Eavesdropping Attack Detection using Machine Learning in Network-on-Chip Architectures

Chamika Sudusinghe[‡], Subodha Charles[‡], Sapumal Ahangama[‡] and Prabhat Mishra[*]
[‡]University of Moratuwa, Colombo, Sri Lanka
[*]University of Florida, Gainesville, Florida, USA

***Abstract*—Advances in chip manufacturing technologies have enabled computer architects to utilize System-on-Chip (SoC) to integrate the intellectual property cores as well as other components. Network-on-Chip (NoC) is widely used to fulfill communication requirements in SoC architectures. Securing NoC is vital for designing trustworthy SoCs. Eavesdropping attacks can exploit NoC vulnerabilities to extract secret information. In this paper, we propose a machine learning based detection of eavesdropping attacks. Our machine learning models are trained offline and have been used for runtime detection with a collective decision making strategy. Experimental results demonstrate that our approach can provide high accuracy with minimal overhead.**

***Index Terms*—System-on-chip, network-on-chip, hardware security, eavesdropping attacks, machine learning.**

I. Introduction

Recent advances in chip manufacturing technologies have enabled computer architects to integrate an increasing number of processor cores and other heterogeneous components on a System-on-Chip (SoC). Modern SoC designs consist of numerous Intellectual Property (IP) cores that communicate using a Network-on-Chip (NoC) architecture. Compared to other communication approaches, NoC is superior in terms of communication performance, power consumption, signal integrity, and system scalability because of its low-latency and high-throughput [1]. This has encouraged the manufacturers to frequently include NoCs in their SoC designs. Due to cost and time-to-market considerations, manufacturers utilize the global supply chain to get third party IP cores from different parts of the world. The distributed nature of the NoC and its increased usage has made the NoC a focal point for potential security threats such as the insertion of hardware Trojans (HTs).

An implanted HT can launch a wide variety of attacks ranging from denial-of-service attacks, eavesdropping attacks, data integrity attacks, spoofing attacks, and side-channel attacks [1]. Eavesdropping attacks can lead to serious consequences in NoC-based SoCs since it allows an attacker to extract secret information without relying on memory access. Furthermore, the attacker can leak these sensitive information over a long period without being detected. Therefore, it is critical to have adequate countermeasures to secure NoC-based SoCs from eavesdropping attacks.

This work was partially supported by NSF grant SaTC-1936040.
Manuscript received May 20, 2022; revised July 29, 2022; accepted August 19, 2022. This article was presented at the NOCS 2022 and appears as part of the Design & Test special issue.

A common threat model used to explore eavesdropping attacks in NoCs is where an HT-infected router colluding with an accompanying malicious application running on another IP core to eavesdrop on packets transferred through the router [1], [2]. Due to the complexity of the NoC design, it is difficult to detect these HT-infected routers via functional verification. Previous work on mitigating eavesdropping attacks have explored different approaches such as authentication and encryption [3], performing additional validation checks [2], information obfuscation, and implementing rule-based checkers in wireless NoCs [4]. While these approaches are promising, they have several practical limitations. They can introduce unacceptable design overhead. Moreover, most of these approaches use highly predictable traffic patterns and cannot handle application or input variations. Therefore, these approaches are not applicable in the presence of realistic scenarios, which consist of application characteristics changes, task preemption, task migration, and input variations. To address these runtime variations, we investigate the viability of applying machine learning for eavesdropping attack detection.

Machine learning (ML) has been employed in NoC based SoCs in the context of NoC power consumption, HT detection within IP cores and detecting denial-of-service attacks [5], [6] with high accuracy. To the best of our knowledge, securing NoC-based SoCs from eavesdropping attacks using machine learning remains an unexplored territory. In this paper, we propose an ML-based eavesdropping attack detection that achieves high accuracy across diverse application scenarios without relying on subjective features such as packet source and destination.

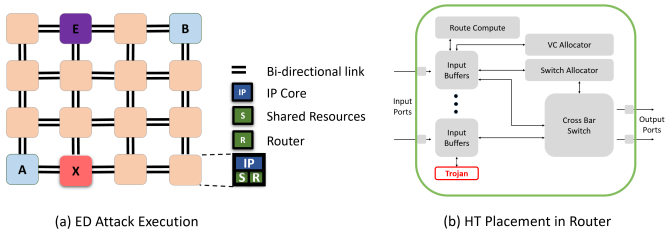
Major contributions of this paper are as follows;

- We present an efficient framework to utilize machine learning techniques combined with a collective decision making strategy for detection of eavesdropping attacks.
- We illustrate how NoC traffic features can be used effectively to handle different execution scenarios.
- We conduct multiple experiments considering variation of application execution scenarios and percentage of information snooping to show the effectiveness our approach.

The paper is organized as follows. Section II outlines the threat model. Section III surveys related efforts. Section IV describes our machine leaning based eavesdropping attack detection methodology. Section V presents the experimental results. Finally, Section VI presents our conclusion and Section **??** describes our limitations and future work.

II. Threat Model

Eavesdropping (ED) attacker listen to confidential information passing through the NoC when other IP cores are communicating. These attackers passively listens to on-chip communication (secure NoC packets) in an attempt to steal sensitive information with intention to leak information over extended periods without being detected. Previous work in this domain has explored several variations of eavesdropping attacks. We consider a well explored threat model [1], [2] in previous research where the malicious NoC router colludes with another malicious application running on a rogue IP core to launch an ED attack. This can be illustrated by a simple example using two trusted IP cores A & B, a malicious IP core E, and a malicious router X as shown in Figure 1(a). This attack is enabled by an HT located inside the router as shown in Figure 1(b), which makes copies of the packets received at the input buffer, alters header information to navigate the packets to E (a different destination), and places the packets back within the input buffer. In our threat model, we consider scenarios with multiple pairs of sources and destinations communicating simultaneously.



(a) ED Attack Execution (b) HT Placement in Router

Fig. 1. A Trojan infected router copying packets and sending them to a malicious application running on another IP core to launch an ED attack.

III. Related Work

Numerous research has been conducted to secure NoC-based SoCs from ED attacks over the past few years. They include mechanisms such as functional verification, authentication and encryption [3], performing additional validation checks [2], information obfuscation, and implementing rule-based checkers in wireless NoCs [4]. However, these defence mechanisms have practical limitations. Functional verification cannot guarantee the detection of all forms of attacks during circuit testing/validation due to multiple reasons such as limited information provided by the third party IP vendors, lack of a golden reference model, smaller footprint, complexity of the NoC design, etc.

Several authenticated encryption schemes using AES Counter mode [1] were prominent in the domain in the recent years. These approaches used tunnel based communication mechanism in isolating sensitive information. Ancajas et al. [7] discusses about having a simple XoR cipher combined with a packet certification technique to calculate a tag and validate it at the receiver. These approaches use dynamic tagging of packets while hiding the location of tag bits. Additional validation checks is another mechanism that can be used to detect packet duplication in NoC routers and NIs. Raparti et al. discussed about a snooping invalidator module to detect data duplication and a snooping detection module to detect malicious applications [2]. They are using this approach to detect HTs in the NIs by discarding packets with invalid header flits being injected into the NoC.

There are a number of efforts to utilize machine learning concepts to detect HTs within IP cores in the SoC context. Most of the existing machine learning approaches have been used to detect denial-of-service (DoS) attacks. Kulkarni et al. had made the first attempt to detect real-time hardware Trojan using machine-learning techniques for NoC-based many-core architectures [8]. A novel method of detecting DoS attacks in NoCs using Spiking Neural Networks (SNN) was proposed by Madden et al. [9]. Their detection approach explored the traffic flow of each channel of a router (North, East, South and West) to analyze the packet exchange between adjacent routers. Another runtime mechanism to detect DoS attacks was implemented by Sudusinghe et al [5]. They have trained multiple machine learning models for different attack and normal scenarios to detect DoS attacks. Sinha et al. have proposed to secure accelerator rich SoCs from flooding type of DoS attacks and to localize DoS attacks in NoC based SoCs [10]. Their approach involves a machine learning approach and a localization algorithm which is used to trace back the attack path. All of these approaches have trained the respective machine learning models offline using the collected data samples to decrease the performance and computational overheads. It has to be noted that most of the existing literature has focused on mitigating DoS attacks using machine learning. To the best of our knowledge, our proposed approach is the first attempt in securing NoC-based SoCs from eavesdropping attacks using machine learning.

IV. ML-based Attack Detection

We propose a machine learning based runtime mechanism to detect ED attacks in the presence of varying NoC application scenarios. Our approach consists of three important phases as shown in Figure 2: design time, training time, and runtime. During design time, we craft features that can be potentially used to detect ED attacks inferring NoC traffic, design probes attached to routers to collect data from NoC packets with minimum performance and power overhead, and develop eavesdropping sensing algorithm (ESA) to detect ED attacks. Then we mimic both normal executions (NEs) and attack executions (AEs) using the gem5 full system simulator. The detailed description of the experimental setup is in Section V-A. NoC traffic data is gathered from these executions and is used to train machine learning models during training time. The trained machine learning models are stored in a dedicated IP core, the *decision unit (DU)*. The concept of DU is inline with the Security Policy Engine" (SPE) [1] in a NoC-based SoC, which depicts the possibility of executing the computations related to a machine learning model within the SoC infrastructure. Utilizing the ML models available at the DU, during runtime, NoC traffic collected by the probes are analyzed to make inferences based on the information in the router and specified time frames.

[1] *Security Policy Engine (SPE)* is a specific IP core in the SoC which is used to implement security constraints to be enforced by the policies.

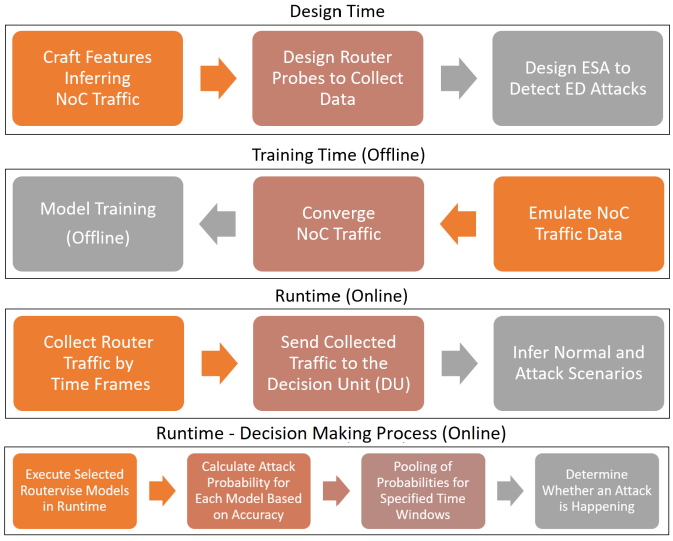


Fig. 2. The three important phases of our proposed ML-based attack detection framework: design time, training time, and runtime.

A. *Design Time*

Analysis of NoC traffic involves data from a large number of features of the packets/flits. While there are a substantial amount of data available, selecting the suitable features and crafting new features are key considerations to develop a robust detection mechanism. Previous studies have explored the usage of numerous features to make inferences. In our approach, we consider a feature vector of 10 features for each machine learning model. The selection of 10 features is based on the relative importance of each feature for the decision making process. Hence, the selected 10 features of each router differs based on the threat model, dataset and placement of routers. We have experimentally evaluated the performance of all available features and the selection was made to accommodate a rational trade off between the number of inspected features and the hardware (performance & area) overhead. The selected feature vector differs for each router and each execution scenario. For example, if the number of application executions is different in two variations of the threat model, the feature vector of a specific router would have different sets of features reflecting the existent execution. Some of the significant NoC features considered in the analysis are router channel traffic from different ports (inports and outports), virtual channel (vc) allocation, packet count, packet transfer ratio, virtual network utilization, and buffer utilization.

The concept of "multiple physical NoCs" has been widely adopted in the domain to facilitate communication among the IP cores within a NoC [5]. This approach allows to have two separate channels to handle packet traversals without causing additional congestion for the packet transfers involved with the application executions. Employing this approach, we use a separate NoC for packet transfers from the router probes to the DU. This allows us to avoid additional performance degradation that can be caused by packet congestion, and evade from carrying out packet header modifications to utilize the same channel for two different packet transfer types. Following the work done by Sudusinghe et al. [5], we use a Data NoC and a Service NoC, where the Service NoC is used for packet transfers from the probes to the DU. Here, the additional power and area overhead caused by the usage of two NoCs compared to a single NoC are 7% and 6%, respectively. This is achieved by fitting design parameters in consistency with Yoon et al's analysis [11] about the usage of multiple physical NoCs.

Data acquisition probes attached to the router are used to collect NoC traffic data and send to the DU for further processing. Previous research has been done to analyze the feasibility of using such probes for various purposes. Sudusinghe et al. [5] have utilized probes attached to routers to collect data and send to a "security engine" (similar to SPE) to infer DoS attacks. After exploring the previous work and implementations, we have used an analogous design for our approach. We made sure that our design is consistent with the prior work, where probes consumed $0.05mm^2$ area in comparison with a $0.26mm^2$ router area when synthesized with 0.13 micron technology [5].

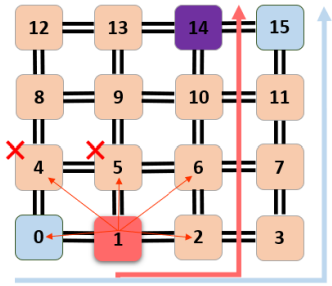


Fig. 3. An example scenario for neighbouring router selection for eavesdropping sensing algorithm (ESA).

The continuous execution of the DU can result in causing additional performance overhead. We use a modified bully algorithm [2] combined with a trigger mechanism to select the actively contributing machine learning models in a particular time frame. The overall mechanism is called the eavesdropping sensing algorithm (ESA) as shown in Figure 3. Algorithm 1 outlines the major steps in ESA. ESA triggers (routerTrigger) when the packet transfer ratio of a particular router is greater than 1 by analyzing the traffic data (θr) passing through routers. Then the trigger mechanism considers that particular router as a potentially rogue router and analyzes the packet transfer ratios and the traffic rates of its neighbouring routers using the modified bully algorithm. If a neighbouring router happens to have a higher packet transfer ratio, then it will be selected as the new potentially rogue router and the process is repeated until the router with the highest ratio is found (modifiedBully). Then the machine learning models will be activated for the selected router and its neighbouring routers (modelActivation) with the highest traffic rates using their traffic data as the datasets (routerTraffic). As shown in Figure 3, the rogue router 1, and its neighbouring routers 0, 2, 6 are selected for modeling while routers 4 and 5 are discarded as no packets are traversing through them. Here, based on the traffic data, router 1 is selected as the router with the highest packet transfer ratio by the modified bully algorithm.

[2]*Bully Algorithm* is a methodology which can dynamically choose a leader or a coordinator from a group of distributed processes.

If there is no router that activates this trigger mechanism, the machine learning models of the three routers with the highest number of traffic rates will be activated after a time threshold, relaxationWindow (10000 cycles). This is implemented as a failsafe mechanism to activate machine learning models in a constant basis. Our results indicate that there is no adverse effect due to the inclusion of this component for the conducted experiments. The values of minNeigh, maxNeigh and relaxationWindow are experimentally evaluated based on the selected NoC topology (4x4 mesh) and the threat model.

Algorithm 1: Eavesdropping Sensing Algorithm

```
1  triggerThreshold = 1;
2  minNeigh = 3;
3  maxNeigh = 5;
4  completionTime = 0;
5  relaxationWindow = 10000;
6  while TRUE do
7      triggerRatio, router = routerTrigger(θr);
8      if triggerRatio > triggerThreshold then
9          rogue, neighbours =
             modifiedBully(router,minNeigh,maxNeigh);
10         traffic = routerTraffic(θr,rogue,neighbours);
11         modelOut =
             modelActivation(rogue,neighbours,traffic);
12         completionTime = getTime(modelOut);
13     else
14         if (getTime() - completionTime) >
             relaxationWindow then
15             routers, traffic =
                 routerTraffic(θr,minNeigh);
16             modelOut =
                 modelActivation(routers,traffic);
17             completionTime = getTime(modelOut);
18         else
19     end
20 end
```

B. Training Time

Machine learning models are trained offline for a predefined execution scenarios which accommodates potential ED attacks that can emerge within the system. This has been an approach that is reliably used in multiple instances, specifically to detect DoS attacks using machine learning [8], [10], [5] as it reduces the additional overhead of the system. Further, having access to a larger dataset for an extended execution period allows the machine learning models to learn from the varying traffic patterns across multiple execution scenarios. During the training time, the emulated NoC traffic data traversing through the routers are collected using the probes to create feature vectors. These feature vectors are created per each flit (atomic pieces that form a packet) and then used for model training, validation and parameter optimization.

C. Runtime

During runtime, probes collect NoC traffic data traversing through the routers for specified time frames and the data is sent for DU to make inferences. Within the DU, the router traffic is analyzed using the ESA to detect whether there is one or more routers with more outgoing packets compared to incoming packets. This is measured using packet transfer ratio. Upon the activation of the trigger mechanism, the feature vectors within a specific time frame are aggregated to be used as the input for the trained models. Here, the routers which will be considered for evaluating will be selected in accordance with the results obtained from ESA. Then a collective decision making strategy (CDMS) is followed to involve all the selected routers to determine whether there is an ED attack taking place or not. The results of CDMS is the aggregated likelihood of an attack taking place in the system. Due to the usage of this combined approach of ESA and CDMS, the likelihood of having false positives for normal congestions in the network is reduced as the ESA triggers at when the packet transfer ratio differs and for CDMS multiple routers are considered.

In the implementation of CDMS, we use an extensively studied approach in the domain of probability theory called *Opinion Pooling* to aggregate all the decisions of multiple machine learning models of different routers. Let us consider a trained machine learning model for router R_a as M_a. The set of feature vectors F_a includes the NoC traffic data for the router R_a within the time frame TF_k to be fed to the machine learning model M_a. For a particular time frame and a particular router, when the feature vectors are fed into the machine learning model, it will yield the ED attack probability EP_a within that time frame with respective to the particular router. These results will be aggregated for all the routers that joining the decision making process and will be divided by the total number of feature vectors traversed through the selected routers within the specific time frame. This is analogous to the number of flit transfers FT_k within the same time frame. The final decision ($eaves_t$) of the CDMS for the given time frame can be illustrated using the following equation.

$$eaves_t = \frac{\sum_{\forall a}(|EP_a(M_a, F_a)|)}{\sum_{\forall k}|(FT_k|)} \tag{1}$$

V. Experiments

In this section, we present our evaluations for the proposed approach. First, we describe our experimental setup. Next, we describe our dataset for different scenarios and analyze different machine learning models to select the best fit. Then, we present the experimental results for multiple execution scenarios and identify the best performing features. Finally, we perform overhead analysis.

A. Experimental Setup

We use a well explored architecture model [7] in developing the experimental setup for our evaluations. The 4×4 mesh NoC was modeled using the "GARNET2.0" interconnection network model integrated with the gem5 full system simulator. It has provided a cycle accurate micro architectural implementation of an on-chip network router. The implementation of the NoC model was done using X-Y routing with wormhole switching. We tested the model using the Garnet Synthetic Traffic Injector which is built on top of the Garnet Standalone

cache coherence protocol. Default Garner2.0 implementation is used for packet formats. For each flit, 128 bits are allocated. The control packet is represented by 1 flit and the data packets are by 5 flits. In control packet, 64 bits are allocated for the payload (address) while data packets have a payload of 512 bits. Both the gem5 simulator and the Garnet protocol have been widely used in the domain [1], [3] and that motivated us to use the same approach. Our experiments were conducted using 51 traffic traces for eavesdropping attacks with varying traffic rates.

During normal execution, two or more application executions can happen within the SoC. During an attack execution, a malicious router in the communication path of two IPs will be copying packets traversing the path and sending them to a malicious attacker IP for further exploitation as shown in Figure 4. For our experiments, modifications were done to the source code of Garnet protocol to send copied packets from the rogue router within the same cycle. Further, the frequency of packet copying is maintained in such a manner that the percentage snooping of information would range from 10% to 100%. Modeling of the attack execution has been done with minimal power and area footprint making it difficult to detect via traditional approaches such as functional verification. Depending on the complexity of the threat model, the number of application executions and the number of malicious routers engaging will be higher.

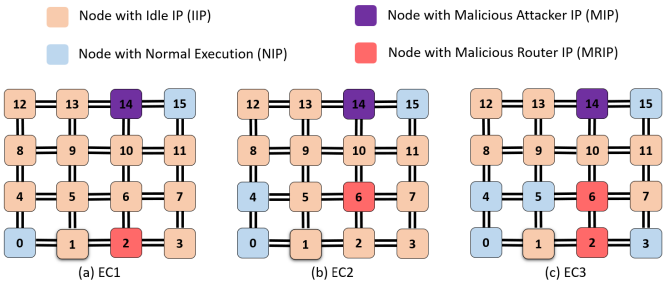


Fig. 4. Different application execution types considered for model training.

B. Dataset

To conduct the experiments, the dataset for both training and testing were emulated considering three different execution scenarios (ECs) in order to inspect the feasibility of using machine learning techniques to detect ED attacks. The training data used for ECs can be described as follows in Table I: two application executions and one malicious router (EC1), three application executions and one malicious router (EC2), and five application executions and two malicious routers (EC3). For each of the ECs, we have considered three cases representing the percentage snooping of information of 25%, 50%, and 100%. Then, the trained models were tested against unseen data for both normal executions and attack executions.

C. Model Selection

We experimented using multiple machine learning models to analyze the feasibility of using a learning based technique to detect an ED attack. Analysis was done considering two approaches, where as model training was done for all data points at once and training the models based on the data passing through the routers. The results from the trained models for all data points did not yield promising results and the rest of the analysis is based on modeling using router-specific data. Figure 5(b) illustrates the results of machine learning models: Linear Regression (LR), MLP Neural Networks (MLP), Decision Tree Classifier (DT), Random Forest Classifier (RF) and XGBOOST Classifier (XGB), used for the analysis considering the execution scenario EC2. Based on our analysis, XGBOOST, which is a distributed gradient boosted decision tree algorithm, has performed well. Similar trends have been observed in previous research for DoS attack detection in NoCs [5] and malicious traffic detection methodologies in the computer security domain. For the rest of the exploration we use XGBOOST for training machine learning models for router-specific data, separately for each router. Each model is trained with different parameters based on the training data and the selected features.

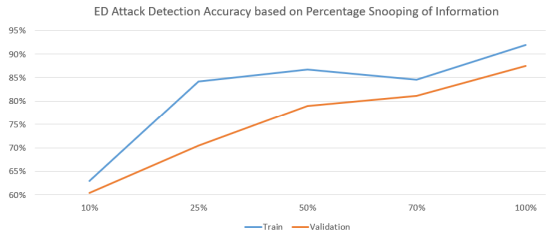
TABLE I
DATASET FOR THREE EXECUTION SCENARIOS (EC1, EC2 AND EC3). HERE, N-5-15 MEANS NORMAL EXECUTION WITH IP 5 (SOURCE) RUNNING THE APPLICATION THAT IS COMMUNICATING WITH IP 15 (DESTINATION). SIMILARLY, A-6-14 MEANS ATTACK SCENARIO INVOLVING MALICIOUS ROUTER IN IP 6 COPYING THE PACKET AND SENDING TO THE COLLUDING APPLICATION RUNNING IN IP 14.

Execution Type (EC)	Train		Test
	Normal	Attack	
EC1	N-0-15	A-2-14	A-1-14
EC2	N-0-15	A-6-14	A-7-14
	N-4-15		
EC3	N-0-15	A-2-14	A-1-14
	N-3-15	A-6-14	A-7-14
	N-4-15		N-2-15
	N-5-15		N-6-15

D. Experimental Results

The performance of the model in the presence of an attack was evaluated based on accuracy, F1 score, precision and recall for all three execution scenarios. We performed a comparison of the training and validation accuracy of datasets based on the percentage snooping of information due to an ED attack as shown in Figure 5(a) for the application execution scenario in EC1. The results manifested the fact that it would be harder to detect smaller attacks, when only 10% of the information is being eavesdropped. But, with the increase of the percentage eavesdropping of information, the detection accuracy significantly increases reaching as high as 87.42%. For the rest of the analysis, a comparison was done among the three application execution types mentioned in Table I considering 25%, 50%, and 100% as the percentage snooping of information.

Figure 5(c) illustrates the performance of the machine learning models both in the presence of seen and unseen application executions. The average accuracy increases when the number of application executions are higher. This trend is evident in the results obtained for EC1, EC2 and EC3. In comparison with EC1 & EC2, EC3 has a rich dataset with higher number of varying application executions. This has enabled the machine learning models to learn from diverse traffic patterns.



(a) XGB model performance with varying snooping percentage for EC1

% Accuracy of Machine Learning Approaches (50%)					
EC2 Dataset	**LR**	**MLP**	**DT**	**RF**	**XGB**
Train	56.38	68.01	74.11	78.35	89.28
Val	53.51	66.67	69.43	72.64	81.92

(b) Comparison of machine learning approaches for the dataset EC2

	ED%								
	25%			50%			100%		
Dataset Type	**Train**	**Val**	**Test**	**Train**	**Val**	**Test**	**Train**	**Val**	**Test**
EC1	84.14	70.42	71.78	86.73	78.91	77.24	91.93	87.42	75.65
EC2	85.77	78.83	74.26	89.28	81.92	80.45	94.48	91.37	91.84
EC3	99.91	99.65	97.37	99.95	99.87	98.39	100	100	99.52

(c) Experimental results for EC1, EC2, & EC3 with varying snooping percentages

Fig. 5. Experimental results for EC1, EC2, & EC3

Further, when the snooping percentage increases, the accuracy increases comparably for all executions, justifying the fact that larger attacks can be detected easily. The results have a similar trend for all training, validation and testing accuracy. Hence, we can safely assume that the trained machine learning models are performing well and not overfitting.

Dataset Type	**Test Data**	**CDMS Test Results (50%)**					**Accuracy**	**F1-Score**	**Precision**	**Recall**
EC1	A-7-14	84.31	79.58	73.14	71.93	-	77.24	0.7793	0.7884	0.7704
EC2	A-7-14	80.64	83.93	77.85	79.39	-	80.45	0.8054	0.8152	0.7958
EC3	A-7-14	99.41	98.13	99.22	99.45	98.44	98.93	0.9882	0.9919	0.9846
	A-7-14	99.27	97.63	98.55	99.38	98.42	98.65	0.9850	0.9906	0.9794
	N-2-15	96.87	98.11	97.43	-	-	97.47	0.9760	0.9755	0.9764
	N-6-15	98.51	98.29	98.67	-	-	98.49	0.9823	0.9805	0.9842

(a) Experimental results for EC1, EC2 and EC3 including the collective decision making strategy (CDMS)

Top 5 Performing Features		
EC1	**EC2**	**EC3**
packet count	packet count	packet count
vc allocation	vc allocation	vc allocation
flit_id	flit_id	router channel traffic
outport_2	buffer utilization	virtual network
buffer utilization	import	outport_1

(b) Top five performing features of router 2 in the presence of different execution types at 50% as the percentage snooping of information

Fig. 6. Experimental results with collective decision making strategy (CDMS) and top performing features.

Further exploration on the results were done considering the collective decision making strategy (CDMS) of the routers. The results are shown in Figure 6(a). In CDMS, based on the traffic rates and other criteria considered in ESA, the number of routers considered for the final detection accuracy and other metrics differs. For example, while only four routers were considered in EC1 & EC2, three routers have involved in the first two scenarios in EC3. Since the last two scenarios are NEs, in accordance with the ESA, only three routers were taken into consideration. This number was decided after experimentally evaluating using multiple application executions and traffic patterns. Apart from the accuracy values, considering the other evaluation metrics, both precision and recall values are ranging between $\sim$0.77% and $\sim$0.99%. While, either of them increases at the cost of the other, the resultant values depict the fact that the machine learning models are performing up to the expectations. This is even justified by results for the F1 score, which is the harmonic mean between precision and recall.

E. Best Performing Features

As discussed in Section IV-A, there is a significant impact for the results from the features utilized by each machine learning model. Unlike the previous attempts at using machine learning for attack detection in NoC based SoCs, in our approach the feature vector used for the model differs based on the application executions and the placement of the routers. For a selected application execution, the routers involve in the CDMS have distinct feature vector for each. A total of 24 feature vectors have been analysed in our experiments as depicted in Figure 6(a). Figure 6(b) illustrates the top five performing features of router 2 for the three different execution types. It can be perceived that both EC1 and EC2 has a similar trend while EC3 deviates from the prior scenarios. It is mainly due to the relative similarities of the application executions happening in EC1 and EC2, whereas EC3 is different. Similar trend has been observed in other routers with respective to the top 5 features in our analysis.

F. Overhead Analysis

Unlike previous work, the implementation of ESA has allowed us to significantly reduce the number of machine learning models contributing in the CDMS while increasing the overall accuracy. Since these models were trained offline in the training time, only the weights of the trained models are used for making predictions reducing computational overhead. The designs of the probes in the routers and multiple physical NoCs are consistent with the prior work in the domain [5], [11]. The power and area overhead incurred by the usage of multiple physical NoCs (Data NoC and a Service NoC) are 7% and 6%, respectively, while the router probes consumed $0.05mm^2$ area in comparison with a $0.26mm^2$ router area. These values fall within the same range of the exiting work in the domain [12], [10], [2].

VI. Conclusion

Eavesdropping attacks have become a major concern in NoC-based SoCs since it allows an attacker to extract secret information. In this paper, we have introduced a machine learning based mechanism to detect eavesdropping attacks in NoC-based SoCs. We have used an extensively explored threat model, where a malicious NoC router colludes with a malicious application running on an IP core to launch an eavesdropping attack. We have developed an efficient framework with machine learning and collective decision making strategy. Experimental results demonstrated the effectiveness of our approach in detecting eavesdropping attacks with high accuracy within a smaller time frame for varying application execution scenarios. Our approach can be integrated with other defense mechanisms to effectively detect eavesdropping attacks with high accuracy.

REFERENCES

[1] S. Charles and P. Mishra, "A survey of network-on-chip security attacks and countermeasures," *ACM Computing Surveys*, vol. 54, no. 5, 2021.

[2] V. Raparti *et al.*, "Lightweight mitigation of hardware trojan attacks in noc based manycore computing," in *DAC*, 2019, p. 48.

[3] S. Charles *et al.*, "Lightweight and trust-aware routing in noc-based socs," in *ISVLSI*, 2020, pp. 33–42.

[4] A. Vashist *et al.*, "Securing a wireless network-on-chip against jamming-based denial-of-service and eavesdropping attacks," *TVLSI*, pp. 2781–2791, 2019.

[5] C. Sudusinghe *et al.*, "Denial-of-service attack detection using machine learning in network-on-chip architectures," in *International Symposium on Networks-on-Chip (NOCS)*, 2021, pp. 35–40.

[6] ——, *Network-on-Chip Attack Detection using Machine Learning*. Springer, 2021, p. 253–275.

[7] D. Ancajas *et al.*, "Fort-nocs: Mitigating the threat of a compromised noc," in *DAC*, 2014, pp. 1–6.

[8] A. Kulkarni *et al.*, "Svm-based real-time hardware trojan detection for many-core platform," in *ISQED*, 2014.

[9] K. Madden *et al.*, "Adding security to networks-on-chip using neural networks," in *SSCI*, 2018.

[10] M. Sinha *et al.*, "Sniffer: A machine learning approach for dos attack localization in noc-based socs," *JETCAS*, pp. 278 – 291, 2021.

[11] Y. J. Yoon *et al.*, "Virtual channels and multiple physical networks: Two alternatives to improve noc performance," *TCAD*, vol. 32 (12), 2013.

[12] S. Charles *et al.*, "Real-time Detection and Localization of Distributed DoS Attacks in NoC based SoCs," *IEEE TCAD*, 2020.