Journal of Advanced Research in Applied Sciences and Engineering Technology

Journal homepage:
https://semarakilmu.com.my/journals/index.php/applied_sciences_eng_tech/index
ISSN: 2462-1943

Improving Hardware Trojan Detection Coverage by Utilizing Features at Different Abstraction Levels

Hau Sim Choo[1], Chia Yee Ooi[1*], Nordinah Ismail[1], Michiko Inoue[2], Chee Hoo Kok[1]

[1] Malaysia–Japan International Institute of Technology, Universiti Teknologi Malaysia, Kuala Lumpur, Malaysia
[2] Nara Institute of Science and Technology, Ikoma, Nara, Japan

ARTICLE INFO

Article history:
Received 25 March 2023
Received in revised form 14 July 2023
Accepted 20 July 2023
Available online 30 August 2023

Keywords:
Hardware Trojan detection; machine learning; integrated circuit; register-transfer level; gate level

ABSTRACT

In this paper, we introduced a solution to improve hardware Trojan (HT) detection coverage by analyzing features at different abstraction levels. We demonstrated our solution with a supervised classification of HT branching statement (BS) in register-transfer-level (RTL) description. The proposed classifier was trained with a double-abstraction-level feature vector consisting of features extracted at RTL and gate level (GL). In the experiment, we evaluated the HT detection coverage of the trained classifier by applying them on 24 self-designed HT circuits. The proposed classifier achieved the highest 87.5% HT detection coverage with 81.25% true positive rate (TPR), 88.44% true negative rate (TNR), and 88.24% accuracy (ACC). The result proved that the double-abstraction-level feature vector outperformed the single-abstraction-level feature vector with a higher HT detection coverage.

1. Introduction

In the semiconductor industry, a term “hardware Trojan” (HT) was coined to refer to a circuitry that is secretly inserted into an integrated circuit (IC) and attempts to attack the system at which the HT resides [1]. The HT can launch harmful attacks such as changing the circuit operation, leaking the critical information, degrading the circuit performance, and a Denial-of-Service attack [2]. The presence of HT causes the trustworthiness issue of ICs.

One of the HT countermeasures is detection. Among the previously proposed detection methods, the machine-learning-based approach is relatively popular, especially for pre-silicon detection. Its self-learning ability helps in building an analytical model without an explicit programming. The model can even be expanded by fitting it with a larger database. Due to these reasons, machine learning can reduce the effort of HT detection. However, we cannot apply only single approach to detect all HTs because of the diversity of HT types. Many HT detection methods have been introduced with each of them having different motivations and different HTs covered.

* *Corresponding author*
E-mail address: ooichiayee@utm.my

https://doi.org/10.37934/araset.32.1.7386

HT detection is preferable to be conducted during pre-silicon stage due to the difficulty of HT removal from a physically fabricated circuit. To prevent unnecessary investment in an infected circuit design, it is always better to identify the HT before the circuit design is mapped into later stages of design process. In addition, the information contained in the circuit design at every stage is different, it could be harder to trace the HT when the circuit design undergoes more mappings at lower abstraction levels. Due to these reasons, register-transfer level (RTL) is an early design stage which is suitable for conducting HT detection. However, conversion of RTL circuit functionality to quantitative data is a challenge.

This paper suggests a solution to HT detection coverage improvement by utilizing double-abstraction-level features. We demonstrated our solution with an RTL branching statement (BS) classification method and the proposed features extracted at both RTL and gate level (GL). To our best knowledge, we are the first to extract GL features from RTL instances for HT detection.

We first discuss on the related HT detection methods and emphases on the differences between each method. Then, we discuss the proposed solution framework including the proposed features, training and validation of classifiers, and evaluation of the proposed solution. Lastly, the experimental results are presented, and discussions are made between previously proposed methods and our method.

2. Related Works

In 2016, a GL HT net classification method was proposed by Hasegawa *et al.,* [3]. A support-vector machine (SVM) was implemented to detect HT nets based on their structural features. It achieved 83% true positive rate (TPR) and 49% true negative rate (TNR). The proposed method was further evaluated by Inoue *et al.,* [4]. The result showed that the method was effective against conditionally triggered HTs but not effective against always-on HTs. Another set of HT net structural features was introduced by Inoue *et al.,* [5] which achieved at least 98.2% TNR on every circuit it tested using a random-forest classifier. The proposed features were adopted by Hasegawa *et al.,* [6] to be implemented with a multi-layer neural network, and it achieved 85% TPR and 70% TNR. The study was further explored by Inoue *et al.,* [7] to optimize the neural network classifier. The experiment result showed that the classifier managed to achieve 72.9% TPR and 90% TNR. Another GL HT net classification method based on net structural features was introduced by Kurihara and Togawa [8] which achieved 63.6% TPR and 100% TNR.

A GL net HT detection method based on net testability was proposed by Salmani [9]. The author employed unsupervised k-means clustering approach to categorize nets into three clusters of possibly genuine nets, HT nets with poor controllability, and HT nets with poor observability, respectively. The proposed method managed to achieve 100% accuracy (ACC) in 21 out of 23 tested HT benchmark circuits.

A GL netlist classification method was proposed by Xie *et al.,* [10] based on both net structural and testability features. The experimental result showed an 100% ACC in which every circuit was correctly classified. Another study of GL net classification based on both types of features was introduced by Kok *et al.,* [11]. The proposed method achieved 99.85% TPR, 99.95% TNR, and 99.90% ACC by using an k-nearest neighbors (k-NN) classifier.

A GL logic classification using an unsupervised Density-Based Spatial Clustering of Applications with Noise (DBSCAN) was introduced by Lu *et al.,* [12] based on information entropy of signal waveforms, and it obtained 79% TPR and 99% TNR.

An RTL signal classification based on RTL description's abstract syntax tree information was introduced by Han *et al.,* [13]. Gradient boosting classifier was employed, and it achieved 100% TPR

and 89.32% TNR. An RTL description classification method using graph neural network (GNN) based on data flow graph was introduced by Yasaei *et al.,* [14], and it achieved 97% TPR.

Based on our literature review, we find that RTL machine-learning-based HT classification method is less explored compared to GL. In addition, no existing study focuses on improving HT detection coverage.

3. HT Classification with RTL and GL Features

The proposed HT detection method utilizes a double-abstraction-level feature vector and a supervised machine learning classifier to identify HT BSs in RTL description. BS refers to any statement that involves conditional operator such as if and if-else, and case. The classification is based on the proposed BS features extracted at both RTL and GL.

The framework of our proposed solution is presented in Figure 1. The samples of BSs are collected from the HT benchmark circuits provided by Trust-Hub [2]. A feature vector consisting of RTL and GL features are extracted from each BS. A label of genuine or HT class is given to each feature vector. After dataset balancing, the collected data forms a training dataset and is used to train classifiers. After performance validation, the selected trained classifier is further evaluated to determine their HT detection coverage.

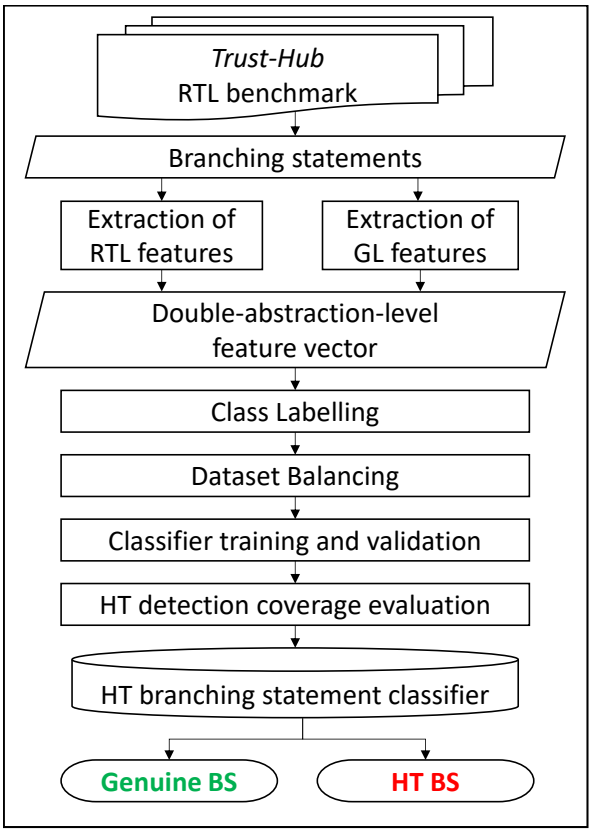

Fig. 1. Our proposed solution framework

3.1 BS Features Extraction

The four (4) RTL features were previously proposed by Choo *et al.,* [15] and are defined based on branching probability and control dependency index which quantifies the difficulty of transition of a

variable from one to another state; the six (6) GL features are related to combinational testability measures based on Sandia Controllability/Observability Analysis Program (SCOAP) [16].

3.1.1 RTL features

3.1.1.1 Branching probability

HTs are usually activated under rare conditions [17]. In other words, HT statements in RTL are expected to have a lower chance of execution. By assuming that the chance of a statement being executed at RTL corresponds to the probability of entering the BS under the specified condition, branching probability could be analyzed for detecting HTs.

Branching probability (P) is defined as the probability of executing a BS

$$\mathrm{P} = \frac{|\mathrm{COND}|}{2^{\mathrm{N_{cond}}}} \tag{1}$$

where COND is a set of binary values that satisfy the branching condition when they are assigned to the conditional variable, and Ncond is the number of bits that compose the conditional variables. For instance, a branching condition (A==2'b01&&B<2'b11) has COND = {0100, 0101, 0110} and Ncond = 4 where the first two bits belong to variable A and the following two bits belong to variable B. The lower the P, the more difficult the BS will be taken.

Pouter of a BS is the effective branching probability of the preceding level BS (which is applicable to any nested BS). For BS at the outermost level, Pouter is, by default, assigned to 1 which is the maximum value indicating that the preceding level BS is always taken.

Effective branching probability of a BS (Pe) refers to the effective probability of a BS being taken instead of other BSs at the same branching level. Pe of a BS is dependent of Pouter and is defined as

$$\mathrm{P_e} = \mathrm{P_{outer}} \times \mathrm{P} \tag{2}$$

where P is the branching probability of the BS being evaluated.

Relative branching probability (RP) of a BS is defined as the branching probability of the BS with respect to the highest branching probability of any BS at the same branching level

$$\mathrm{RP} = \frac{\mathrm{P}}{\mathrm{P_{most}}} \tag{3}$$

where P¬¬most represents the highest branching probability of the BS at the same level with the BS under evaluation. Low relative branching probability indicates that the BS is suspicious because the branching condition is difficult to be fulfilled, compared to other BSs at the same branching level.

Since P can be derived from Pouter and Pe using Eq. (2), it could be considered as a redundant feature. Only Pouter, Pe and RP are selected to be used as the input features for HT detection.

3.1.1.2 Control dependency index

On one hand, a sequential HT may need to transition through several register states before the HT is activated. On the other hand, the HT triggering variable tends to stay in the same register state for a long duration before proceeding to the final state to activate. This HT characteristic could be observed at a BS that uses control dependent variable in the branching condition along with an additional difficult branching condition. Control dependent variable refers to a variable which is an

output variable(s) of the block (variable being used as the left-hand-side variable by any statement in any BS within the block) and is also being used in the BS's condition. If the control dependent variable is also the state of the sequential block, the variable may determine the number of transitions of the states before the HT activates. One example is as shown below

if(A==2'b01 && B==6'b110110) A=2'b10;

Variable A is a control dependent variable which requires itself to be 2'b01 to transition to state 2'b10 with an additional requirement of (B==6'b110110) making the transition of the states more difficult. This kind of control dependent variable has a difficult sequential transition pattern due to the additional requirement. It could be part of the HT trigger signals, especially the sequential HT. Control dependency index (Cdep) of a BS is introduced to quantify the difficulty of the control dependent variable to trigger transition from one to another state, and is defined as

$$C_{dep} = \begin{cases} 1, & N_{dep} = 0 \\ \frac{N_{dep}}{N_{cond}}, & N_{dep} > 0 \end{cases} \quad (4)$$

where Ndep is the number of bits of the control dependent variable in a branching condition; Ncond is the total number of bits that compose the conditional variables in the BS. If Ndep is 0, Cdep is taken as 1, which means there is no control dependency between the conditional operator and the output variables of the block. Otherwise, Cdep equals to the ratio of Ndep to Ncond.

The lower the Cdep, the more difficult it is for the variable to transition from one state to another state. This also means that the variable tends to stay in the same state for a long period, which is one of the characteristics of hardware HTs. Thus, Cdep is used as one of the proposed features for HT classification.

3.1.2 GL features

Goldstein *et al.,* [16] estimates and quantifies the difficulties of controlling and observing a signal in a net. For conventional testing, these measurements are desired to be low to reduce the complexity of test generation. However, depending on the inserted HTs, some HT related nets could be high in these measurements as the HTs usually want to bypass the testing. In this paper, SCOAP is used to measure three measurements, combinational 0-controllability (CC0), combinational 1-controllability (CC1), and combinational observability (CO). We further analyze the measurements and process them into the six (6) features representing a BS.

First, the given RTL description is compiled into a dummy GL netlist without any extra design constraints or optimization to avoid any further difficulty of HT detection. During this compilation, the RTL variables are converted into either a net, a primary input, or a primary output. RTL variables refer to the variables defined as reg (registers) and net (nets) in the RTL description written in Verilog HDL. From here, the set of nets corresponding to RTL variables is defined as SRTL, which is a subset of the set of all nets, S

$$S_{RTL} \subseteq S \quad (5)$$

We use Synopsys TetraMax to calculate the SCOAP measurements. In theory, the combinational SCOAP measurements can be ranging from 1 to infinity for CC0 and CC1, 0 to infinity for CO. However,

Synopsys TetraMax has a limitation that the measured values for CC0, CC1, and CO have a threshold of 254 where any value higher than 254 will be replaced by the symbol of asterisk (*). This threshold is easily exceeded especially by large circuits.

D flip-flop is a sequential logic that complicates the combinational testability, resulting in a high SCOAP measurement value. In this paper, we simplify the previously obtained GL netlist by removing all D flip-flops under the assumption of full scan technique being applied to the circuit. The removed D flip-flops' data input nets and data output nets are then converted to a new primary outputs and inputs, respectively. By doing so, we can simplify the SCOAP computation by just considering combinational circuit parts. With D flip-flop removal, the issue of exceeding threshold is resolved and the SCOAP measurement becomes possible with generally all values below 254.

After obtaining the SCOAP measurements of each net, we extract all the nets of SRTL for further analysis and filter out any net which does not belong to SRTL. For each SRTL net, the magnitude of combinational controllability (CC) is calculated, which indicates the degree of difficulty to control a net to either 0 or 1. The CC of a net w in SRTL is defined in terms of its CC0 and CC1 as

$$\mathrm{CC(w)} = \sqrt{\mathrm{CC0(w)}^2 + \mathrm{CC1(w)}^2},\ \mathrm{w} \in \mathrm{S_{RTL}} \tag{6}$$

For each BS, the CCs of each SRTL net that correspond to the RTL variables being used in branching condition are summed up to be one of the proposed GL features, ΣCC. If ΣCC is high, the BS is suspected to have a hard-to-achieve condition that could be possibly used as a rare HT activation event. ΣCC is defined as

$$\Sigma\mathrm{CC} = \sum_{\mathrm{i=1}}^{\mathrm{N_{cond}}} \mathrm{CC(w_i)}\,, \mathrm{w_i} \in \mathrm{S_{RTL}} \tag{7}$$

where w_i is the net corresponding to bit i of the conditional variables.

In a similar way, another BS feature ΣCO is calculated using CO. If ΣCO is high, the BS is suspected to use hard-to-observe variables in the branching condition to hide the trigger signal or the activation condition from the detection

$$\Sigma\mathrm{CO} = \sum_{\mathrm{i=1}}^{\mathrm{N_{cond}}} \mathrm{CO(w_i)}\,, \mathrm{w_i} \in \mathrm{S_{RTL}} \tag{8}$$

HT BS is always expected to have much higher CC or CO values than the genuine class BS. Therefore, the HT can be considered as the outlier [9]. A simple way to determine the outlier is by using the mean and the standard deviation of the samples. By measuring the difference between the feature value of a sample and its standard deviation as well as the mean, the outlier can be determined when the difference value is beyond a given threshold. However, the threshold is unable to be confirmed because it would vary according to circuit and HT design. Instead, we utilize the mean and the standard deviation of CC and CO directly as our proposed GL features for classification between the genuine circuits and the HT circuits.

Assuming that SRTL contains N number of nets, the proposed features CC's mean (μCC) and the CC's standard deviation (σCC) of all SRTL nets in the netlist are defined as

$$\mu\mathrm{CC} = \frac{\sum_{\mathrm{i=1}}^{\mathrm{N}} \mathrm{CC_i}}{\mathrm{N}} \tag{9}$$

$$\sigma\mathrm{CC} = \sqrt{\frac{\sum_{\mathrm{i=1}}^{\mathrm{N}} (\mathrm{CC_i} - \mu\mathrm{CC})^2}{\mathrm{N}}} \tag{10}$$

In a similar way, the proposed features CO's mean (μCO) and the CO's standard deviation (σCO) of all SRTL nets in the netlist are defined as

$$\mu CO = \frac{\sum_{i=1}^{N} CO_i}{N} \tag{11}$$

$$\sigma CO = \sqrt{\frac{\sum_{i=1}^{N} (CO_i - \mu CO)^2}{N}} \tag{12}$$

3.2 Class Labelling

For the classifier training, each BS in the HT circuits is given a class label which is either a positive HT class or a negative genuine class. In this paper, the labelling of HT class depends on how the HT trigger signal is designed for the HT circuit. A HT trigger signal is defined as a maliciously added RTL variable which controls the HT execution.

There are three different scenarios we have considered. First, for HT circuits that do not have a HT trigger signal at RTL, we label the BS which directly executes the HT payload as a HT class, as highlighted below (BS #1 in the example in Figure 2).

```
if (cond == HT_cond)                    //BS #1 [HT class]
   out = HT_payload;
else                                    //BS #2 [Genuine class]
   out = genuine_function;
```

Fig. 2. Scenario 1: HT trigger signal does not exist at RTL

Second, for HT circuits that use a BS to control the HT trigger signal, we label the BS as a HT class (BS #1 in the example in Figure 3).

```
if (cond == HT_cond)                    //BS #1 [HT class]
   HT_trigger = 1;
else                                    //BS #2 [Genuine class]
   HT_trigger = 0;

if (cond == HT_trigger)                 //BS #3 [Genuine class]
   out = HT_payload;
else                                    //BS #4 [Genuine class]
   out = genuine_function;
```

Fig. 3. Scenario 2: HT trigger signal is controlled by a branching statement

Third, for HT circuits that control HT trigger signal without using a conditional statement, the BS which tests the HT trigger signal and executes the HT payload is labelled as HT class (BS #1 in the example in Figure 4).

```
HT_trigger = in1&in2&in3&in4

if (cond == HT_trigger)                 //BS #1 [HT class]
   out = HT_payload;
else                                    //BS #2 [Genuine class]
   out = genuine_function;
```

Fig. 4. HT payload (a) Without attribute III (b) With attribute III

3.3 Dataset Balancing

HT detection using machine learning approach suffers from an imbalanced dataset issue due to insufficient HT sample availability. This imbalanced dataset issue will cause biased training towards the major class. We balance the dataset using the adaptive synthetic sampling algorithm (ADASYN) [18]. ADASYN is an oversampling method that generates synthetic minor class data based on a weighted distribution which describes the learning difficulty of each of the minor class data. In this paper, the minor class is the HT class. The generated synthetic HT class data is combined with the original extracted dataset to form a balanced dataset which will be used as the training set. The ratio of HT class data and genuine class data is aimed to achieve approximately 1:1.

3.4 Training and Validation of Classifiers

Three supervised machine learning algorithm candidates are used to develop the HT classifiers. These machine learning algorithms are the popular binary classification algorithms including decision tree (DT), k-NN, and SVM. To obtain a fair classifier validation result, we implement k-fold cross-validation method. The flowchart of k-fold cross validation is illustrated in Figure 5. In this paper, we applied k=10 for the k-fold cross-validation. The best performing algorithm is selected for the proposed HT classification.

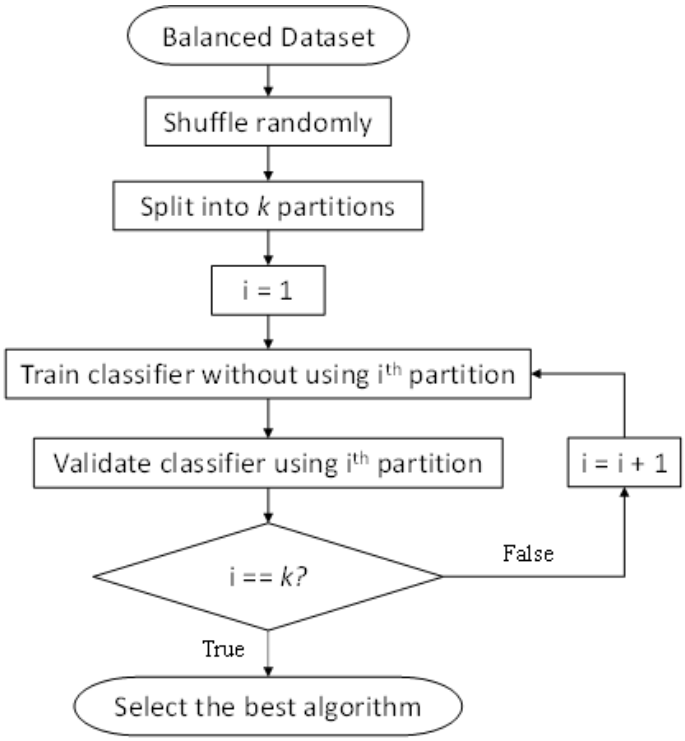

Fig. 5. Flow of k-fold cross validation

There are a few performance metrics commonly used for machine learning approaches. Positives refer to the HT predictions; negatives refer to the genuine predictions. True positive (TP) indicates the event where a HT class data is correctly predicted; false negative (FN) indicates the event where a HT class data is incorrectly predicted as genuine class. Similarly, true negative (TN) indicates the event where a genuine class data is correctly predicted; false negative (FP) indicates the event where a genuine class data is incorrectly predicted as HT class. TPR, or sensitivity, measures the proportion

of the correct prediction among the positives; TNR, or specificity, measures the proportion of the correct prediction among the negatives. TPR and TNR can also be explained as the estimated correct prediction rate among the positives and the negatives, respectively. Lastly, ACC is used to measure the overall percentage of correct prediction over all the samples. TPR, TNR, and ACC are defined as

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP+FN}} \tag{13}$$

$$\mathrm{TNR} = \frac{\mathrm{TN}}{\mathrm{TN+FP}} \tag{14}$$

$$\mathrm{ACC} = \frac{\mathrm{TP+TN}}{\mathrm{TP+TN+FP+FN}} \tag{15}$$

3.5 HT Detection Coverage Evaluation

We further evaluate the trained classifiers with another set of HT circuits to determine their HT detection coverage. The HT circuits are designed based on a HT modelling method which categorically models the HTs based on four HT attributes. The designed HT circuits should originate from an unseen circuit which is not included in the training data.

3.5.1 HT modelling

This paper targets HTs that are stealthy and tend to bypass conventional testing. Based on this assumption, we list out four HT binary attributes which can describe the stealthiness of the targeted HTs, as listed in Table 1. The HT attributes are then used to design some HT circuits for the evaluation of the HT detection coverage. Each of the designed circuits has its own unique combination of the HT attributes and attack effect.

Table 1
HT modelling attributes

Attribute#	Description
I	Does the HT have a deactivation mechanism?
II	Does the HT require multiple trigger patterns at multiple clock cycles to be activated?
III	Is the HT effect observable at only the timeframe at which circuit output is not a concern (idle state)?
IV	Is the HT trigger logic distributed throughout multiple blocks instead of only a single block?

For Attribute I, we consider that a HT with a deactivation mechanism is a stealthier HT which could have a higher chance of bypassing the testing. Assuming that the HT could be deactivated some time frames after executing the attack, the HT is harder to be detected even when an unusual activity is found during testing.

For Attribute II, we consider that the HT requiring multiple trigger patterns at multiple clock cycles to be activated is a stealthier HT since the condition is much rarer than using single trigger pattern.

For Attribute III, we consider that the HT whose effect is observable only at the time frame when the circuit output is not a concern such as idle state is a stealthier HT. Such HT can leak information without affecting the major functionality of the circuit, and thus is harder to be detected.

For Attribute IV, we consider that the HT whose trigger logic is distributed throughout multiple blocks instead of a single block is a stealthier HT. The distributed structure of the HT makes the tracing of HT signal becomes harder and confusing.

3.5.2 HT detection coverage

In this paper, we use another metric, HT detection coverage, to measure what types of HTs can or cannot be successfully detected by the classifier. We measure the classifier's HT detection coverage based on the HT attributes previously discussed. The analysis of HT detection coverage will be further discussed in detail with some examples in Chapter 4. In addition, the percentage of HT detection coverage is also included as another performance metric to quantify the HT detection coverage, which is defined as

$$\text{HT Coverage}\% = \frac{\text{number of HT circuits with at least one TP detection}}{\text{total number of HT circuits}} \times 100\% \quad (16)$$

4. Experimental Result and Discussion

For the experiment, we trained the classifiers using different feature vectors including RTL, GL, and the proposed double-abstraction-level feature vectors. Our aim was to show the effect of utilizing double-abstraction-level feature vector as compared to single-abstraction-level feature vector.

The circuits used in the experiment are listed in Table 2. For the classifier training, a total of 19 HT benchmark circuits were collected from Trust-Hub; for the evaluation of HT detection coverage, 24 HT circuits are designed based on a genuine Keccak encryption circuit.

Table 2
List of HT circuits used for the experiment

Classifier Training & Validation		HT Detection Coverage Evaluation	
AES-T400	RS232-T600	Genuine	TL04
AES-T700	RS232-T700	TC00	TL05
AES-T800	RS232-T900	TC01	TL06
AES-T900	RS232-T901	TC02	TL07
AES-T1000	wb_conmax-T200	TC03	TL08
AES-T1100	wb_conmax-T300	TC08	TL09
AES-T1200		TC09	TL10
AES-T1600		TC10	TL11
AES-T1700		TC11	TL12
RS232-T100		TL00	TL13
RS232-T300		TL01	TL14
RS232-T400		TL02	TL15
RS232-T500		TL03	

4.1 Validation Result of Trained Classifiers

Table 3 showed the result of classifier training using different classification algorithms and feature vectors. As depicted in the table, DT generally achieved the highest ACC as compared to the other two classification algorithms. Among the DT classifiers, the proposed classifier trained with the double-abstraction-level feature vector achieved the highest ACC of 99.97%. It proved that the double-abstraction-level feature vector provided a better learning process.

Table 3
Result of classifier validation after training

Classifier	Feature Vector	TP	FN	TN	FP	TPR (%)	TNR (%)	ACC (%)
DT	RTL	6253	10	5885	379	99.84	93.95	96.89
	GL	6231	18	6231	32	99.71	99.49	99.60
	Proposed (RTL+GL)	6259	1	6261	3	99.98	99.95	99.97
k-NN	RTL	5826	437	6261	3	93.02	99.95	96.49
	GL	5975	274	6229	35	95.62	99.44	97.53
	Proposed (RTL+GL)	6242	18	6259	5	99.71	99.92	99.82
SVM	RTL	6258	5	5169	1095	99.92	82.52	91.22
	GL	5510	739	5623	641	88.17	89.77	88.97
	Proposed (RTL+GL)	6260	0	6099	165	100.00	97.37	98.68

4.2 Evaluation Result of HT Detection Coverage

In the next evaluation using self-designed circuits, as shown in Table 4, the proposed classifier achieved the second highest ACC of 88.23% which was 8.91% lower than the GL classifier. However, the GL classifier had a very low sensitivity whereby its TPR had only 6.25%. Since sensitivity is highly focused on HT detection to avoid HT bypassing the detection, we consider that the proposed classifier has a better performance in which the highest TPR of 81.25% was achieved. We find that there is a large performance difference between the first validation and the second evaluation results, especially for the RTL and the GL classifiers. It is possibly due to the overfitting issue whereby the classifiers were biased towards the training data and not robust against the unseen circuits in the second evaluation. Therefore, we think that the second evaluation stage using circuits not included in training data is mandatory to identify such issue.

Table 4
Result of classifier evaluation using self-designed HT circuits

Classifier	Feature Vector	TP	FN	TN	FP	TPR (%)	TNR (%)	ACC (%)
DT	RTL	24	24	671	964	50.00	41.04	41.31
	GL	3	45	1632	3	6.25	99.82	97.15
	Proposed (RTL + GL)	39	9	1446	189	81.25	88.44	88.24

In Table 5, the HT detection coverage of each classifier is shown in detail. Based on the assessment measures (<IV, III, II, I>), we found that the proposed classifier is unable to detect HTs with attributes <1,X,1,0>. "X" means that the attribute is a "don't care" attribute and does not seem to affect the HT coverage. In other words, the proposed classifier cannot detect the HTs with all the following characteristics: i) it does not have a deactivation mechanism, ii) its HT activation requires multiple trigger patterns, and iii) its HT trigger has a distributed structure. Similarly, we found that the RTL classifier cannot detect <1,X,X,X>, while the GL classifier can only detect <1,X,0,1>.

The RTL classifier detected 12 out of 24 HT circuits. The GL classifier detected 3 out of 24 HT circuits. The RTL classifier and the GL classifier have different HT detection coverage. The proposed classifier achieved an improved HT detection coverage whereby 21 out of 24 HT circuits are successfully identified. The HT circuits that were detected by the single-abstraction-level classifiers were also detected by the proposed classifier. In addition, it successfully identified other 6 HT circuits which were not detected by any of the single-abstraction-level classifiers, including TC08, TC10, TL08, TL11, TL12, and TL 15. Furthermore, as with the GL classifier, the proposed classifier does not have any false positive detection in the genuine circuit, which makes its positive detection reliable. The

comparison proves that the proposed double-abstraction-level feature vector can improve the detection performance in term of HT coverage.

Table 5
HT detection coverage evaluation based on HT modelling attributes

Circuit	HT Effect	Attributes				Testing Result											
						RTL (DT)				GL (DT)				Proposed (DT)			
		IV	III	II	I	TP	FN	TN	FP	TP	FN	TN	FP	TP	FN	TN	FP
Genuine	-	-	-	-	-	0	0	23	28	0	0	51	0	0	0	51	0
TC00	Change functionality	0	0	0	0	1	0	23	33	0	1	56	0	1	0	55	1
TC01		0	0	0	1	1	0	23	34	0	1	57	0	1	0	54	3
TC02		0	0	1	0	3	0	23	33	0	3	56	0	3	0	55	1
TC03		0	0	1	1	3	0	25	32	0	3	57	0	3	0	54	3
TC08		1	0	0	0	0	1	28	36	0	1	64	0	1	0	55	9
TC09		1	0	0	1	0	1	28	37	1	0	64	1	1	0	54	11
TC10		1	0	1	0	0	3	31	55	0	3	86	0	0	3	70	16
TC11		1	0	1	1	0	3	31	56	0	3	87	0	3	0	68	19
TL00	Leak information	0	0	0	0	1	0	23	33	0	1	56	0	1	0	55	1
TL01		0	0	0	1	1	0	23	34	0	1	57	0	1	0	54	3
TL02		0	0	1	0	3	0	25	31	0	3	56	0	3	0	55	1
TL03		0	0	1	1	3	0	25	32	0	3	57	0	3	0	54	3
TL04		0	1	0	0	1	0	25	31	0	1	56	0	1	0	55	1
TL05		0	1	0	1	1	0	25	32	0	1	57	0	1	0	54	3
TL06		0	1	1	0	3	0	25	31	0	3	56	0	3	0	55	1
TL07		0	1	1	1	3	0	25	32	0	3	57	0	3	0	54	3
TL08		1	0	0	0	0	1	28	36	0	1	64	0	1	0	55	9
TL09		1	0	0	1	0	1	28	37	1	0	64	1	1	0	54	11
TL10		1	0	1	0	0	3	31	55	0	3	86	0	0	3	70	16
TL11		1	0	1	1	0	3	31	56	0	3	87	0	3	0	68	19
TL12		1	1	0	0	0	1	30	34	0	1	64	0	1	0	55	9
TL13		1	1	0	1	0	1	30	35	1	0	64	1	1	0	54	11
TL14		1	1	1	0	0	3	31	55	0	3	86	0	0	3	70	16
TL15		1	1	1	1	0	3	31	56	0	3	87	0	3	0	68	19

4.3 Comparison to Previous Studies

We find that the reviewed studies did not evaluate their trained classifiers using truly unseen circuits. In other words, the classifiers were trained and evaluated based on different HT circuits, but which originated from the same genuine circuit. For example, Han *et al.,* [13] trained and validated the classifier using 21 same AES circuits with only the inserted HTs are different. Since HTs usually occupy a tiny portion of the circuit, the genuine signals are mostly identical in every circuit. If there is an overfitting issue, it cannot be identified.

Since no reviewed studies was evaluated based on truly unseen circuits (as we did in the second evaluation), we compare their results with our validation result, as summarized in Table 6. Our validation result is relatively high as compared to most of the reviewed studies [3-8,11-12,14].

Table 6
Comparison between our proposed solution to previous studies

Method	Classifier	Level	Validation (%)		
			TPR	TNR	ACC
Hasegawa *et al.,* [3]	SVM	GL	83.00	49.00	-
Inoue *et al.,* [4]	SVM	GL	100.00	<70.00	-
Hasegawa *et al.,* [5]	Random forest	GL	-	98.20	-
Hasegawa *et al.,* [6]	Neural network	GL	85.00	70.00	-
Inoue *et al.,* [7]	Neural network	GL	72.90	90.00	-
Kurihara and Togawa [8]	Random forest	GL	63.60	100.00	
Salmani [9]	*k*-means clustering	GL	100.00	100.00	100.00
Xie *et al.,* [10]	SVM	GL	100.00	100.00	100.00
Kok *et al.,* [11]	*k*-NN	GL	99.85	99.95	99.90
Lu *et al.,* [12]	DBSCAN	GL	79.00	99.00	
Han *et al.,* [13]	Gradient boosting	RTL	100.00	89.32	-
Yasaei *et al.,* [14]	GNN	RTL	-	97.00	
Our proposed solution	DT	RTL	99.98	99.95	99.97

There are a few studies which achieve a higher performance with their proposed method. In Salmani [9], the proposed unsupervised k-means clustering method achieved 100% ACC. However, the proposed method has a limitation that it is not suitable to be used for very large circuits. Besides, it relies on a threshold which is set manually. On the contrary, our proposed method seems not to be affected by the circuit size and does not require any manual analysis after the classifier is developed. In Xie *et al.,* [10], an SVM classifier achieved ACC of 100%. Unlike our method which can pinpoint the HT branching statements, their proposed method classifies HT GL netlist, but the HT's location is unknown. This makes further analysis of the HT difficult when it is required.

5. Conclusion

We proposed a HT branching statement classification method based on a double-abstraction-level feature vector. As per our knowledge, we are the first to propose to use features extracted at different abstraction levels (RTL and GL) to improve HT detection coverage. Based on the experimental result, the proposed double-abstraction-level classifier achieved 81.25% TPR, 88.44% TNR, and 88.24% ACC in evaluation utilizing unseen circuits. We proved that the proposed classifier outperforms the single-abstraction-level classifiers with 87.5% HT detection coverage, whereby 21 out of the 24 HT circuits were successfully identified and no false positive was found in the genuine circuit.

The proposed method suggested using features of different levels to increase the HT detection coverage. To further prove the HT detection effectiveness of the proposed method, it is necessary to involve more than two levels. Besides, we presume that the proposed method is not only effective to HT branching statements or HT nets. To prove this, another research on the effectiveness of the proposed method on other HTs is required.

Acknowledgement
This research was not funded by any grant.

References
[1] Tehranipoor, Mohammad, and Farinaz Koushanfar. "A survey of hardware trojan taxonomy and detection." *IEEE Design & Test of Computers* 01 (2016): 1-1. https://doi.org/10.1109/MDT.2009.159

[2] Salmani, Hassan, Mohammad Tehranipoor, and Ramesh Karri. "On design vulnerability analysis and trust benchmarks development." In *2013 IEEE 31st international conference on computer design (ICCD)*, pp. 471-474. IEEE, 2013. https://doi.org/10.1109/ICCD.2013.6657085

[3] Hasegawa, Kento, Masaru Oya, Masao Yanagisawa, and Nozomu 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)*, pp. 203-206. IEEE, 2016. https://doi.org/10.1109/IOLTS.2016.7604700

[4] Inoue, Tomotaka, Kento Hasegawa, Masao Yanagisawa, and Nozomu Togawa. "Designing hardware trojans and their detection based on a SVM-based approach." In *2017 IEEE 12th international conference on ASIC (ASICON)*, pp. 811-814. IEEE, 2017. https://doi.org/10.1109/ASICON.2017.8252600

[5] Hasegawa, Kento, Masao Yanagisawa, and Nozomu 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)*, pp. 1-4. IEEE, 2017. https://doi.org/10.1109/ISCAS.2017.8050827

[6] Hasegawa, Kento, Masao Yanagisawa, and Nozomu 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)*, pp. 227-232. IEEE, 2017. https://doi.org/10.1109/IOLTS.2017.8046227

[7] Inoue, Tomotaka, Kento Hasegawa, Yuki Kobayashi, Masao Yanagisawa, and Nozomu Togawa. "Designing subspecies of hardware Trojans and their detection using neural network approach." In *2018 IEEE 8th International Conference on Consumer Electronics-Berlin (ICCE-Berlin)*, pp. 1-4. IEEE, 2018. https://doi.org/10.1109/ICCE-Berlin.2018.8576247

[8] Kurihara, Tatsuki, and Nozomu Togawa. "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)*, pp. 1-4. IEEE, 2021. https://doi.org/10.1109/IOLTS52814.2021.9486700

[9] Salmani, Hassan. "COTD: Reference-free hardware trojan detection and recovery based on controllability and observability in gate-level netlist." *IEEE Transactions on Information Forensics and Security* 12, no. 2 (2016): 338-350. https://doi.org/10.1109/TIFS.2016.2613842

[10] Xie, Xin, Yangyang Sun, Hongda Chen, and Yong Ding. "Hardware Trojans classification based on controllability and observability in gate-level netlist." *IEICE Electronics Express* 14, no. 18 (2017): 20170682-20170682. https://doi.org/10.1587/elex.14.20170682

[11] Kok, Chee Hoo, Chia Yee Ooi, Michiko Inoue, Mehrdad Moghbel, Sreedharan Baskara Dass, Hau Sim Choo, Nordinah Ismail, and Fawnizu Azmadi Hussin. "Net classification based on testability and netlist structural features for hardware Trojan detection." In *2019 IEEE 28th Asian Test Symposium (ATS)*, pp. 105-1055. IEEE, 2019. https://doi.org/10.1109/ATS47505.2019.00020

[12] Lu, Renjie, Haihua Shen, Zhihua Feng, Huawei Li, Wei Zhao, and Xiaowei Li. "HTDet: A clustering method using information entropy for hardware Trojan detection." *Tsinghua Science and Technology* 26, no. 1 (2020): 48-61. https://doi.org/10.26599/TST.2019.9010047

[13] Han, Tao, Yuze Wang, and Peng Liu. "Hardware trojans detection at register transfer level based on machine learning." In *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1-5. IEEE, 2019. https://doi.org/10.1109/ISCAS.2019.8702479

[14] Yasaei, Rozhin, Shih-Yuan Yu, and Mohammad Abdullah Al Faruque. "Gnn4tj: Graph neural networks for hardware trojan detection at register transfer level." In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1504-1509. IEEE, 2021. https://doi.org/10.23919/DATE51398.2021.9474174

[15] Choo, Hau Sim, Chia Yee Ooi, Michiko Inoue, Nordinah Ismail, Mehrdad Moghbel, and Chee Hoo Kok. "Register-transfer-level features for machine-learning-based hardware trojan detection." *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences* 103, no. 2 (2020): 502-509. https://doi.org/10.1587/transfun.2019EAP1044

[16] Goldstein, Lawrence H., and Evelyn L. Thigpen. "SCOAP: Sandia controllability/observability analysis program." In *Proceedings of the 17th Design Automation Conference*, pp. 190-196. 1980. https://doi.org/10.1145/800139.804528

[17] Bhunia, Swarup, Michael S. Hsiao, Mainak Banga, and Seetharam Narasimhan. "Hardware Trojan attacks: Threat analysis and countermeasures." *Proceedings of the IEEE* 102, no. 8 (2014): 1229-1247. https://doi.org/10.1109/JPROC.2014.2334493

[18] He, Haibo, Yang Bai, Edwardo A. Garcia, and Shutao Li. "ADASYN: Adaptive synthetic sampling approach for imbalanced learning." In *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*, pp. 1322-1328. IEEE, 2008. https://doi.org/10.1109/IJCNN.2008.4633969