A configurable, programmable and software-defined network on chip. In *2014 IEEE Workshop on Advanced Research and Technology in Industry Applications (WARTIA)*, pages 813–816. IEEE, 2014.
15. Nick McKeown. Software-defined networking. *INFOCOM keynote talk*, 17(2):30–32, 2009.
16. Natalie Enright Jerger, Tushar Krishna, and Li-Shiuan Peh. On-chip networks. *Synthesis Lectures on Computer Architecture*, 12(3):1–210, 2017.
17. Giorgos Dimitrakopoulos, Anastasios Psarras, and Ioannis Seitanidis. *Microarchitecture of Network-on-chip Routers*, volume 1025. Springer, 2015.
18. Ge-Ming Chiu. The odd-even turn model for adaptive routing. *IEEE Transactions on parallel and distributed systems*, 11(7):729–738, 2000.
19. Niket Agarwal, Tushar Krishna, Li-Shiuan Peh, and Niraj K Jha. Garnet: A detailed on-chip network model inside a full-system simulator. In *2009 IEEE international symposium on performance analysis of systems and software*, pages 33–42. IEEE, 2009.
20. Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Information processing & management*, 45(4):427–437, 2009.
21. Tom Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006.