International Journal of Engineering

Journal Homepage: www.ije.ir

Graph Centrality Algorithms for Hardware Trojan Detection at Gate-Level Netlists

M. Hashemi+[a], A. Momeni+[a], A. Pashrashid[b], S. Mohammadi*[a]

[a] *School of Electrical and Computer Engineering, College of Engineering, University of Tehran, Tehran, Iran*
[b] *Department of Computer Engineering, Sharif University of Technology, Tehran, Iran*

PAPER INFO

Paper history:
Received 12 January 2022
Received in revised form 24 April 2022
Accepted 25 April 2022

Keywords:
Hardware Trojan Detection
Structural Gate-Level Analysis
Graph Centrality Algorithms
Feature Extraction
Feature Selection

ABSTRACT

The rapid growth in the supply chain of electronic devices has led companies to purchase Intellectual Property or Integrated Circuits from unreliable sources. This dispersion in the design to fabrication stages of IP/IC has led to new attacks called hardware Trojans. Hardware Trojans can bargain information, reduce performance, or cause failure. Various methods have been introduced to detect or prevent hardware Trojans. Machine learning methods are one of these. Selecting the type and number of input variables in the learning algorithm has an important role in the performance of the learning model. Some previous hardware Trojan detection studies have used structural gate-level features to create data sets for machine learning models. In this paper, a method based on directed graphs for extracting features is proposed. The proposed method use Graph Centrality Algorithm and structural gate-level features. To examine the importance and the impact of the extracted features with the proposed method, three types of data sets are created as input to the learning model made with XGBoost. The trained learning models based on these three data sets show that extracting graph-based features has improved the F1-score by 10% and the ROC by 22%. The combination of these features with the structural gate-level features improved the F1-score by 17.5% and the ROC by 38.5%.

***doi**: 10.5829/ije.2022.35.07a.16*

1. INTRODUCTION

Hardware Trojan (HT) can be defined as an action meant to change the circuit or cause intentional damage to the circuit in order to change the circuit's functionality or reduce circuit reliability or reveal the circuit information. A simple block diagram of a HT is depicted in Figure , which contains two main parts, including Trigger and payload. The Trojans can be added to the circuit in all producing stages (descriptive IC, design, manufacturing, and test) by the attacker. Hence, the HT detection and confrontation with them are more complicated than fault detection during manufacturing. In the design stage, the Trojan can be inserted easily by changing the hardware description. Therefore, it is critical to identify Trojans in the design stage.

According to the literature, machine learning can be used as one of the effective methods in identifying the HTs. The detection of HTs can be considered as a classification process. Also, gate-level netlists can be used to create features that distinguish between Trojan and normal nets. Therefore, machine learning algorithms and efficient mathematical calculations can make a more accurate classification between Trojan and normal nets.

The distinguishing features of Trojan nets and the normal nets is the key point in this paper in order to

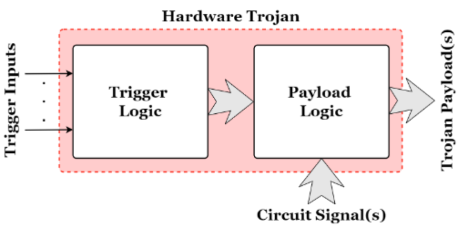


Figure 1. General structure of a HT [1]

*Corresponding Author Institutional Email: smohamadi@ut.ac.ir (S. Mohammadi)
+M. Hashemi and A. Momeni equally contributed in this work

identify HTs using the supervised learning method. If the extracted feature sets are too small, the machine learning cannot cluster and identify the Trojans. In addition, if the feature sets are too large, the Curse of Dimensionality event occurs, and the clustering will not be done in a correct manner. Therefore, it is essential to extract the features of Trojan nets and then, reduce the number of the features so that the model be efficient. Several approaches can be employed to extract the features of the HTs as follows:

- The structural features of the circuit: for instance, the number of fan-ins, the number of the flip-flops, the distance of each net from the primary input or primary output, etc.;
- The test features of the circuit: these features include Sandia Controllability/Observability Analysis Program (SCOAP) parameters [2], like the level of controllability and observability of the circuit nets;
- The Register Transfer Level (RTL)-code features of the circuit: the structure of the RTL code is investigated, and the features like the number of the existing modules in the code, the number of existing signal types in each module, the number of "always" statements in each module, the number of primary input or output variables, and the number of register-type variables, etc. are extracted.

The idea of graph-level features in detecting HTs using machine learning methods can be a starting point for future work and development. While using machine learning to detect HTs, selecting the type and number of input variables of the learning method has a significant role in the performance of the learning model. In previous studies, the structural features of the gate-level netlists have been used to create data sets for the machine learning model used to detect HTs. Extracting some of these features at the gate-level is difficult and requires creating complex data structures from the circuit netlist and then implementing specific algorithms to extract each feature [3]. In this research, two methods based on directed graphs for extracting gate-level features are proposed. The proposed tool does converting a gate-level netlist to a directed graph easily. We use available optimal graph algorithms to work on graphs or extract graph-level features. The contributions of this paper are summarized as follows: 1) as the size of the circuits increases, analyzing the structure of the gate-level netlist and extracting its features becomes more complex. In this research, in order to simplify and increase the efficiency of HT analysis, we used a gate-level netlist mapped to directed graphs; 2) the information of graph is maintained by an efficient data structure to be analyzed. In addition, directed graphs can express more information about the behavior of the circuit, thereby helping detect Trojans; 3) in this study, new approaches have been proposed in order to simplify the extraction of appropriate features, so that the accuracy of the trained model is improved; 4) the proposed method uses Graph Centrality Algorithm and structural gate-level features; 5) to examine the importance and the impact of the extracted features with the proposed method, three types of data sets are created as input to the learning model made with XGBoost [4].

The rest of the paper is organized as follows. Section II provides some background regarding the gate-level features of HT(s) and the issues of the existing approaches. Section III introduces the methodology of the proposed methods as well as the metrics that can help evaluate the quality of detection. Section IV discusses the results in terms of accuracy of detection. Finally, section V concludes this study and suggest some points that can improve the quality of HT detection.

2. BACKGROUND AND RELATED WORK

HT detection at the gate-level is essential in identifying Trojans in the design stage. The identifying methods in the gate-level enable the developers and the System-On-Chip (SOC) designers to test the IPs provided from insecure resources [5]. Using machine learning is one of these approaches that can be utilized to identify HTs. This approach is based on extracting appropriate feature sets for training the model in order to identify Trojans. Extraction or calculation of appropriate features that can increase the accuracy of trained models is complicated and challenging in large gate-level netlists. In this section, the conducted studies in the field of identifying HTs for gate-level netlists will be reviewed.

HT detection methods usually include two stages as follows:

- First, the circuit features are extracted;
- Then, the extracted features are investigated and analyzed using different methods.

According to the analysis methods of the extracted features, the Trojan identification approaches can be divided into three main classes as follows:

- Search-based methods;
- Threshold-based methods;
- Machine-learning-based methods.

In the following subsection, the conducted works in these fields are reviewed briefly.

2. 1. Search-based Identification Methods

In this methods, the netlist of the circuit is processed to find the nets with the Trojan feature. Unused Circuitry Identification (UCI) [6] is a method that is applied to code coverage. However, UCI has also been used to identify HT at the gate-level [7]. It is designed to identify the parts of the circuit that are inactive during the execution or when they are inactive most of the time. UCI algorithm makes a Data-flow graph, in which the nodes are the circuit nets, and the edges indicate the current between the circuit nets. In the next step, it is investigated whether during the simulation the current flows between every two nodes of the corresponding graph. Therefore, the unused nets can be identified in this way. The disadvantage of this method is that it cannot be applied to large circuits because the exhaustive simulation of these circuits is very time-consuming, and sometimes impossible.

VeriTrust is another work in this field [7] that includes a tracer and a checker. The checker identifies the activation history of the SOP and POS sections of the circuit. These inputs are investigated using the checker in three aspects: additional inputs, non-additional inputs, and logic synthesis so that the circuit function is simplified again. This solution requires a white-box accessibility of the hardware IP since it uses static analysis of the code and is based on functional verification. Besides, this method is time-consuming.

2. 2. Threshold Value-based Methods In these approaches, a threshold value is defined for each of the circuit features, and if any net exceeds this threshold limit, the net is recognized as a Trojan. A method called FANCI is presented by Waksman et al. [8] which is based on functional analysis. This method distinguishes the parts of the circuit that are inactive. For this purpose, they proposed a metric called control value(CV) to identify nearly-unused logic. This metric measures the degree of control that an input has on the operation and outputs of a digital circuit. Next, the Trojan nets and the normal nets are identified according to CV and a threshold value. The complexity of CV calculation increases exponentially with the circuit's size. Therefore, the authors have approximated the CV value using innovative techniques. Hence, this method is still time-consuming, and the statistics show that it is not accurate, as it identifies many healthy nets as Trojan nets. Another disadvantage of this method is that it only can be applied to hybrid circuits., Other approaches have also been proposed by Fyrbiak et al. [9] and Sullivan et al. [10] in order to employ this method in hybrid circuits. Nevertheless, this method does not have proper performance in hybrid circuits with high clock levels.

2. 3. Machine Learning-based Methods As machine learning technology grows rapidly, more researchers have become interested in this method to identify the HTs. Machine learning methods have been utilized for identifying HTs at the gate-level for the first time [11]. The gate-level hardware features have been employed to accomplish this as follows:

- $LGFi$ (Logic-gate fan-in): The number of inputs of two previous levels of the gate;
- FFi, FFo: The least distance (level) of each net from the input and output of a flip-flop, respectively;
- PI, PO: The least distance of a net to the primary inputs and outputs of the circuit, respectively.

Figure 2 shows an example of calculating the structural gate-level features, and the values of these five features have been demonstrated for net n. The fan-in value of net n at the second level is four. Since the distance of flip-flop A distance from net n is two, FFi is equal 2.

These features are employed as input in Support Vector Machine (SVM) clustering. The main problem in this method is the long distance between the number of normal nets and the Trojan nets. In order to address this issue and data balancing, the repeated-feature vectors for the healthy nets are eliminated so that the number of the normal nets and the Trojan nets are equaled.

Hasegawa et al. [12] and Ye et al. [13] extracted 51 features of the Trojan nets have been from the netlists of Trust-Hub benchmarks. Salmani et al. [14] and Shakya, 15] applied machine learning in the first stage in order to perform an effective clustering of HTs. In the next step, 11 features (logic gate fan-in, the number of input/output-side net flip-flops, the number of input/output-side net multiplexers, the number of input/output-side net loops, the constant values, and the distance of net from the primary inputs or outputs) have been selected among 51 features in order to reduce the dimensions and prevent dimension congestion event. Random forest classifier has been used to select more essential features.

Ye et al. [13] and Hoque et al. [16] have balanced first the data set by generating new HT data. In this approach, the benchmarks are produced by using Trojan-insertion tools in IP so that the inserted Trojans are difficult to distinguish. Second, the related data to Trojan features are extracted from them, and a trained model is made based on the extracted data set. This trained model is used for identifying HTs.

Salmani [17] utilized combinational testability measured as HT features. By this purpose, the testability indices is calculated using SCOAP. They include controllability and observability of zero and one values of circuit nets. In tsalmani's work [17], an unsupervised method has been used for clustering the Trojans. Some

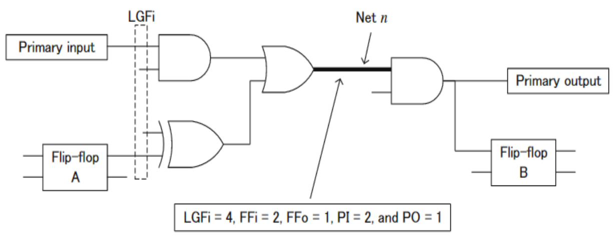


Figure 2. An example of structural gate-level features calculation [11]

Trojans change the sequential nets and signals as well as the circuit hybrid signals. Xie et al. [18] utilized the sequential testability indices as the features of the HTs.

An ensemble learning based method has been proposed by Wang et al. [19] to identify HT features in which the trigger part of Trojans has been used. In this approach, as shown in Figure , two different learning models have been made to identify hybrid and sequential triggers. These two models have been combined using hybrid learning.

Kurihara and Togawa [20] have proposed a 25 hardware-Trojan features based on the structure of trigger circuits for machine-learning-based HT detection. Their experimental results show that the average true positive rate (TPR) and the average true negative rate (TNR) are 63.6% and 100.0%, respectively. However, compared to our proposed method, the average TPR is 15.46% lower.

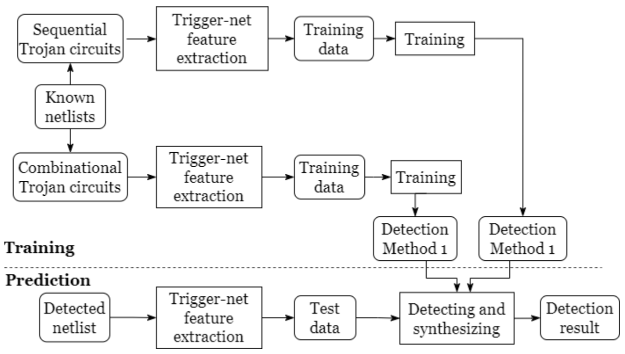
3. METHODOLOGY AND PROPOSED METHOD

Security analysis for Trojan detection in IP cores have been explored mostly in gate-level netlists. In these types of analysis, structural features are extracted to find the hidden structure of Trojans. Considering the size of real circuits, structural analysis to extract features is too complex to be done easily. In this paper, we have used a simple mapping to convert gate-level netlist to a directed graph. This method not only helps to present an efficient analysis, but also provides more information about the functionality of the circuit. Thus, this would be helpful for Trojan detection.

3. 1. Preliminaries

3. 1. 1. Gate-level Netlist to Directed Graph Mapping

The construction of a directed graph of a gate-level netlist is done as follows: we consider the inputs and outputs of each gate as nodes and the gate as an edge, and then the graph based on the relations between the gates in the list. Figure 4 shows directed graph representations for some gate examples.



Sequential Trojan circuits
Trigger-net feature extraction
Training data
Training
Known netlists
Combinational Trojan circuits
Trigger-net feature extraction
Training data
Training
Detection Method 1
Detection Method 1
Training
Prediction
Detected netlist
Trigger-net feature extraction
Test data
Detecting and synthesizing
Detection result

Figure 3. Overall stages of identifying Trojans based on hybrid/sequential triggers [19]

Figure 5 shows the directed graph of the RS232-T1000 Trojan circuit of the Trust-Hub benchmarks [14, 15]. To construct a digraph, using a tool implemented in Python, the netlist is processed to read the individual gates (nets) of the circuit and, depending on the input and output of each gate, it extracts an edge-list file containing all the pairs (input-output) of the gates. It means this is a file containing edges of the directed graph. Then, with this file we can create a graph and display it by graph tools and Python libraries. For example, with a graph library like networkx this edge-list is read and the graph is displayed. Consider an AND gate with a, b as its inputs and c as its output. The developed tool adds edges a-c and b-c to the edge-list file.

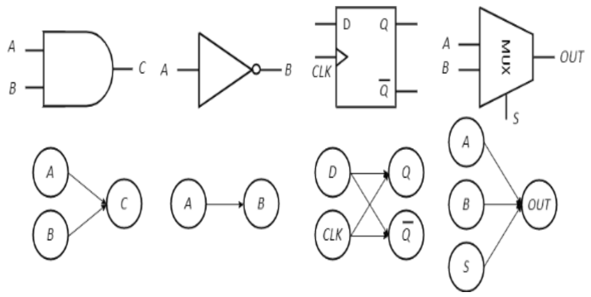


Figure 4. Directed graph representations for some gates

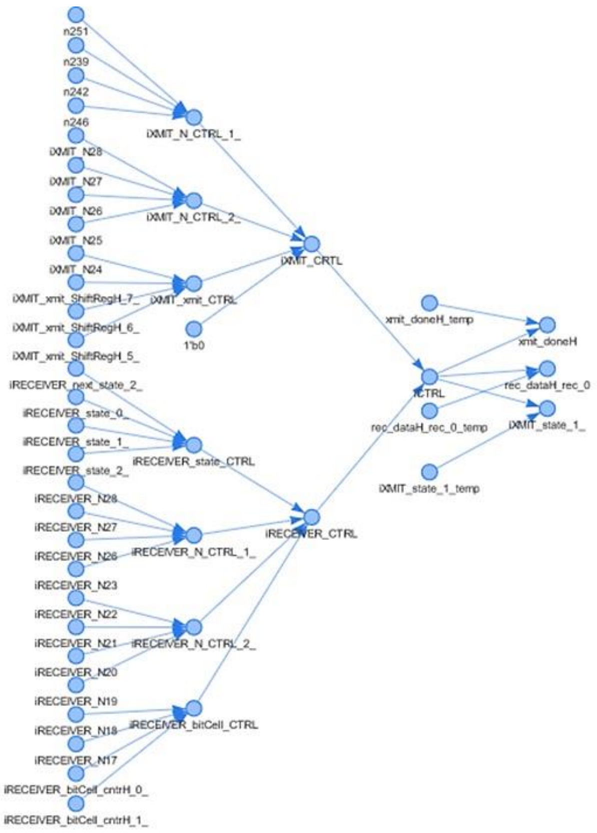


Figure 5. Directed graph of RS232-T1000

3. 1. 2. Graph Centrality Algorithms In graphs or graphs, centrality measures are used to determine the importance of nodes. A centrality measure is a function that assigns a number to each node according to the importance of that vertex in the graph. Extensive research has been done to calculate centrality measures in order to reduce its calculation as much as possible or to be able to perform these calculations in parallel by distributing them on different computers. The most important centrality measures are:

Degree centrality: It is defined as the number of links incident upon a node (i.e., the number of ties that a node has). The most important node has higher degree.

Betweenness centrality: It measures the number of times a node lies on the shortest path between other nodes. In fact, it calculates how many nodes need this node to communicate faster (with less intermediaries). The higher betweenness of a node means more information passes through this node. The betweenness of node v in a graph is calculated as follows [22]:

$$C_B(v) = \sum_{s\neq v\neq t\in V} \frac{\sigma_{st}(v)}{\sigma_{st}} \tag{1}$$

where σ_{st} is total number of shortest paths between node s and node t, and $\sigma_{st}(v)$ is the number of paths between s and t that pass through node v.

Closeness centrality: A node is considered as a close node if it requires a small number of interfaces to communicate with other nodes [23].

Eigen vector centrality: The importance of a node is calculated based on adjacent nodes. If a node is connected to important nodes, its importance also increases under their influence. This method repeatedly considers the importance of neighbors to calculate eigenvector centrality. All nodes are first given an initial score and then, continued in a chain until they reach stability. Scoring in this method is based on the concept that nodes with high connections help the nodes that follow them in terms of eigenvector centrality [24].

PageRank: The rank of a page depends on the rank of the pages that are linked to it. Since we do not initially know the rank of pages, in this algorithm, all pages are first given the same rank as the initial rank. Then, the algorithm is run repetitively until the rank of each page converges to a number. Only nodes that have a neighbor and a link from others to themselves will be ranked, otherwise their rank will be zero [25, 26].

Hub centrality: It is the ability of a node to form a relation with other nodes in a graph.

Authority centrality: It is calculated based on the number of relations that other nodes have with a node.

Clustering coefficient: This criterion calculates the tendency of a node to create a cluster with other nodes. For example, 0.5 for a node means that there is 50% chance of communication between neighboring nodes.

Modularity class: It calculates the effect of a node on other nodes in the community in which it is located and on the nodes in other communities.

3. 2. Preparing Data Set This is the most important step in the supervised machine learning process. The input data set in a learning model must be carefully prepared both in terms of size and quality of its features so that the learning model can be well trained and be used for a high-precision prediction model. In this study, in the first step to prepare a data set for detecting HTs using machine learning, the benchmarks of Table 1 have been collected [14, 15]. The used benchmarks are Verilog-HDL gate-level netlists and we know beforehand which net is a Trojan net and which net is a normal net. Then, the data sets shown in Table 2 have been created.

3. 2. 1. First Data Set: Specified by Graph Centrality Algorithm In the first step, we collect the gate-level benchmarks mentioned in the previous section. In the second step, using our tool written in Python, we start to process each benchmark and generate a .csv file containing the edges of the graph. In the third step, using Gephi tool [29], we read each of these files to calculate the centrality criteria of each graph and generate their report as output. In the last step, we read these generated reports using a tool written in Python to produce a data set containing the

TABLE 1. Properties of used benchmarks [14, 15]

Benchmark	No. of nets		Benchmark	No. of nets	
	Normal	Trojan		Normal	Trojan
RS232-T1000	297	13	s38584-T100	7342	19
RS232-T1100	297	12	s15850-T100	2417	28
RS232-T1200	297	14	s35932-T100	6405	15
RS232-T1300	297	9	s35932-T200	6382	17
RS232-T1400	297	13	s35932-T300	6405	36
RS232-T1500	297	15	s38417-T100	5798	12
RS232-T1600	297	13	s38417-T200	5798	15
			s38417-T300	5827	15

TABLE 2. Data sets used in this work

Data Set	Description
Proposed 1	Based on graph centrality features of directed graphs extracted from benchmarks
SGL	**S**tructural **G**ate-**L**evel features [12, 27, 28]
Proposed 2	Merge features extracted from Prop.1 and SGL

nets of all circuits along with 18 centrality features for each net. Given that in the first step we identify the Trojan sections of each circuit, in this step the labeling of each net would be also done for the data set. We label Trojan nets as “1” and other nets as “0”.

3. 2. 2. Second Data Set: Specified by Structural Gate-Level Features

The use of structural features of gate-level netlist in detection of HTs by machine learning has been studied in several studies. Here, we use the structural gate-level features published [12, 27, 28] to construct the data set. Because we do not have a database based on this type of features, we have written a tool in Python to extract these types of features from the gate-level netlist. Then the information of each net (including input/output to net, type of gate, and a gate-level structural feature vector) has been calculated and stored in the dictionary. After completing dictionary, we store its information as a data set file in the output.

3. 2. 3. Third Data Set: Specified by Structural Gate-Level Features and Graph Centrality Algorithm

In order to add more accuracy to the extracted features of the proposed method with the structural features of the gat-level netlist, we have created a new data set based on the integration of these two types of data sets, which are mentioned in Table 3. Some structural features that are introduced in [12, 27, 28], are:

- Fan_in_x: In case of combinational circuits, trigger circuits require multiple logic gates since they have to implement complex trigger conditions. If the trigger is a rare condition, the number of fan-ins tends to become large. Since hts tend to have rare trigger conditions, the number of fan-ins in Trojan nets must be large compared to normal nets. Hence, fan_in_x that is defined as the number of fan-ins up to x-level away from the net n is an important feature to detect hts.
- In_ff_x (out_ff_x), in_nearestff (out_nearestff): Since the hts circuits are too small and placed locally, the level of flip-flops for sequential-trigger circuits must be small enough. So, in_ff_x that is designed as the number of flip-flops up to x-level away from the input (output) side of the net n, plays an important role to detect hts. Also, the levels of the nearest flip-flops from the input (output) side of the net n are defined as in_nearest flip-flop and out nearest flip-flop, respectively and are extracted as Trojan features.
- In_mux_x (out_mux_x), in_nearestmux (out_ nearestmux): Some hts have multiplexers which receive trigger signals from trigger circuits and switch output signals to activate malfunctions. Therefore, the number of multiplexers up to x-level away from the input side and output side of the net n (in multiplexer x and out multiplexer x, respectively), and the level of the nearest multiplexers from the input side and output side of the net n (in nearest multiplexer and out nearest multiplexer respectively) are extracted as Trojan features.
- In_nearestpi (out_nearestpi): Primary inputs (PI) are often selected as triggers of hts. Primary outputs (PO) are often used as output ports of internal signals for malfunctions. It means Trojan nets are likely to be placed close to pis and pos. So, in_nearestpi (out_nearestpi) that equals the minimum levels from net n to any PI (PO) is extracted as Trojan feature.

Graph centrality features that are used by Gephi tool are introduced by Tarjan [30], hence, we skip detailed introduction. As a brief review:

- Strong component: a strongly connected component of a digraph is a maximal set of vertices that there is a path from any one vertex to any other vertex in the set.

3. 3. Use of Machine Learning

XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. The implementation of the algorithm was engineered for efficiency of compute time and memory resources. A design goal was to make the best use of available resources to train the model. Some key algorithm implementation features include:

- Sparse aware implementation with automatic handling of missing data values;
- Block structure to support the parallelization of tree construction;

TABLE 3. Merging extracted features from Prop.1 and SGL

Graph Centrality Features		**Structural Features ($1 \le x \le 5$) [12, 27, 28]**
Strong Component no.	PageRank	fan_in_x
Component no.	Authority	in_ff_x
weighted degree	degree	out_ff_x
Weighted in Degree	In degree	in_mux_x
Weighted out Degree	Out degree	out_mux_x
Eigen Centrality	Hub	in_nearestFF
Closeness Centrality	Eccentricity	out_nearestFF
Betweenness Centrality		in_nearestPI
Harmonic Closeness Centrality		out_nearestPO
Clustering Coefficient		in_nearestMux
Modularity Class		out_nearestMux

- Continued training to further boost an already fitted model on new data.

In this research, XGBoost learning model is used for classification. The inputs to the model are the data sets with their related set of features and the learning parameters. Default parameters are used in this learning model, some of them are listed in Table 4. The output is the trained model that we use for HT detection. Also, We have used three different methods, XGBoost, Scikit-learn, and SHAP libraries, to analyze the importance and impact of the proposed extracted features of three data sets on the accuracy of HT detection:

- Method 1: Use XGBoost function to determine the importance of features. In XGBoost algorithm, the relative importance of features is measured by several criteria. One of these criteria is the split weight, which is the number of times a feature has been used to separate data in a tree in all boosted trees. More important features are more involved in the construction of trees, and the other features are used to reduce errors;
- Method 2: Use Scikit-learn library for computing permutation importance of features. In fact, we used the feature permutation method that is available in Scikit-learn library to calculate the importance of features. In this method, the increasing rate of the learning model prediction error is measured for each change in features (until the relation between the feature and the correct output is lost). Permuting the feature that has lower importance will not greatly affect the accuracy of the output of model;
- Method 3: Determine the importance of the features based on the calculated Shapley additive explanation (SHAP) values. SHAP is a library that provides a mechanism to calculate Shapley values. In this method that is based on the concept of Shapley values in game theory, the effect of each feature on the output of the learning model is measured using co-operative game theory. Each feature is considered as a player in the game and the output of the predicted model is the final reward of the game. Shapley values determine the role of each player (feature) in the final reward (predicted model).

TABLE 3. Used learning model parameters

Paremeter	Value
base score	0.5
booster	gbtree
Learning rate	0.1
Max depth	3

TABLE 4. Statistical features of the first data set

Feature	mean	std	min	0.25	0.5	0.75	max
in degree	3.0352	1.33	0	2	4	4	6
out degree	3.0352	37.69	0	1	2	3	2553
degree	6.0704	37.65	1	4	5	7	2553
weighted in degree	3.0352	1.33	0	2	4	4	6
weighted out degree	3.0352	37.695	0	1	2	3	2553
weighted degree	6.0704	37.65	1	4	5	7	2553
eccentricity	62.16	37.245	0	37	49	97	171
ClosnessCentrality	0.0528	0.072	0	0.03	0.037	0.06	1
HarmonicClosenessCentrality	0.0623	0.076	0	0.039	0.046	0.071	1
BetweennessCentrality	0.0044	0.014	0	0.001	0.001	0.003	0.37
authority	0.0085	0.015	0	0	0	0.019	0.114
hub	0.0008	0.018	0	0	0	0	0.72
ModularityClass	9.1958	8.79	0	2	7	15	38
ClusteringCoefficient	0.9583	2.75	0	0	0.1	0.3	15
PageRank	0.0003	0.001	0	0	0	0	0.022
component no.	0.0164	0.478	0	0	0	0	22
StrongComponent no.	104.63	149.05	0	37	59	85	1010
EigenCentrality	0.277	0.202	0	0.14	0.24	0.363	1

4. RESULTS AND ANALYSIS

Based on the two proposed methods and the structural features of the gate-level netlist, three data sets on 15 benchmarks have been created in total. Using XGBoost learning model, we have evaluated and compared these data sets and their features. We used Jupyter to develop our Python tool on a Core(TM) i5-4200M CPU machine.

4. 1. Statistical Results of the First Data Set

Table 5 summarizes the statistical characteristics of the first data set with 48,722 rows and 18 columns. The rows of this data set contain all 15 netlists of the Trust-Hub benchmarks [14, 15]. Its columns contain 18 features of graph centrality extracted from directed graphs of benchmarks and one column for the label. Statistical characteristics include mean, standard deviation, minimum, maximum and quartile values of each feature.

As shown in Figure 6, there is a significant difference between the mean features of the normal nets and Trojan nets in this data set. The value distribution of different features for Trojan and normal nets has a significant difference in Figure 7. So, these features can be used to detect Trojan nets using a machine learning model.

4.2 Importance of the First Data Set Features and the Accuracy of the Learning Model

Figure shows the importance of the features using the XGBoost

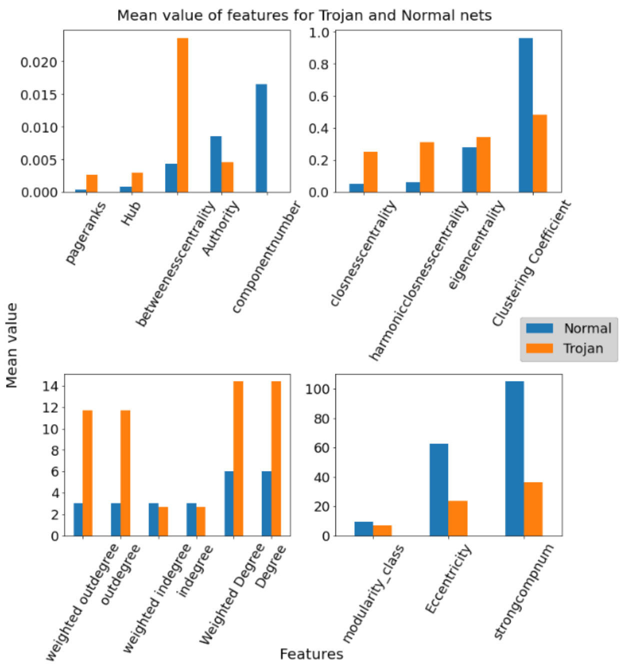


Figure 6. Mean value of features for normal and Trojan nets

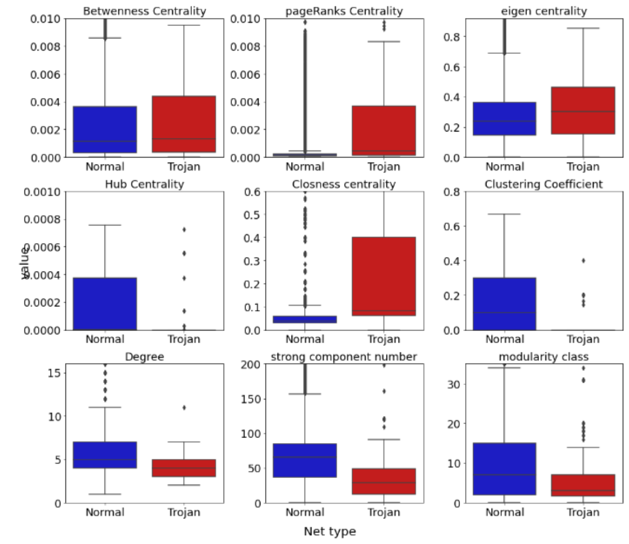


Figure 7. Value distribution of some features in normal and Trojan nets

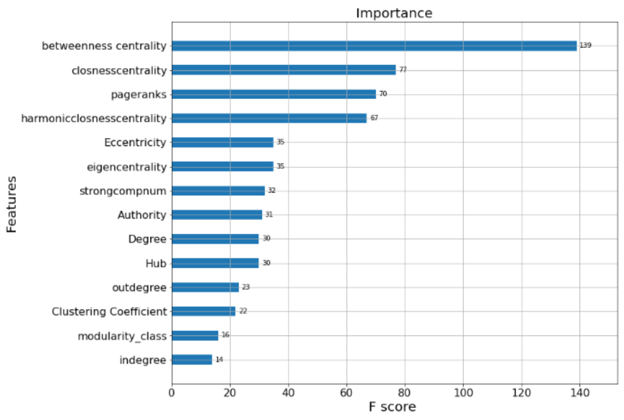


Figure 8. High importance features of the first data set in XGBoost

functions. Sorting these types of features is based on the effectiveness of each feature in improving the F-score value. Figure 9 shows the importance of the features by permutation method using Scikit-learn library. High importance features in XGBoost method are also ranked higher with a slight difference.

Figure 10 shows the importance of the features of this data set. It also shows how each features affects the output, using Shapley values method. Shapley is a unified framework for interpreting predictions that assigns each feature an important value for a particular prediction. According to Shapley values chart, the Hub has the highest importance. The lower value of the Hub per net increases the probability of being a Trojan net. The betweenness centrality feature ranks second and higher value for that means high probability to be Trojan. The betweenness centrality determines how

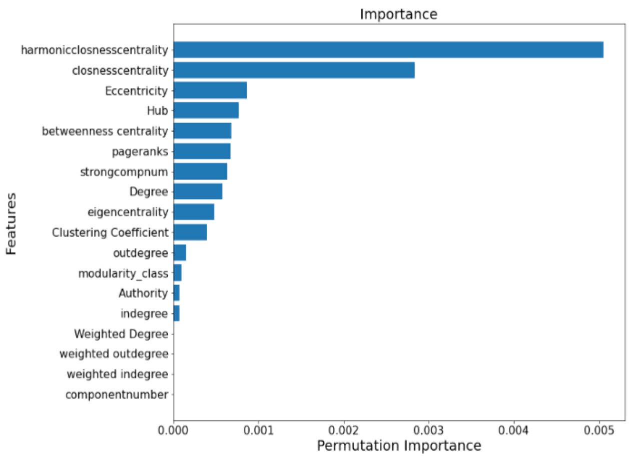


Figure 9. Features of the first data set with permutation and Scikit-learn library

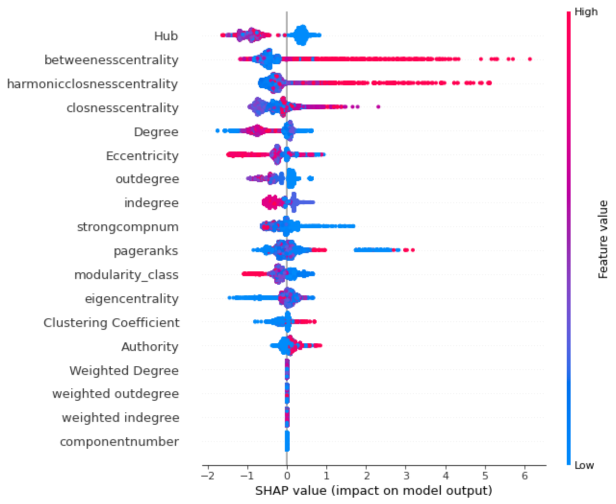


Figure 10. Importance of the features of the first data set by Shapley values

many times a node is in the shortest paths in a graph. Because there is a complex circuit with many inputs before a Trojan net, these nets have higher betweenness centrality in the directed graph. The harmonic closeness centrality is the next important feature. It can be seen that nets with higher centrality features have greater probability of being Trojan nets. This is because Trojan nets are placed farther away from other nets, as a result, their average distance from other nets is greater in directed graph Table 6 and Table 7. show the accuracy of XGBoost learning model trained with the first data set. The experimental results show that the average true positive rate (TPR), and the average true negative rate (TNR) are 66.66 and 99.99%, respectively.

4. 3. Statistical Results of the Second Data Set

Table 8 summarizes the statistical characteristics of the second data set. There are 48,321 rows and 32 columns

TABLE 5. Accuracy of classifying the trained learning model with first data set

Class	Precision	Recall	F1-score	Support
Normal nets	1.00	1.00	1.00	14542
Trojan nets	0.96	0.67	0.79	75

TABLE 6. Confusion matrix based on first data set

Predicted / Actual	Normal	Trojan
Normal nets	14540	2
Trojan nets	25	50

in this data set. The rows of this data set contain all 15 netlists of the benchmarks. Its columns contain 31 gate-level structural features and 1 column for label. Statistical characteristics include mean, standard deviation, minimum, maximum, and quartile values of each feature.

While calculating features such as out-nearestMUX, which indicates the distance of the nearest multiplexer to a net, if there is no multiplexer after a net to the main outputs of the circuit, the value of that feature is set to a constant value (approximately equivalent to the longest path of the circuit).

4. 4. Analysis of the Third Data Set

In previous sections, the importance of structural gate-level features and features based on the graph centrality algorithm were discussed. In addition, the effect of each of these features on the accuracy of the learning model was considered. In this section, the third data set that is a combination of these two data sets is examined in terms of the importance of features and the accuracy of the trained learning model.

Figure 11 shows the features of this data set obtained with the XGBoost functions. As shown in this figure, the centrality features of the graph are mostly in higher order than the structural features of the gate-level. Only out-ff-5 that is a gate-level feature is among the top 10 features. The three most important features in this data set are the graph centrality features (PageRank, harmonic closeness centrality, and betweenness centrality) which are far from the others.

As **Error! Reference source not found.** shows, the centrality features of the graph are mostly in higher order compared to others in the third data set with permutation and Scikit-learn library. In this method, out-ff-5, out-nearestMUX, in-nearestPI, and in-nearestFF features, which are structural gate-level features, have been able to be among the top features.

The importance of the features of the third data set with Shapley values is shown in Figure . In this method, graph centrality features are still in higher

TABLE 8. Statistical features of the second data set

Feature	Count	mean	std	min	0.25	0.5	0.75	max
fan-in-1	48321	3.07	1.37	0	2	4	4	6
fan-in-2	48321	6.31	3.12	0	5	6	8	23
fan-in-3	48321	11.55	6.6	0	7	10	16	58
fan-in-4	48321	20.39	13.21	0	11	19	30	129
fan-in-5	48321	34.86	25.18	0	15	31	52	258
in-mux-1	48321	0	0.026	0	0	0	0	2
in-mux-2	48321	0	0.039	0	0	0	0	2
in-mux-3	48321	0	0.053	0	0	0	0	2
in-mux-4	48321	0	0.073	0	0	0	0	2
in-mux-5	48321	0	0.098	0	0	0	0	2
out-mux-1	48321	0	0.035	0	0	0	0	3
out-mux-2	48321	0	0.059	0	0	0	0	3
out-mux-3	48321		0.08	0	0	0	0	3
out-mux-4	48321	0.1	0.12	0	0	0	0	3
out-mux-5	48321	0.01	0.16	0	0	0	0	3
out-ff-1	48321	1.7	37.81	0	0	1	2	2552
out-ff-2	48321	3.24	37.9	0	0	2	4	2552
out-ff-3	48321	5.9	38.2	0	2	4	7	2553
out-ff-4	48321	10.13	39.13	0	4	7	11	2553
out-ff-5	48321	16.54	41.77	0	7	12	18	2553
in-ff-1	48321	0	0.03	0	0	0	0	2
in-ff-2	48321	3.13	2.05	0	2	3	4	15
in-ff-3	48321	6.26	3.887	0	4	6	8	33
in-ff-4	48321	11.17	7.14	0	7	10	15	69
in-ff-5	48321	18.4	12.52	0	11	16	25	139
out-nearestMUX	48321	58.91	42.4	0	14	99	99	99
out-nearestPO	48321	4.76	5.88	0	3	4	6	99
out-nearestDFF	48321	3.15	12.6	0	0	1	3	99
in-nearestMUX	48321	79.37	35.02	0	99	99	99	99
in-nearestPI	48321	2.27	4.05	0	1	2	3	99
in-nearestDFF	48321	10.24	28.8	0	0	1	2	99
net_type	48321	0.01	0.07	0	0	0	0	1

ranks compared to the structural gate-level features. Positive and negative impact of each feature on the output of the learning model can also be seen in this figure.

Table 9 shows the accuracy of XGBoost learning model trained with third data set, which shows better results than all previous methods. Confusion matrix of this learning model is shown in Table 10. The experimental results demonstrate that the average true positive rate (TPR), and the average true negative rate (TNR) become 79.06 and 100.0%, respectively. It shows this trained learning model improves the average TPR, while keeping the average TNR comparable to the existing state-of-the-art methods.

4. 5. Accuracy of Trained Learning Models with Different Data Sets

Figure 14 compares the accuracy of the trained learning models based on

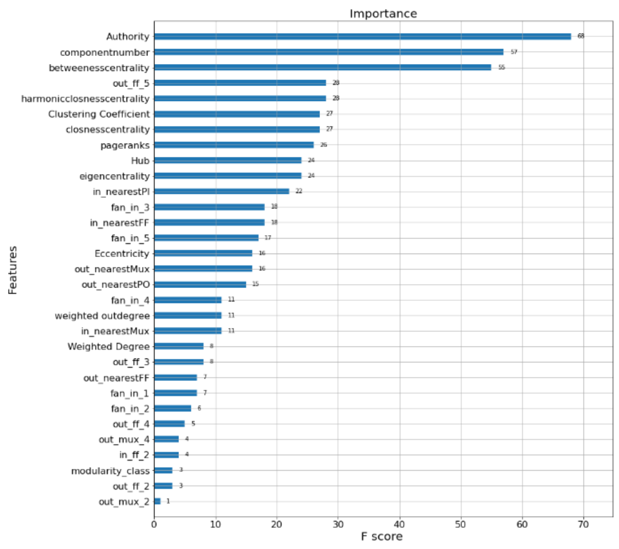


Figure 11. High importance features of the third data set in XGBoost

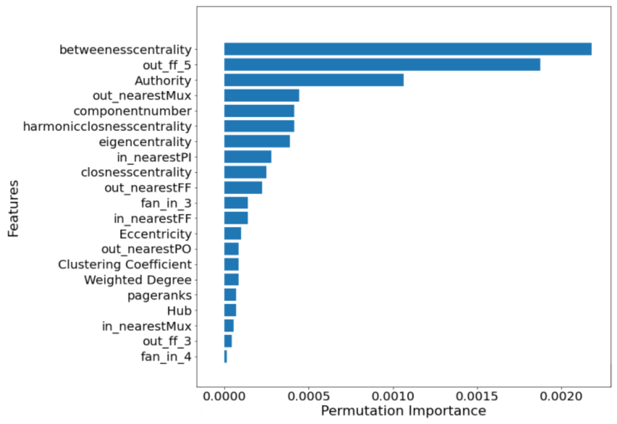


Figure 12. Features of the third data set with permutation and Scikit-learn library

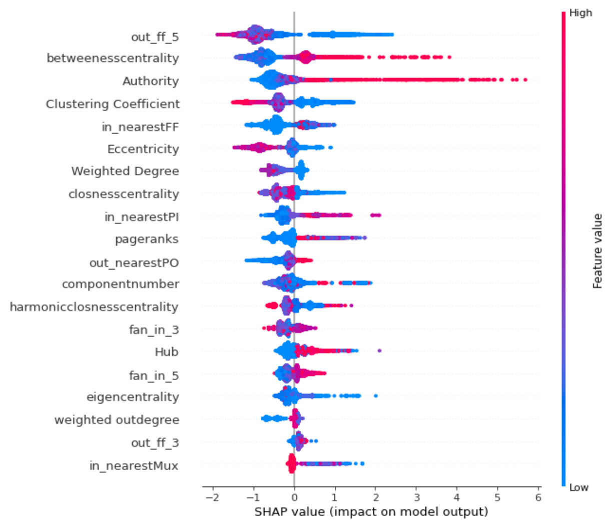


Figure 13. Importance of the features of the third data set by Shapley values

TABLE 9. Accuracy of classifying the trained learning model with third data set

Class	Precision	Recall	F1-score	Support
Normal nets	1.00	1.00	1.00	14400
Trojan nets	1	0.79	0.88	86

TABLE 10. Confusion matrix based on third data set

Actual \ Predicted	Normal	Trojan
Normal nets	14400	0
Trojan nets	18	68

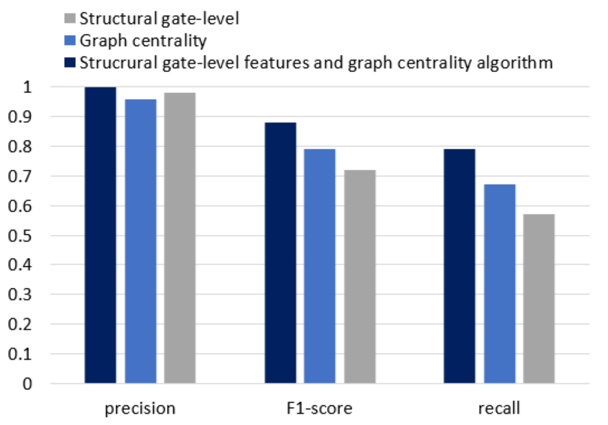


Figure 14. Accuracy of trained learning models with different feature extraction methods

different feature sets. As seen in the figure, the trained learning model has the highest accuracy with a combination of graph centrality and structural gate-level features.

Another method for evaluating performance of binary classification is the Receiver Operating Characteristic (ROC) curve. The performance of binary classifier algorithms is usually measured by parameters called sensitivity or recall. Both of these parameters are combined and displayed as a curve in the ROC diagram. Figure shows ROC diagram of three proposed data sets. As it turns out, the learning model based on combining graph centrality features and structural gate-level features has better ROC with $AUC = 1$.

4. 6. Runtime Overhead

Since Hasegawa et al. [12], liu et al. [27] and Kurihara et al. [28] did not report the execution time of each feature for the benchmarks and we did not have access to those functions, we wrote the calculation functions of those features based on the description presented by et al. [12], liu et al. [27] and Kurihara et al. [28] to be able to provide comparison. Some of the functions were too complex with long

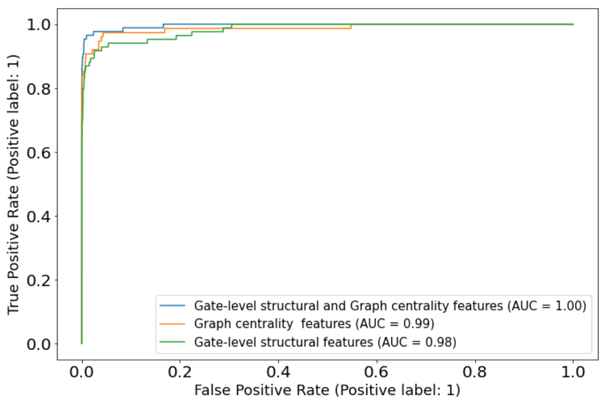


Figure 15. ROC diagrams of trained learning models with three data sets

execution time and motivated us to present graph centrality features to be able to use optimal graph algorithms available in Gephi tool with a very convenient time complexity.

Time complexity of the training phase is directly related to the size of the data set. In all used data sets, the number of rows is the same. Our proposed data set (based on graph centrality features) has a smaller number of features, and as a result, the step of calculating the importance of features and selecting them is done in less time.

5. CONCLUSION AND FUTURE WORKS

In this paper, a method based on directed graphs for extracting features is proposed. The proposed method use Graph Centrality Algorithm and structural gate-level features. To examine the importance and the impact of the extracted features with the proposed method, three types of data sets are created as input to the learning model made with XGBoost. The trained learning models based on these three data sets shows that extracting graph-based features has improved the F1-score by 10% and the ROC by 22%. The combination of these features with the structural gate-level features improved the F1-score by 17.5% and the ROC by 38.5%.

In the proposed method, Gephi tool is used to extract the graph-level features. To integrate the steps of creating a data set and training the learning model and then HT detection, a suitable library in Python can be used to calculate these features. Also, an API can be designed to communicate with Gephi tool.

To improve the accuracy of HT detection, graph embedding methods or graph node classification can be used. In this way, first the directed graph should be extracted. Then, a set of features would be assigned to each nodes in the graph. This set of features can be obtained in the following ways for each node in graph:

- Using the Node2vec method;
- Calculate the structural features of the gate-level and assign these features to graph nodes.

After performing the above steps, we will have a multigraph where a feature vector is assigned to each its node. Thereby, it is possible to use different methods of node classification such as GCN and GraphSAG to classify Trojan nodes.

6. REFERENCES

1. Bhunia, S., Hsiao, M.S., Banga, M. and Narasimhan, S., "Hardware trojan attacks: Threat analysis and countermeasures", ***Proceedings of the IEEE***, Vol. 102, No. 8, (2014), 1229-1247, doi: 10.1109/JPROC.2014.2334493.
2. Goldstein, L.H. and Thigpen, E.L., "Scoap: Sandia controllability/observability analysis program", in Proceedings of the 17th Design Automation Conference., (1980), 190-196, doi: 10.1145/800139.804528.
3. Esfandian, N. and Hosseinpour, K., "A clustering-based approach for features extraction in spectro-temporal domain using artificial neural network", ***International Journal of Engineering,Transactions B: Applications***, Vol. 34, No. 2, (2021), 452-457, doi: 10.5829/IJE.2021.34.02B.17.
4. Chen, T. and Guestrin, C., "Xgboost: A scalable tree boosting system", in Proceedings of the 22nd acm sigkdd international conference on knowledge discovery anionsd data mining. Vol., No. Issue, (2016), 785-794, doi: 10.1145/2939672.2939785.
5. Yang, Y., Ye, J., Cao, Y., Zhang, J., Li, X., Li, H. and Hu, Y., "Survey: Hardware trojan detection for netlist", in 2020 IEEE 29th Asian Test Symposium (ATS), IEEE. , (2020), 1-6, doi: 10.1109/ATS49688.2020.9301614.
6. Hicks, M., Finnicum, M., King, S.T., Martin, M.M. and Smith, J.M., "Overcoming an untrusted computing base: Detecting and removing malicious hardware automatically", in 2010 IEEE symposium on security and privacy, IEEE., (2010), 159-172, doi: 10.1109/SP.2010.18.
7. Zhang, J., Yuan, F., Wei, L., Liu, Y. and Xu, Q., "Veritrust: Verification for hardware trust", ***IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems***, Vol. 34, No. 7, (2015), 1148-1161, doi: 10.1109/TCAD.2015.2422836.
8. Waksman, A., Suozzo, M. and Sethumadhavan, S., "FANCI: Identification of stealthy malicious logic using boolean functional analysis", in Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security., (2013), 697-708, doi: 10.1145/2508859.2516654.
9. Fyrbiak, M., Wallat, S., Swierczynski, P., Hoffmann, M., Hoppach, S., Wilhelm, M., Weidlich, T., Tessier, R. and Paar, C., "Hal—the missing piece of the puzzle for hardware reverse engineering, trojan detection and insertion", ***IEEE Transactions on Dependable and Secure Computing***, Vol. 16, No. 3, (2018), 498-510, doi: 10.1109/TDSC.2018.2812183.
10. Sullivan, D., Biggers, J., Zhu, G., Zhang, S. and Jin, Y., "Fight-metric: Functional identification of gate-level hardware trustworthiness", in Proceedings of the 51st Annual Design Automation Conference., (2014), 1-4, doi: 10.1145/2593069.2596681.
11. Hasegawa, K., Oya, M., Yanagisawa, M. and Togawa, N., "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), IEEE., (2016), 203-206, doi: 10.1109/IOLTS.2016.7604700.

12. Hasegawa, K., Yanagisawa, M. and Togawa, N., "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), IEEE., (2017), 1-4, doi: 10.1109/ISCAS.2017.8050827.
13. Ye, J., Yang, Y., Gong, Y., Hu, Y. and Li, X., "Grey zone in pre-silicon hardware trojan detection", in 2018 IEEE International Test Conference in Asia (ITC-Asia), IEEE., (2018), 79-84, doi: 10.1109/ITC-Asia.2018.00024.
14. Salmani, H., Tehranipoor, M. and Karri, R., "On design vulnerability analysis and trust benchmarks development", in 2013 IEEE 31st international conference on computer design (ICCD), IEEE., (2013), 471-474, doi: 10.1109/ICCD.2013.6657085.
15. Shakya, B., He, T., Salmani, H., Forte, D., Bhunia, S. and Tehranipoor, M., "Benchmarking of hardware trojans and maliciously affected circuits", ***Journal of Hardware and Systems Security***, Vol. 1, No. 1, (2017), 85-102, doi: 10.1007/s41635-017-0001-6.
16. Hoque, T., Cruz, J., Chakraborty, P. and Bhunia, S., "Hardware ip trust validation: Learn (the untrustworthy), and verify", in 2018 IEEE International Test Conference (ITC), IEEE., (2018), 1-10, doi: 10.1109/TEST.2018.8624727.
17. Salmani, H., "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, (2016), 338-350, doi: 10.1109/TIFS.2016.2613842.
18. Xie, X., Sun, Y., Chen, H. and Ding, Y., "Hardware trojans classification based on controllability and observability in gate-level netlist", ***IEICE Electronics Express***, Vol. 14, No. 18, (2017), 20170682-20170682, doi: 10.1587/elex.14.20170682.
19. Wang, Y., Han, T., Han, X. and Liu, P., "Ensemble-learning-based hardware trojans detection method by detecting the trigger nets", in 2019 IEEE International Symposium on Circuits and Systems (ISCAS), IEEE., (2019), 1-5, doi: 10.1109/ISCAS.2019.8702539.
20. Kurihara, T. and Togawa, N., "Hardware-trojan classification based on the structure of trigger circuits utilizing random forests", in 2021 IEEE 27th International Symposium on On-Line Testing and Robust System Design (IOLTS), IEEE., (2021), 1-4, doi: 10.1109/IOLTS52814.2021.9486700.
21. Hagberg, A., Swart, P. and S Chult, D., *Exploring network structure, dynamics, and function using networkx*. 2008, Los Alamos National Lab.(LANL), Los Alamos, NM (United States).
22. Brandes, U., "A faster algorithm for betweenness centrality", ***Journal of Mathematical Sociology***, Vol. 25, No. 2, (2001), 163-177, doi: 10.1080/0022250X.2001.9990249.
23. Chen, D., Lü, L., Shang, M.-S., Zhang, Y.-C. and Zhou, T., "Identifying influential nodes in complex networks", ***Physica a: Statistical mechanics and its applications***, Vol. 391, No. 4, (2012), 1777-1787, doi: 10.1016/j.physa.2011.09.017.
24. Rodrigues, F.A., "Network centrality: An introduction", in A mathematical modeling approach from nonlinear dynamics to complex systems. 2019, Springer. 177-196, doi: 10.48550/arXiv.1901.07901.
25. Upstill, T., Craswell, N. and Hawking, D., "Predicting fame and fortune: Pagerank or indegree?", (2003).
26. Jaderyan, M. and Khotanlou, H., "Automatic hashtag recommendation in social networking and microblogging platforms using a knowledge-intensive content-based approach", ***International Journal of Engineering, Transactions B: Applications***, Vol. 32, No. 8, (2019), 1101-1116, doi: 10.5829/IJE.2019.32.08B.06.
27. Liu, Q., Zhao, P. and Chen, F., "A hardware trojan detection method based on structural features of trojan and host circuits", ***IEEE Access***, Vol. 7, (2019), 44632-44644, doi: 10.1109/ACCESS.2019.2908088.
28. Kurihara, T., Hasegawa, K. and Togawa, N., "Evaluation on hardware-trojan detection at gate-level ip cores utilizing machine learning methods", in 2020 IEEE 26th International Symposium on On-Line Testing and Robust System Design (IOLTS), IEEE., (2020), 1-4, doi: 10.1109/IOLTS50870.2020.9159740.
29. Bastian, M., Heymann, S. and Jacomy, M., "Gephi: An open source software for exploring and manipulating networks", in Third international AAAI conference on weblogs and social media., (2009) , doi: 10.13140/2.1.1341.1520.
30. Tarjan, R., "Depth-first search and linear graph algorithms", ***SIAM Journal on Computing***, Vol. 1, No. 2, (1972), 146-160, doi: 10.1137/0201010.
31. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R. and Dubourg, V., "Scikit-learn: Machine learning in python", ***the Journal of Machine Learning Research***, Vol. 12, (2011), 2825-2830, doi: 10.5555/1953048.2078195.
32. Lundberg, S.M. and Lee, S.-I., "A unified approach to interpreting model predictions", in Proceedings of the 31st international conference on neural information processing systems., (2017), 4768-4777, doi: 10.48550/arXiv.1705.07874.

Persian Abstract

چکیده

پراکندگی موجود در مراحل طراحی تا ساخت مدارهای مجتمع، به حمله‌های جدیدی به نام تروجان سخت‌فزاری در زنجیره تولید مدارهای مجتمع منجر شده است. این حمله‌ها می‌توانند اطلاعات رمزنگاری شده را سرقت کنند، عملکرد مدار را کاهش دهند یا موجب خرابی کلی تراشه شوند. روش‌های مختلفی برای تشخیص یا جلوگیری از تروجان‌های سخت‌افزاری معرفی شده که استفاده از یادگیری ماشین یکی از این روش‌ها بوده که اخیرا مورد توجه بسیار قرار گرفته است. در پژوهش‌های قبلی از ویژگی‌های ساختاری سطح گیت برای ایجاد مجموعه داده در مدل یادگیری ماشین و تشخیص تروجان‌های سخت‌افزاری استفاده شده است. در این مقاله دو روش مبتنی بر گراف‌های جهت‌دار برای استخراج ویژگی‌ها پیشنهاد شده است: روش اول استفاده از معیارهای مرکزیت گراف و روش دوم ترکیب این معیارها با ویژگی‌های ساختاری سطح گیت است. بر اساس این دو روش پیشنهادی و روش استخراج ویژگی‌های ساختاری سطح گیت، در مجموع سه مجموعه داده متفاوت ایجاد شده است. برای بررسی میزان اهمیت و تاثیر ویژگی‌های استخراج شده با روش‌های پیشنهادی و مقایسه آنها با روش استخراج ویژگی‌های سطح گیت، سه نوع مجموعه داده ایجاد شده به عنوان ورودی به مدل یادگیری ساخته شده با XGBoost داده شد. بررسی دقت مدل‌های یادگیری آموزش دیده براساس این سه مجموعه داده نشان داد که روش استخراج ویژگی‌های مبتنی بر مرکزیت گراف، معیار F1-score را به میزان ۱۰ درصد و معیار ROC را به میزان ۲۲ درصد بهبود داده است. ترکیب این ویژگی‌ها با ویژگی‌های ساختاری سطح گیت، معیار F1-score را به میزان ۱۷.۵ درصد و معیار ROC را به میزان ۳۸.۵ درصد بهبود داده است.