# Semi-supervised Trojan Nets Classification Using Anomaly Detection Based on SCOAP Features

Pei-Yu Lo*, Chi-Wei Chen*, Wei-Ting Hsu*, Chih-Wei Chen‡, Chin-Wei Tien†, and Sy-Yen Kuo*
*Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan
† Graduate Institute of Communication Engineering, National Taiwan University, Taipei, Taiwan
‡ Cybersecurity Technology Institute, Institute for Information Industry, Taipei, Taiwan
Email: {r08921a29, r08921a28, r09921a30}@ntu.edu.tw, chihweichen@iii.org.tw, {cwtien, sykuo}@ntu.edu.tw

***Abstract*—Recently, hardware Trojan has become a serious security concern in the integrated circuit (IC) industry. Due to the globalization of semiconductor design and fabrication processes, ICs are highly vulnerable to hardware Trojan insertion by malicious third-party vendors. Therefore, the development of effective hardware Trojan detection techniques is necessary. Testability measures have been proven to be efficient features for Trojan nets classification. However, most of the existing machine-learning-based techniques use supervised learning methods, which involve time-consuming training processes, need to deal with the class imbalance problem, and are not pragmatic in real-world situations. Furthermore, no works have explored the use of anomaly detection for hardware Trojan detection tasks. This paper proposes a semi-supervised hardware Trojan detection method at the gate level using anomaly detection. We ameliorate the existing computation of the Sandia Controllability/Observability Analysis Program (SCOAP) values by considering all types of D flip-flops and adopt semi-supervised anomaly detection techniques to detect Trojan nets. Finally, a novel topology-based location analysis is utilized to improve the detection performance. Testing on 17 Trust-Hub Trojan benchmarks, the proposed method achieves an overall 99.47% true positive rate (TPR), 99.99% true negative rate (TNR), and 99.99% accuracy.**

***Index Terms*—hardware security, hardware Trojan, gate-level, testability, machine learning, anomaly detection**

## I. Introduction

As the global economy becomes increasingly diverse, IC design and manufacturing are outsourced and distributed worldwide. However, outsourcing manufacturing poses a severe threat of hardware Trojan attacks on IC products. These Trojans may leak confidential information, change functionality, or degrade system performance. Hardware Trojans typically consist of a trigger and a payload. The trigger activates the payload when trigger condition(s) are met. Stealthy hardware Trojans have rare trigger conditions and low trigger probabilities, making them extremely difficult to detect through conventional IC testing and verification processes. At the gate level, many machine-learning-based detection techniques have been developed. However, most of them use supervised learning methods, which involve time-consuming training processes, need to deal with the class imbalance problem, and are not pragmatic in real-world situations. Furthermore, no works have explored the use of anomaly detection for hardware Trojan detection tasks.

This paper proposes a semi-supervised learning method to detect Trojan nets in gate-level netlists. We first extract the SCOAP [1] values considering all types of D flip-flops with/without clock and asynchronous/synchronous preset/clear. Two anomaly detection techniques, outlier detection and novelty detection, which do not need class label information during the training processes and effectively handle the class imbalance problem, are then utilized to identify suspicious nets. Finally, a novel topology-based location analysis is adopted to determine the Trojan nets from suspicious nets. The contributions of this paper are as follows:

1) Propose a semi-supervised hardware Trojan detection method.

2) Ameliorate existing computation of SCOAP values by considering all types of D flip-flops.

3) The first to utilize anomaly detection technique for Trojan nets classification.

4) The first to propose a topology-based location analysis to determine Trojan nets.

The remainder of this paper is organized as follows: Section II discusses the existing machine-learning-based hardware Trojan detection methods in gate-level netlists. Section III presents the proposed semi-supervised hardware Trojan detection approach. Section IV summarizes the experiment results. Section V concludes the paper.

## II. Gate-level Nets Classification Methods Based on Machine Learning

The first machine learning method in gate-level netlists was proposed by [2], which extracted five netlist structural features to detect Trojan nets using a support vector machine. More features were explored in [3], which extracted 51 netlist features and selected the best 11 features maximizing the F-measures using a random forest model. Further, neural-network-based techniques were developed. [4] proposed a single-layer neural network for detection, and [5] proposed multi-layer neural networks to detect Trojan nets. Testing on Trust-Hub benchmark circuits [6], [7], these methods achieved an average TPR and an average TNR ranging from 68.3% to 85% and 49% to 99.7%, respectively.

Later, in contrast to the netlist structural features, testability-based techniques have been proposed. [8] first used combina-

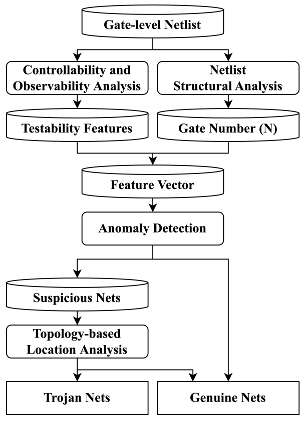

Fig. 1. The proposed framework.

tional testability measures calculated by SCOAP as features for detection using unsupervised k-means clustering. The method claimed to achieve a 100% TPR and TNR in 21 of 23 Trust-Hub benchmark circuits. The use of sequential testability measures as features was proposed in [9], which detected small-sized sequential Trojan circuits without scan chains. The method was tested on six small embedded circuits, and no comparisons have been made.

Further, supervised learning methods using testability features were proposed. [10] extracted both combinational and sequential SCOAP values as features for detection using bagged trees. An adaptive synthetic sampling technique was adopted to cope with the imbalanced data. [11] consolidated SCOAP values and netlist structural features to train classifiers and selected the best feature combination using the minimum redundancy maximum relevance method. [12] combined SCOAP parameters with Controllability/Observability Program parameters and circuit size as the features for training and classification. Finally, [13] used the best set of SCOAP features and the proposed class-weighted extreme gradient boosting model for detection. Testing on different benchmark circuits, these methods achieved an average TPR and average TNR ranging from 82.46% to 100% and 90.06% to 100%, respectively.

## III. Proposed Method

This paper proposes a semi-supervised learning method for Trojan nets classification. The framework of the proposed method is shown in Fig. 1. First, we collect 17 Trust-Hub gate-level benchmark circuits as our dataset. SCOAP values and the gate number are extracted from the netlist to form the feature vector. After that, semi-supervised anomaly detection is utilized to identify the suspicious nets. Lastly, we analyze these suspicious nets based on the topology information to detect Trojan nets.

### A. Net Testability Features

Testability measures were first developed to measure the degree of difficulty to test a circuit and estimate the fault

TABLE I
Gate Propagation Rules for Controllability

| **Gate Inputs** | **Output Controllability** |
|---|---|
| All inputs control output | $\sum$(input controllabilities) $+ 1$ |
| Multiple input sets control output | $\min$(controllabilities of input sets) $+ 1$ |
| Fan-out branches | stem controllability |

| **Gate Output** | **Input Observability** |
|---|---|
| Logic gates | output observability $+$ controllability of other inputs to non-controlling value $+1$ |
| Fan-out stem | $\min$(branch observabilities) |

coverage in design for testability. The Sandia Controllability/Observability Analysis Program (SCOAP) [1] is a topology-based testability analysis algorithm with linear time complexity. It assumes signals are independent and gives numerical estimations of the controllability and observability for each signal through six values: combinational 0-controllability (CC0), combinational 1-controllability (CC1), combinational observability (CO), sequential 0-controllability (SC0), sequential 1-controllability (SC1), and sequential observability (SO). The method first calculates the controllability of each gate from primary inputs to primary outputs. After controllability values are established, it computes observability values in a reverse order starting from primary outputs moving back to the primary inputs. The gate propagation rules of controllability and observability are shown in Table I.

Stealthy hardware Trojans have rare trigger conditions and low trigger probabilities, making them extremely difficult to detect through conventional IC testing and verification processes. In other words, they are hard to be controlled and observed and are expected to have high SCOAP values. This characteristic has been demonstrated using Trust-Hub benchmarks in [8], [10].

In previous works, SCOAP values were calculated using either Synopsys's TetraMAX or an open-source tool, Testability Measurement Tool [14]. However, the values extracted by TetraMAX have a threshold value of 254, which is easy to achieve by normal nets in large-sized circuits and becomes a limitation for research. The Testability Measurement Tool, in contrast, does not have a threshold. Nonetheless, it only implements the SCOAP method for D flip-flops with/without clock and with/without synchronous reset. To obtain more accurate SCOAP values and a better result, we derive the formula for D flip-flops with synchronous/asynchronous set and reset based on the gate propagation rules shown in Table I. For example, the derived combinational formulas of a low-active D flip-flop with input $D$, output $Q$, clock $CLK$, and an asynchronous set $SET$ are shown as follows:

$$
\begin{aligned}
&CC0(Q) = CC0(D) + CC1(CLK) + CC0(CLK) + CC1(SET)\\
&CC1(Q) = \min\{CC0(SET), CC1(D) + CC1(CLK) + CC0(CLK)\}\\
&CO(D) = CO(Q) + CC1(CLK) + CC0(CLK) + CC1(SET)\\
&CO(SET) = CO(Q) + CC0(Q) + CC0(SET)\\
&CO(CLK) = \min\{CO(Q) + CC1(Q) + CC1(SET) + CC0(D)\\
&\qquad + CC1(CLK) + CC0(CLK), CO(Q) + CC0(Q)\\
&\qquad + CC1(SET) + CC1(D) + CC1(CLK) + CC0(CLK)\}
\end{aligned}
$$

We use three combinational SCOAP values (CC0, CC1, CO) and the gate number (N) as features for detection. The gate number representing the circuit size (N) is considered since the average SCOAP values become higher with increasing circuit size.

## B. Semi-supervised Anomaly Detection

Since hardware Trojan circuits occupy only a tiny part of the circuit, the unbalanced datasets have always been a significant problem when using supervised learning models, which assume a relatively balanced data distribution. Moreover, supervised learning methods involve time-consuming training processes that rely on class label information and thus are not pragmatic in real-world situations.

Anomaly detection is the process of identifying unexpected or rare items, events, and observations, whose characteristics are significantly different from the majority of the data. Therefore, we believe this technique is suitable for detecting tiny and stealthy hardware Trojans in circuits. Unlike supervised learning methods, semi-supervised anomaly detection techniques do not need class label information during the training process and effectively handle the class imbalance problem.

Two methods, outlier detection and novelty detection, can be used for anomaly detection. In outlier detection, the training data consist of normal observations and anomalies, while in novelty detection, training data consist of only normal observations. For outlier detection, we apply the isolation forest algorithm [15], which isolates anomalous points rather than building a model of normal instances. The algorithm has linear time complexity with low memory requirement and thus works well with high volume data. For novelty detection, we utilize an autoencoder model, which is a type of artificial neural network used to learn efficient codings of unlabeled data. An encoder compresses the input data to lower dimensions (bottlenecks), and a decoder reconstructs the compressed representation of the original input. Reconstruction error (the error between the original data and its low dimensional reconstruction) is used as an anomaly score to detect anomalies.

## C. Topology-based Location Analysis

Previously, nets classification methods in gate-level netlists mainly focused on statistical analyses and did not involve topology examinations of the circuit. Since hardware Trojan can be regarded as a tiny circuit, Trojan nets are close to each other in terms of their locations. According to this observation, we propose a novel topology-based location analysis method to determine the Trojan nets after anomaly detection.

We define anomalous nets identified by anomaly detection as suspicious nets. The proposed algorithm is shown in Algorithm 1. First, a given netlist is abstracted as a circuit graph $G = (V, E)$, in which vertices represent circuit elements (gates), and edges represent the signal propagation paths (nets). The nearby edges of each suspicious edge $s$ are denoted as $\Gamma(s) = \{\{x, w\} \in E | x \in \tau(s)\} - \{s\}$, where

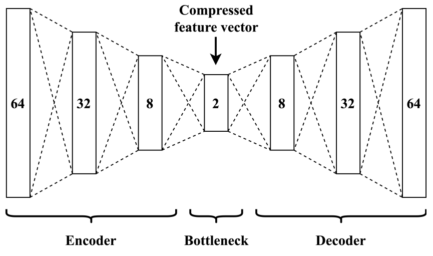

Fig. 2. The architecture of the autoencoder model.

$\tau(s) = \{u, v \in V | \{u, v\} = s\}$ denotes the vertices connected to $s$. Then, for each suspicious edge $s$, if there exists any nearby edge that is suspicious and is not primary input or primary output, we conclude that $s$ is a Trojan net; otherwise, it is a genuine net. The analysis can effectively filter out nets misclassified as anomalous by anomaly detection.

**Algorithm 1:** Topology-based Location Analysis

```
Input: netlist $N$, suspicious nets
Output: Trojan nets $troNet$
1  Abstract $N$ and as circuit graph $G = (V, E)$;
2  $susEdge \leftarrow \{v \in V | v$ is suspicious$\}$;
3  $troNet \leftarrow \emptyset$;
4  for $s \in susEdge$ do
5      $connNode \leftarrow \{u, v \in V | \{u, v\} = s\}$;
6      $connEdge \leftarrow \{\{x, w\} \in E | x \in connNode\} - \{s\}$;
7      for $e \in connEdge$ do
8          if $e \in susEdge$ && !($e$ is primary input $||e$ is
             primary output) then
9              $troNet \leftarrow troNet + e$ ;
10             break ;
11         end
12     end
13 end
```

## IV. Experiment Results

## A. Experimental Setup

We collect 17 Trust-Hub gate-level Trojan Benchmarks [6], [7] as our dataset, as listed in Table II. We consider all nets inside the Trojan circuit as Trojan nets, and boundary nets are regarded as genuine nets. The framework is programmed in Python language, and the experiments are carried out on a Ubuntu 20.04 server with an Intel i9-10900K CPU running at 3.70GHz and 128 GB memory.

In the isolation forest model, we set the parameter *contamination* to the proportion of Trojan nets in each netlist. The structure of the autoencoder model is shown in Fig. 2. We set the *batch size* to 64 and train ten epochs. A leave-one-out cross-validation method is used while training. The detection performance is evaluated by TPR, TNR, accuracy, and area under the receiver operating characteristic curve (AUROC).

## B. Results and Analysis

Table II shows the final result of our proposed method. Note that s35932-T200 and s38584-T100 have low SCOAP

TABLE II
EXPERIMENT RESULTS OF OUR PROPOSED METHOD (%)

| Circuit | Total Nets (Trojan Nets) | Isolation Forest | | | | Autoencoder | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | TPR | TNR | Accuracy | AUROC | TPR | TNR | Accuracy | AUROC |
| **RS232-T1000** | 311(11) | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| **RS232-T1100** | 312(11) | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| **RS232-T1200** | 315(13) | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| **RS232-T1300** | 307(7) | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| **RS232-T1400** | 311(12) | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| **RS232-T1500** | 314(12) | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| **RS232-T1600** | 312(10) | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| **s15850-T100** | 2445(26) | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| **s35932-T100** | 6420(13) | 100 | 99.98 | 99.98 | 99.99 | 100 | 99.84 | 99.84 | 99.92 |
| **s35932-T200** | 6417(12) | 100 | 100 | 100 | 100 | 83.33 | 100 | 99.97 | 91.67 |
| **s35932-T300** | 6441(36) | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| **s38417-T100** | 5810(11) | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| **s38417-T200** | 5813(11) | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| **s38417-T300** | 5842(44) | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| **s38584-T100** | 7361(8) | 0 | 100 | 99.89 | 50 | 0 | 100 | 99.89 | 50 |
| **s38584-T200** | 7469(126) | 100 | 100 | 100 | 100 | 100 | 99.97 | 99.97 | 99.99 |
| **s38584-T300** | 8486(1143) | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| **Overall** | 64686(1506) | 99.47 | 99.99 | 99.99 | 99.73 | 99.34 | 99.98 | 99.97 | 99.66 |

TABLE III
TOPOLOGY-BASED LOCATION ANALYSIS RESULT (%)

| Model | Circuit | Before | | After | |
|---|---|---|---|---|---|
| | | TNR | Accuracy | TNR | Accuracy |
| **Isolation Forest** | s35932-T100 | 99.97 | 99.97 | 99.98 | 99.98 |
| | s38584-T100 | 99.99 | 99.88 | 100 | 100 |
| | s38584-T300 | 99.95 | 99.95 | 100 | 100 |
| **Autoencoder** | s35932-T200 | 99.94 | 99.91 | 100 | 99.97 |
| | s38584-T200 | 99.84 | 99.84 | 99.97 | 99.97 |
| | s38584-T300 | 99.82 | 99.85 | 100 | 100 |

TABLE IV
COMPARISON WITH SUPERVISED LEARNING METHODS (%)

| Evaluation | [10] | [11] | [12] | Proposed Technique | |
|---|---|---|---|---|---|
| | K-NN | | | Isolation Forest | Autoencoder |
| **Number of features** | 6 | 11 | 5 | 4 | 4 |
| **TPR** | 95.71 | 99.85 | 99 | 99.47 | 99.34 |
| **TNR** | 99.68 | 99.95 | 99.9 | 99.99 | 99.98 |
| **Accuracy** | 99.7 | 99.9 | 99.5 | 99.99 | 99.97 |

values of Trojan nets compared to other circuits. It has been shown in [8] that the two circuits are frequently activated by only applying a few test patterns, which means that they can be found easily by the testing and verification processes and therefore are not likely to appear in the real world. Despite the low SCOAP values of Trojan nets in s35932-T200 and s38584-T100, our method can detect s35932-T200 with 100% accuracy using the isolation forest model and achieves a 99.89% accuracy in s38584-T100. Except for s38584-T100, the proposed method achieves 100% TPR for all circuits, indicating that the method can correctly detect Trojan nets in the circuit.

Table III shows the results before and after topology-based location analysis. According to Table III, after applying the location analysis, we further improve the TNR to 100% in s35932-T200, s38584-T100, and s38584-T300. The result indicates that the topology-based analysis effectively filters out nets misclassified as anomalous by anomaly detection and improves the detection performance. Overall, the proposed technique achieves a constant high TPR and TNR in all circuits tested.

## *C. Discussion*

We compare our approach with several existing testability-based techniques. Although [8], an unsupervised learning method claimed to achieve 100% TPR and TNR in most circuits tested, it failed in testing s35932-T200 and s38584-T100 circuits, while our method can detect s35932-T200 with 100% accuracy. Furthermore, [8] cannot apply to detect Trojan-free circuits. Our method, however, is efficient in classifying all nets in Trojan-free circuits correctly.

Table IV shows the comparison results with supervised learning methods [10], [11], and [12] testing on different numbers of Trust-Hub gate-level benchmarks. According to Table IV, the proposed semi-supervised technique outperforms these supervised learning methods with fewer features. Using the isolation forest model, we obtain an overall 99.47% TPR, 99.99% TNR, and 99.99% accuracy.

## V. CONCLUSION

This paper presents a semi-supervised hardware Trojan detection method at the gate level using anomaly detection. In contrast to most supervised learning methods, the proposed method does not need class label information while training, effectively handles the class imbalance problem, and thus is more pragmatic in real-world situations. Furthermore, we ameliorate the existing computation of SCOAP values and propose a novel topology-based location analysis to improve the detection performance. The proposed method outperforms the existing supervised learning methods with fewer features by achieving an overall 99.47% TPR, 99.99% TNR, and 99.99% accuracy.

## References

[1] L. H. Goldstein and E. L. Thigpen, "SCOAP: Sandia controllability/observability analysis program," in *17th Design Automation Conference*, 1980, pp. 190–196.

[2] K. Hasegawa, M. Oya, M. Yanagisawa, and N. Togawa, "Hardware Trojans classification for gate-level netlists based on machine learning," in *2016 IEEE 22nd International Symposium on On-Line Testing and Robust System Design (IOLTS)*, 2016, pp. 203–206.

[3] K. Hasegawa, M. Yanagisawa, and N. Togawa, "Trojan-feature extraction at gate-level netlists and its application to hardware-Trojan detection using random forest classifier," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2017, pp. 1–4.

[4] Kento Hasegawa, M. Yanagisawa, and N. Togawa, "A hardware-Trojan classification method using machine learning at gate-level netlists based on Trojan features," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E100.A, no. 7, pp. 1427–1438, 2017.

[5] K. Hasegawa, M. Yanagisawa, and N. Togawa, "Hardware Trojans classification for gate-level netlists using multi-layer neural networks," in *2017 IEEE 23rd International Symposium on On-Line Testing and Robust System Design (IOLTS)*, 2017, pp. 227–232.

[6] H. Salmani, M. Tehranipoor, and R. Karri, "On design vulnerability analysis and trust benchmarks development," in *2013 IEEE 31st International Conference on Computer Design (ICCD)*, 2013, pp. 471–474.

[7] B. Shakya, M. T. He, H. Salmani, D. Forte, S. Bhunia, and M. M. Tehranipoor, "Benchmarking of hardware Trojans and maliciously affected circuits," *Journal of Hardware and Systems Security*, vol. 1, no. 1, pp. 85–102, 2017.

[8] H. Salmani, "COTD: reference-free hardware Trojan detection and recovery based on controllability and observability in gate-level netlist," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 2, pp. 338–350, 2017.

[9] P. Dharmadhikari, A. Raju, and R. Vemuri, "Detection of sequential Trojans in embedded system designs without scan chains," in *2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2018, pp. 678–683.

[10] C. H. Kok, C. Y. Ooi, M. Moghbel, N. Ismail, H. S. Choo, and M. Inoue, "Classification of Trojan nets based on SCOAP values using supervised learning," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2019, pp. 1–5.

[11] C. H. Kok, C. Y. Ooi, M. Inoue, M. Moghbel, S. Baskara Dass, H. S. Choo, N. Ismail, and F. A. Hussin, "Net classification based on testability and netlist structural features for hardware Trojan detection," in *2019 IEEE 28th Asian Test Symposium (ATS)*, 2019, pp. 105–1055.

[12] M. Priyadharshini and P. Saravanan, "An efficient hardware Trojan detection approach adopting testability based features," in *2020 IEEE International Test Conference India*, 2020, pp. 1–5.

[13] R. Sharma, N. K. Valivati, G. K. Sharma, and M. Pattanaik, "A new hardware Trojan detection technique using class weighted XGBoost classifier," in *2020 24th International Symposium on VLSI Design and Test (VDAT)*, 2020, pp. 1–6.

[14] S. M. S. Samimi. (2016) Testability measurement tool. [Online]. Available: https://sourceforge.net/projects/testabilitymeasurementtool/

[15] F. T. Liu, K. M. Ting, and Z. H. Zhou, "Isolation forest," in *2008 Eighth IEEE International Conference on Data Mining*, 2008, pp. 413–422.