Journal of Electronic Testing (2023) 39:465–485
https://doi.org/10.1007/s10836-023-06080-9

Structural and SCOAP Features Based Approach for Hardware Trojan Detection Using SHAP and Light Gradient Boosting Model

Richa Sharma[1] · **G. K. Sharma**[1] · **Manisha Pattanaik**[1] · **V. S. S. Prashant**[1]

Received: 23 March 2023 / Accepted: 16 August 2023 / Published online: 22 September 2023

Abstract
Hardware Trojan (HT) is the most critical threat due to outsourcing of Integrated circuit designing phases. Existing machine learning-based HT detection techniques at the pre-silicon IC phase use the structural or SCOAP gate-level netlist features for detection. However, these techniques either fail to detect the always-on-Trojans or low SCOAP Trojans, thus provides large false positives/negatives. Moreover, they fail to interpret the model prediction locally due to model-specific feature importances, identify the best feature subset in large retraining rounds, and also drop some relevant features. Therefore, to tackle these limitations, this paper proposes a new technique that utilizes structural and SCOAP features to detect HT from the gate-level netlist. The proposed technique utilizes the fastest model Light Gradient Boosting that uses gradient-based one-side sampling and exclusive feature bundling to reduce the computational time. Further, a model agnostic Shapley additive explanations (SHAP) is employed to identify each feature global and local impact on model prediction, thus making the prediction transparent. Moreover, a quartile-based feature selection method is proposed, which uses SHAP to identify the optimal feature set by keeping low retraining rounds. Experimental results show that the proposed technique accurately detects always-on-Trojans and HT nets from Trust-Hub, DeTrust, DeTest and MIMIC based Trojan benchmarks.

Keywords Hardware trojan · Structural & SCOAP features · Machine learning · SHAP · Light gradient boosting

1 Introduction

Nowadays, in order to lessen the cost and expedite the process, semiconductor firms mostly outsourced Integrated circuit (IC) supply chain phases to third party houses, which makes them vulnerable to malicious Hardware Trojan (HT) intrusion [5]. These external houses got ample chances to insert HT in these IC phases, e.g., they either manipulate the lithographic IC masks or modify the functioning of the gate-level netlist by deliberately inserting the additional gates in it [43]. An adversary designed the stealthiest Trojan, which is triggered by the rarest input combinations to perform malicious functions such as denial of service (DoS), sensitive information leakage, tampering the circuit functionality, etc. [3]. The presence of HT's in the IC's proves to be harmful and life-threatening for various sensitive and real-life applications such as self-driving vehicles [7], surveillance systems [47], IoT [50], defense and cyber security systems [48]. Therefore, to combat this attack, several countermeasures are proposed by the researchers at both pre-silicon and post-silicon IC phases [27, 44]. However, this paper mainly focuses on the untrusted pre-silicon IC designing phase, where an adversary alters the original gate-level netlist by inserting the HT in it. Conventional verification approaches such as UCI [17], FANCI [45] etc. flag the suspicious circuits by performing the formal verification or functional analysis on the circuits. However, these approaches suffer from low extensibility, require large computational effort, fail to detect implicit HTs, threshold dependent thus provide large false positives/negatives because some Trojans smartly bypass these verification/functional analysis [27, 40, 53].

Responsible Editor: U. Guin

✉ Richa Sharma
richa@iiitm.ac.in

G. K. Sharma
gksharma@iiitm.ac.in

Manisha Pattanaik
manishapattanaik@iiitm.ac.in

V. S. S. Prashant
vssprashant@gmail.com

[1] ABV-Indian Institute of Information Technology and Management, Gwalior 474015, India

Recently, gate level feature analysis-based approaches are also reported, which tend to be faster and more accurate than verification approaches [18, 22]. They mainly extracts the structural/SCOAP features from gate-level netlist and develop threshold or machine learning (ML) based approach to identify the Trojan nets [15, 24, 32]. Hasegawa et al. [16] extracts the structural features from gate-level netlist and selects the best feature set to detect HT nets using a Random forest classifier. Similarly, Salmani [35] extracts the SCOAP values and applies k-means clustering to perform detection. However, the use of only structural/SCOAP features could not identify either the always-on-Trojans or low SCOAP value Trojans. Therefore, Kok et al. [24] combines structural and SCOAP features for detection and train decision tree on feature subset chosen by Maximum relevance-Minimum redundancy method. However, the use of model-specific feature importance by [16] gives inconsistent ranking, whereas the relevancy method used in [24] is affected by the outliers and sensitive to noise. Further, they only provide the importance at the global level and fail to identify the local feature contribution and feature interaction during prediction [31]. Besides, to identify the best feature subset, they follow the partial/full iterative process by either dividing the feature set into halves repetitively or including every feature iteratively to compute performance which is time-consuming and may drop out some relevant features. Moreover, the use of class weighting/oversampling for class imbalance either makes the model biased towards minority class or causes overfitting. Besides, ML models used by them are nonresistant to noise, complex, and time-consuming, which affects the model training and generalization procedure.

Our previously developed approach [39] utilized SCOAP features and the XGBoost model for HT detection. To address class imbalance, we proposed a class weighting scheme which gives higher weights to minority Trojan inserted class. Additionally, we employed permutation feature importance to prioritize the features and introduced a new mean feature selection method to identify the optimal feature set. One of the main limitations is its reliance on a limited feature set, which may result in the inability to detect certain types of HTs, including the newly developed Trojans such as DeTest. These Trojans can reduce the SCOAP feature values of Trojan nets to 10%, making them more challenging to detect. Furthermore, the previous approach lacks model explainability, as it only provides global feature importance without detailed insights into individual features' contributions. This limitation hampers our ability to understand the underlying factors influencing HT detection and may hinder the interpretability of the results. Additionally, the use of mean in the feature selection method introduces sensitivity to noise, which can lead to the exclusion of relevant features and potentially impact the overall performance of our approach. Moreover, the class weighting scheme we proposed to address class imbalance may sometimes become biased towards the Trojan class, resulting in an increased False Negative Rate in certain cases. This imbalance can affect the overall performance and accuracy of our approach.

Therefore, to overcome the above limitations, a new HT detection technique is proposed in this paper, which utilizes both structural and SCOAP features to identify the HT nets from gate-level netlist. The proposed technique employs a fast and efficient Light Gradient Boosting (LGB) model which uses gradient-based one side sampling that focuses on the undertrained samples by selecting the large gradient samples from the dataset and uses exclusive feature bundling, which combines mutually exclusive features into single bundle thus reduces the memory and time consumption. Moreover, it grows the tree leaf-wise by choosing the leaf with a maximum loss instead of level-wise, thus building the tree faster and can handle the missing values and overfitting by providing inbuilt support to ensure correct training. Further, model agnostic Shapley additive explanations (SHAP) is utilized that provide both global and local feature contribution for every sample and capture the interaction between features that helps to analyze which feature influences the LGB model predictions most. Besides, a quartile-based feature selection method is proposed, which identifies the optimal feature set using SHAP feature importance by minimizing the model retraining rounds. Further, to avoid the limitations that arise in oversampling/weighting, a combined sampling SMOTE-Tomek Links is used, which apply oversampling and then undersampling to balance classes. Finally, the proposed technique is scalable because new features can be added easily whenever new stealthy Trojans are designed. The major contributions are given as follows:

1. A new LGB model-based HT detection technique is proposed, which utilizes structural and SCOAP gate-level netlist features during training for fast and accurate detection.
2. Thorough experimental analysis is carried out using model agnostic SHAP to interpret which features interacted and influenced the LGB model prediction most at the global and local level.
3. A Quartile-based feature selection method is proposed that selects the best subset of features using SHAP feature importance in minimum model retraining rounds.
4. Experimental evaluation on Trust-Hub, DeTrust, DeTest benchmarks, always-on Trojans and MIMIC based Trojan benchmarks provides on-an-average approximately 99% accuracy, nearly 0% false positive and 2% false negative rates, respectively.

The rest of the paper is organized as follows: Section 2 gives a literature review analysis of existing gate-level netlist-based HT detection techniques. Section 3 discussed HT features, attack model and problem statement. Section 4 explains the proposed HT detection technique, which includes LGB model learning followed by a new feature selection method and HT detection algorithm. Experimental results and comparative analysis is presented in Section 5. Finally, Section 6 concludes the paper.

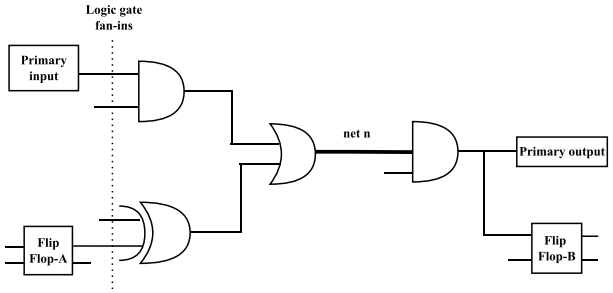
2 Literature Review: Analysis

Several HT detection techniques have been explored by the researchers at different IC phases, and a detailed explanation is given in [22, 28]. However, this section contains the analysis of the existing gate-level netlist feature analysis-based HT detection techniques. Oya et al. [32] proposed a score-based technique, in which nine feature templates are extracted, i.e., weak nets from the gate-level netlist, and computed the score based on HT structure present in each feature. Further, a score threshold is chosen to identify the netlist as Trojan free (TF)/ Trojan inserted (TI). However, it fails to detect smartly crafted Trojans that do not resemble any weak net structure. Moreover, setting the correct threshold is a cumbersome task that leads to large false positives (FP)/ false negatives (FN). Hasegawa et al. [15] extracted the five gate-level netlist structural features, and class weighting is used that assign higher weights to the minor TI class to identify the Trojan nets using a support vector machine (SVM). However, they fail to accurately detect Trojan in several benchmarks due to a lack of sufficient features. Besides, the weighting scheme might get biased towards the TI class, thus generating larger FP's.

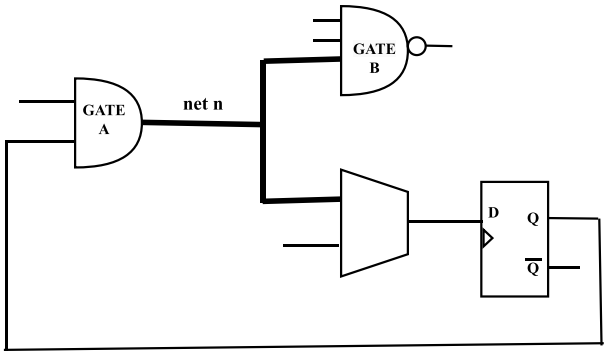
Further, 51 features are extracted by them in [16] and feature importance is computed by Random Forest Classifier (RFC). Besides, the features are divided into halves iteratively until the best average F-measure is achieved, and when the best set is obtained, a single feature is dropped from that set to compute the average (avg) F-measure again, this process keeps on repeating for each feature present in the set. Then the whole single feature drop process is repeated on the new set until no further improvement is seen and RFC is trained on the best set for detection. However, model-specific feature importance provides a ranking based on node splitting criterion, which is inconsistent and could not identify the feature contribution locally, thus fail to interpret the model predictions [31]. Besides, feature selection is time-consuming, and a random division of feature sets may drop the relevant features. Further, oversampling replicates the TI samples multiple times, which tends the model to overfit, thus fail to identify HT's in many benchmarks. Furthermore, the same previous best feature set is utilized by them in [14], and neural network (NN) is trained for detection. Since this technique uses the same features, thus provides small true positives (TP)/ true negatives (TN) in many benchmarks. Moreover, choosing the best set of neurons in NN is time-consuming and complex. Wang et al. [46] extracted the trigger net features and trained two XGBoost models to detect combinational and sequential Trojans. However, training two detection models is a tedious task, and it achieves low detection accuracy and on-an-avg low true TP/TN. Besides both [14] and [46] uses the similar weighting scheme as used in [15], which also affects the detection results. Similarly, Kurihara and Togawa [25] extracted 25 structural trigger features, combined them with 11 features, and trained the RFC model to detect HT but provides lower recall in several benchmarks.

In contrast, Salmani [35] proposed COTD in which SCOAP features from the circuits are extracted and then k-means clustering is applied to identify the HT nets. However, this technique missed out on low SCOAP HT nets or may predict high SCOAP TF nets as TI. Further, k-means clustering is highly affected by noise and possess less generalization capability, which affect the detection rate. Similarly, Xie at al. [49] also applies k-means clustering and extracts the intercluster distance (one feature) from SCOAP values and combines it with three other circuit primitive features to perform the detection using SVM. However, they randomly assign higher/lower weights to each feature and fail to locate the HT specifically because the detection is performed at the circuit level instead of the net level. Further, oversampling makes the model prone to overfit, and it incurs a large overhead due to the use of multiple classifiers. Kok et al. [24], utilizes the SCOAP values of combinational and sequential circuits as features, and for class balancing, ADASYN is used which generates the synthetic data for TI class to perform detection using Bagged trees. However, some of the generated artificial data get similar to the TF class, which affects the detection rate and F-measure in several benchmarks.

Our earlier work [39], use the same SCOAP features as [24] and a new weighting scheme are used in class weighted XG Boost classifier, which is trained on optimal feature set selected by new feature selection method to detect HT nets. Tebyanian et al. [42] proposed SC-COTD in which they also use combinational and sequential SCOAP features and apply k-means clustering with different cluster values. Finally, the clustering results are fed to the decision-making system, which identifies the circuits as Trojan-free/inserted based on the majority voting scheme. Salmani [36] improved the COTD by performing a multiple rounds of clustering and post-processing technique is applied to reduce false positives. Gaikwad et al. [12] propose a trust verification framework using supervised learning to detect HTs.



(a) Logic gate fan-in, Flip-Flop, Primary input & Primary output features of target net n in a gate-level netlist.



(b) Loop feature of target net n in a gate-level netlist.

Fig. 1 Structural features of gate-level netlist [15, 16]

Recently, DeTest [52] defeated SCOAP based techniques by designing new Trojans, which decreases the SCOAP values of HT nets up to 10%. Moreover, structural/SCOAP features-based techniques fail to detect either always-on-Trojans or low SCOAP HT nets. Kok et al. [23], combines structural and SCOAP features, and the Maximum relevance-Minimum redundancy method (MRMR) method is applied for feature ranking. Further, features are included iteratively, the highest accuracy feature subset is chosen, and the Decision Tree model is trained for detection. However, MRMR is sensitive to outliers, and noise [20] which affect the ranking process, and iterative feature selection is also time-consuming. Moreover, data generated by ADASYN also affects the detection accuracy.

3 Background

3.1 Structural & SCOAP HT Features

An adversary can implant different types of HT in the IC designing phase with varying triggering mechanisms and payload [3]. The functional Trojans possess trigger circuitry, which, once activated by certain input conditions, executed the payload and performed the desired malicious functions [5]. Besides, there are also exist always-on-Trojans which does not require any trigger to activate and continuously execute the malicious functions all time [48]. However, these HT's exhibit some common traits through which they can be detected, e.g., they might be connected to the low switching activity nets, which are rarely triggered or contain larger fan-ins [51]. Therefore, we have extracted several structural and SCOAP features of combinational and sequential circuits to detect HT nets. Structural features mainly give insight about the connection that exists among several logic gates, thus providing information about the HT's present in the circuits. We have extracted the common structural features suggested by existing techniques [16, 46] as shown in Table 1. It has been analyzed that mostly combinational trigger circuits possess large logic gate fan-ins, thus feature fan_in_x represents the number of logic gate fan-ins, x level away from net n, where x ranges from ($1 \leq x \leq 5$). For example, Table 2 shows the average fan-ins feature values of Trojan and genuine nets of Trust-Hub benchmark RS232-T1400, which shows that the Trojan nets possess higher fan-in values than genuine nets.

Similarly, for sequential triggered nets, the level of flip flops is small because they are placed closely. Thus, features in_flipflop_x/out_flipflop_x and in_nearest_flipFlop/out_nearest_flipFlop indicates the number of flip-flops, x

Table 1 Structural features extracted from gate-level net-list

Structural Trojan Features	Explanation($1 <= x <= 5$)
fan_in_x	Total logic-gate fan-ins, x-logic level away from net n
in_flipFlop_x & out_flipFlop_x	Total number of flip-flops, x- logic level away from the input/output side of the net n
in_nearest_flipFlop & out_nearest_flipFlop	Minimum logic level to flip flops from input/output side of net n
in_loop_x & out_loop_x	Total number of x-logic level loops, for the input/output side of net n
in_nearest_pin & out_nearest_pin	Minimum logic level to the primary input/output from net n

Table 2 Average fan-ins Values for RS232-T1400 circuit

Net Type	fan_in_1	fan_in_2	fan_in_3	fan_in_4	fan_in_5
Genuine net	1.735	2.743	3.823	5.217	6.398
Trojan net	3.53	8.461	13.769	19.46	23.692

logic level away, and minimum level to flip-flops from input and output side of net n. Further, Trojan nets may frequently found close to primary inputs or primary output because earlier is often used as a Trojan trigger and later used to propagate malicious output. However, HTs can potentially be located anywhere within in the circuit, including leveraging scan-chains for insertion. But, in this paper we aim to detect the Trojans resides close to primary input or primary output, therefore, we extract the minimum level to primary input/output (in_nearest_pin & out_nearest_pin) features from net n. Besides, sequential HT often contains loops thus, the feature in_loop_x/out_loop_x tells us the number of x-level loops from the input and output side of net n. For example, Fig. 1(a) shows the extracted feature values for target net n up to level 2, it can be seen that the extracted fan-in feature value (fan_in_2 = 4). The total number of flip-flops from the input and output side of target net n, i.e., features in_flipFlop_2 & out_flipFlop_2 are 1. Moreover, the nearest input from flip-flop A to target net n (in_nearest_flipFlop = 2) and the nearest output from flip-flop B (out_nearest_flipFlop = 1). Further, the level of nearest primary input (in_nearest_pin) is two, and primary output (out_nearest_pin) is one. Similarly, Fig. 1(b) shows the example of a three-level loop feature in a circuit, it can be seen that Gate A is directly connected from net n, and there also exists an input loop (in_loop_3 = 3), i.e., the Gate A is three-level way from the net n.

However, the extracted 29 structural features alone fail to detect all the Trojans completely and also could not identify the always-on-Trojans. Therefore, we also incorporate SCOAP feature values because HT is mostly inserted at low switching activity area, thus possess large controllability or observability values [35]. We have extracted the SCOAP values of both combinational and sequential circuits, i.e., combinational & sequential-0 controllability ($CC0$, $SC0$), combinational & sequential-1 controllability ($CC1$, $SC1$), combinational & sequential observability (CO, SO) using the SCOAP method [13] to perform detection. Further, Table 3 shows the average SCOAP feature values of Trojan and genuine nets of Trust-Hub benchmark RS232-T1400, which indicates that Trojan nets possess higher controllability or observability values than genuine nets.

Table 3 Average SCOAP Values for RS232-T1400 circuit

Net Type	(CC0, CC1)	CO	(SC0, SC1)	SO
Genuine net	(13.63, 39.67)	108.27	(1.202, 3.657)	9.488
Trojan net	(416, 10.84)	1267.15	(41.15, 1)	125.92

However, sometimes SCOAP values of Trojan nets also become low, e.g., Trust-Hub benchmarks s35932-T200 and s38584-T100 possess low SCOAP values than genuine nets. Moreover, Trojans designed by DeTest [52] also reduces the SCOAP values, which clearly shows that SCOAP features alone are also not sufficient to detect Trojans precisely. Therefore, we have combined both extracted 29 structural and 6 SCOAP features for accurate Trojan detection.

3.2 LGB Model Based Learning

LGB model is an efficient and fastest ensemble model [21] which uses effective gradient-based one-side sampling (GOSS) and exclusive feature bundling (EFB) to reduce computational time, thus provides improved learning and accurate detection. Suppose there are N input training samples, LGB sequentially trains several base learners, i.e., T decision trees (DT) in an additive manner using gradient descent to minimize the loss (L), i.e., the gradient of L with respect to (w.r.t) base learner model is decreasing. The current base learner (P_T) is trained on the pseudo-residuals obtained by the previous base learner ($P_{(T-1)}$) and the final output is updated as given below:

$$P_T(x_N) = P_{(T-1)}(x_N) + \alpha h_T(x_N) \tag{1}$$

where α is a learning rate that indicates gradient step size, x_N is the current input sample, and $h_T(x_N)$ is a predictor function of $P_T(x_N)$ trained on the previous model residual. The above equation shows that final prediction of $P_T(x_N)$ is the summation of previous model prediction $P_{(T-1)}(x_N)$ and current $h_T(x_N)$ prediction. For each P_T, the classic GB model [11] enumerates all presorted feature values to compute the information gain of all possible splits in T, which is inefficient in terms of time and memory. In comparison, XGboost [6] use histogram-based splitting, which buckets the feature values of samples into several bins and performs splitting using these bins. However, it has been pointed out by [21] that gradients denote the degree of error, i.e., samples that possess small gradients are trained well and have low training error.

Therefore, to reduce the complexity and fasten the process, LGB uses optimized histogram-based splitting by applying the concept of GOSS which reduces the search space by down-sampling the small gradients samples while retaining the untrained large gradients samples which contribute more during tree building. However, to maintain the data distribution, it randomly chooses the small gradient samples from the training dataset and follows the leaf-wise tree growth approach instead of levelwise. Moreover, EFB bundles the mutually exclusive features, i.e., sparse features, into a single bundle to reduce the feature

dimensionality and to speed up the training. During each iteration of P_T, the gradients of samples are computed, then instead of iterating through all the samples, GOSS sort them based on their absolute gradients values and selects top $l\%$ large gradient samples to make subset C and randomly choose $s\%$ smaller gradient samples from the remaining data to make subset D. Now, the variance gains $Vag_m(o)$ of splitting m^{th} feature at point o for any fixed tree node is computed on samples present in $C \cup D$ given as:

$$Vag_m(o) = \frac{1}{N}\Bigg(\frac{(\sum_{x_i \epsilon C_l} g_i + \frac{1-a}{b}\sum_{x_i \epsilon D_l} g_i)^2}{N_l^m(o)} + \frac{(\sum_{x_i \epsilon C_r} g_i + \frac{1-a}{b}\sum_{x_i \epsilon D_r} g_i)^2}{N_r^m(o)}\Bigg) \tag{2}$$

where g_i are the gradients, $C_l = \{x_i \epsilon C : x_{im} \leq o\}$, $C_r = \{x_i \epsilon C : x_{im} > o\}$, $D_l = \{x_i \epsilon D : x_{im} \leq o\}$, $D_r = \{x_i \epsilon D : x_{im} > o\}$ are the left and right child of the node respectively. Coefficient $\frac{(1-a)}{b}$ is a constant multiplier that is used to normalize the sum of small gradients to balance data distribution. In this way, the LGB efficiently reduces the time and memory consumption by effectively removing the small gradients samples using GOSS and reduces the feature dimensions by combining the mutually exclusive features into a single bundle using EFB.

4 Light Gradient Boosting Model Based HT Detection Technique

This section explains the new HT detection technique which detects the HT nets from IC gate-level netlist followed by a proposed HT detection algorithm.

4.1 Attack Model and Problem Statement

To reduce the time and cost, semiconductor firms incorporate third-party IC's/ external vendors in the pre-silicon IC designing phase. An untrusted vendor has full access to the IC gate-level netlist and may intentionally insert the HT by adding the malicious gates. Our work in this paper mainly focuses on determining the combinational and sequential Trojans inserted by the untrusted vendors in the gate-level netlist during IC pre-silicon designing phase. We have formulated this HT detection problem as the classification problem where our main aim is to classify the nets as TI/TF in unknown gatelevel netlist using the LGB model. The problem is stated as follows, suppose during training, we have $'m'$ features of known gate-level netlist (training dataset), consisting of $'p'$ TI nets and $'q'$ TF nets with labels. The proposed technique trained the LGB model on optimal feature set obtained by proposed feature selection method. Finally, during testing, the feature set of unknown gate-level netlist is provided to the trained model which predicted the correct classes of nets, i.e., TF/TI present in the gate-level netlist.

4.2 Proposed HT Detection Technique

The proposed HT detection technique is shown in Fig. 2 which utilizes the LGB model and SHAP to detect the HT inserted by the attackers in the gate-level netlist during IC designing. Initially, the SCOAP and structural features of circuits are extracted and stored with labels (More details are provided in Section 5.1). Further, a combined sampling SMOTE-Tomek Links [2] is used to tackle class imbalance, which first oversamples the minority TI class using SMOTE, and then Tomek-links undersample the unnecessary/noisy data, thus providing a more balanced training dataset. Afterward, the LGB model is trained and SHAP feature importance is computed, then the optimal feature set is identified using a new feature selection method until the best performance is obtained. Finally, the LGB model is retrained on the best optimal feature set and provides predictions for unseen test data. The overall technique is presented in the following subsections.

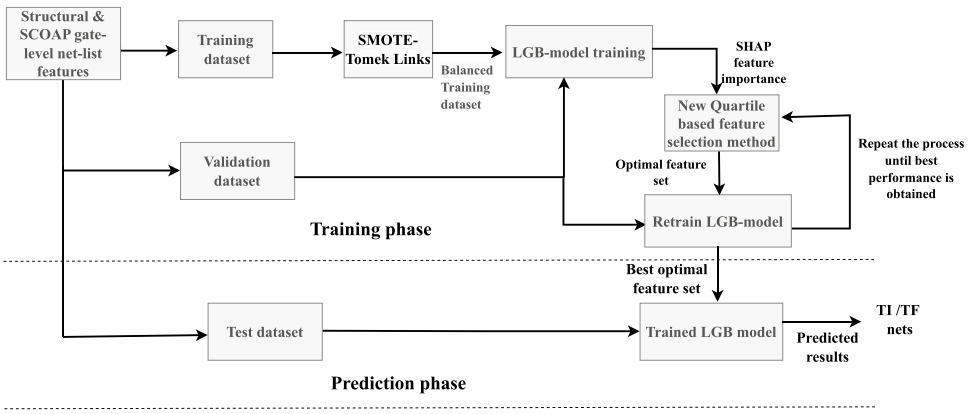


Fig. 2 Proposed HT detection technique

4.2.1 LGB Model for HT Detection

The accurate identification of Trojan nets relies on two factors: the selection of the correct ML model and how its hyperparameters are tuned during training according to the HT detection problem. Therefore, an efficient LGB model is utilized which focused mainly on large gradient samples during training and combines the mutually exclusive features to improve and fasten the learning process. The leaf-wise approach grows the asymmetric tree vertically by choosing the leaf node with maximum loss (large gradients) while leaving the other nodes at the same level, thus reducing more loss and converging faster. The leafwise tree is shown in Fig. 3, it can be seen that the root node splits feature out_nearest_pin with threshold 4.5 and after that, instead of splitting every node at that level, it only splits left child feature node CC0 while leaving the right child node as a leaf. Further, it again splits the left child feature CO at the next level and grows the tree only one side asymmetrically instead of levelwise. Whereas the positive/negative value in the leaf node shows its prediction values which shows that the tree is trained on the previous tree residuals.

We have applied the grid search to identify the correct values of LGB model parameters. Grid search is a systematic approach that evaluates different combinations of hyperparameter values to determine the set of values that leads to the highest performance according to a chosen evaluation metric. To conduct the grid search, we defined a range of potential values for each LGB model hyperparameter. These hyperparameters, such as learning rate, maximum depth, and number of estimators, play a crucial role in the performance of the model. The grid search algorithm then exhaustively evaluated all possible combinations of these parameter values. The values that produced the best performance are then selected as the final set of hyperparameters for the LGB model. This process of fine-tuning the model using grid search ensured that the LGB model is properly trained to achieve the optimal performance in HT detection. By exploring the hyperparameter space and considering all provided parameter combinations, the grid search methodology enabled us to make data-driven decisions and select the most effective parameter values. This rigorous approach ensures that the LGB model is fine-tuned to attain the best possible performance in HT detection.

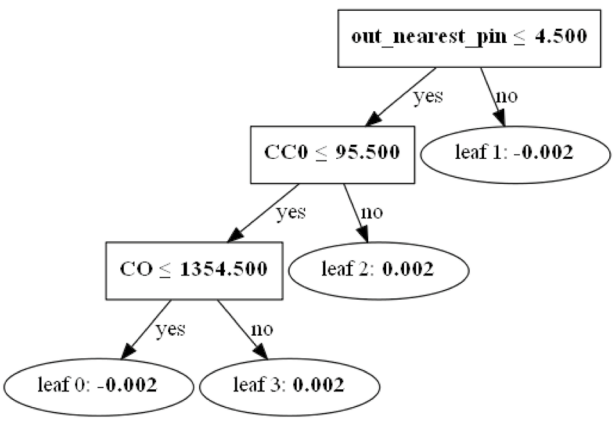


Fig. 3 LGB leafwise tree growth

We found that the following parameter values provide the best performance for HT detection: *n_estimators* = 5000 which specifies the number of boosting iterations or the number of decision trees to be built, *max_depth* = 4 which controls the maximum depth of each individual decision tree in the ensemble, num_leaves=40 determines the maximum number of leaves (terminal nodes) in each decision tree. Further, the regularization (reg_λ) is set to 0.5, which prevents the model from overfitting. The learning rate (α) is set to 0.01, which needs to be set properly because if its value is too small, it slows down the process else, it skips the global optimum solution. The model training takes place with these optimal parameters, and the loss becomes minimized iteratively, which finally provides the robust tree for accurate detection.

4.2.2 New Feature Selection Method

Structural and SCOAP features define the essential traits of TI/TF nets, but not all the features are equally important and contribute to the prediction. Partially relevant, irrelevant, and redundant features increase the model training time and negatively impact the model performance therefore, it is necessary to remove these features to improve the model prediction. Besides, narrowing down the feature set also increases model generalizability, avoids overfitting, and decreases the model complexity by making it easier to interpret [41]. The feature selection method used in existing techniques either partition the feature-set into halves and compute F-score. Then discard a single feature from the best set and keep on repeating this procedure until the max F-score is obtained [16] or compute all the feature subsets by iteratively adding the features and choosing that subset which provides the highest accuracy [23]. However, both these methods are time-consuming because of their partial/complete iterative process and dropped out some relevant features due to random partition. Besides, the use of model-specific feature importance (FI) or relevancy method seems to be inconsistent and inaccurate because the model-specific methods are based on mean decrease impurity (MDI), which compute importance based on splitting order that may change the scores of equally important features [31]. Whereas the relevancy method ranks the feature based on mutual information scores which are affected by noise and outliers [20]. Moreover, they computed collective global FI of all samples because they view ML

models as a black box, thus lacking thorough analysis of local feature contribution for individual samples w.r.t prediction, which is different for each sample. Further, it could not capture any feature interaction, i.e., which features are interacted most and influence the prediction.

Therefore, a model agnostic approach SHAP [30] is employed, which uncovers the black-box nature of any ML model by explaining its predictions both locally and globally. It follows a game-theoretic approach that analyzes the relationship between the features to identify the influence of features on the individual sample. The usefulness of SHAP can be explained by the fact that besides providing the global FI's, it also gives insight into the local explanations, i.e., what feature values affect the model predictions per sample. SHAP possesses the intrinsic ability to understand the model and dataset traits completely and explains the prediction for any TI/TF instance x by calculating the sum of contributions of individual features present in the feature set (f_{set}). It is an additive feature attribution method that uses the local explainability property which expressed the complex prediction model $F(x)$ in terms of simple linear function explanatory surrogate model $G(z')$ as shown:

$$G(z') = \phi_0 + \sum_{i=1}^{m} \phi_i z'_i \quad z' \epsilon (0,1)^m \tag{3}$$

where m is the number of Trojan features present in f_{set}, z'_i indicates whether Trojan feature i is present (1) or not (0), ϕ_i estimates how the feature i contributed in the final prediction and ϕ_0 indicates the model decision when all the features are not considered. SHAP utilizes the concept of Shapley values [29], but with an approximation for faster computation of ϕ_i. It evaluates the importance of feature i by calculating the difference between prediction made by model $F(x)$ with and without feature i across all possible feature coalitions, which is expressed as:

$$\phi_i = \sum_{s \subseteq m \backslash \{i\}} \frac{|s|!(|m| - |s| - 1)!}{m!} [F_{(s \cup \{i\})}(x_{(s \cup \{i\})}) - F_s(x_s)] \tag{4}$$

where $s \subseteq m$ is the feature subset. Weighting factor $\frac{|s|!(|m| - |s| - 1)!}{m!}$ counts the number of permutations of s. Prediction difference of two models $F_{(s \cup \{i\})}(x_{(s \cup \{i\})}) - F_s(x_s)$ represents the marginal contribution of including feature i into s. This difference is computed for all possible subsets $s \subseteq m \backslash \{i\}$. Intuitively, model G can interpret both the local and global importance by utilizing the average feature contributions across all data.

Further, a new quartile based feature selection method is proposed which identify the optimal feature subset using SHAP FI's. Proposed method identify the importance of each feature present in the $f_{set} = (f_1, f_2, .., f_k, .. f_m)$ using SHAP and then sorted the obtained global FI's $fi = (fi_1, fi_2 ..., fi_k, .. fi_m)$ in ascending order. Afterwards fi is partitioned into three quartiles Q_1, Q_2 and Q_3 as shown in Fig. 4 which represents the median of the complete fi, first smallest $fi's$ from $(fi_1 - fi_{(k-1)})$ before Q_1 and last largest $fi's$ from $(fi_{(k+1)} - fi_m)$ after Q_1 respectively.

Quartiles are chosen because it divides the fi into four equal groups, thus providing a clear view of the lower and higher FI extremes groups. Moreover, quartiles use the median for data division which is robust and less affected by the outliers and noise [34]. Since our main aim is to identify the optimal feature set (f_{optm}), the proposed method focuses on the Q_2 quartile, which represents the median of the lowest FI groups. We chose Q_2 as the cut-off point and discarded all the features whose $fi < Q_2$. Now, the model is retrained with the newly obtained feature subset (f_s) and the new accuracy (n_{acc}) and f-measure (n_{fmeas}) is evaluated on validation dataset (VaD). The obtained n_{acc} and n_{fmeas} is compared with the previously evaluated accuracy (p_{acc}) and f-measure (p_{fmeas}) which is initially calculated on f_{set}. If the $n_{acc} > p_{acc}$ and $n_{fmeas} > p_{fmeas}$, then the obtained f_s is better than f_{set}. This procedure is repeated and stopped when $n_{acc} < p_{acc}$ and $n_{fmeas} < p_{fmeas}$, which implies that previously obtained f_s is better than the current f_s and it becomes the final f_{optm}. Finally, the model is again retrained with the previous f_{optm}, which detects the HT with the highest accuracy on VaD. The proposed method is robust, reliable, and faster than the discussed feature selection methods as it does not partition the FI randomly and retrains the model only a few times. The whole proposed technique is presented in the form of an HT detection algorithm discussed in the next subsection.

4.3 Proposed HT Detection Algorithm

The proposed Algorithm 1 takes the IC dataset D which consists of $'N'$ samples and $'y'$ true labels, number of trees (T), k-fold rounds (K) as an input and provides test set accuracy TS_{acc} as an output. The LGB model is initialized with

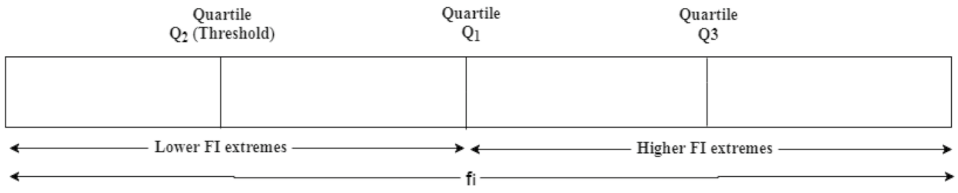


Fig. 4 Quartile division of SHAP global FI

constant value $'c'$ which initially optimize the L. Preprocessing is applied on D (discussed in Section 5.1) and it is divided into training (TrD) and testing (TeD) datasets. TrD is shuffled and divided into K folds and during every fold, $(K-1)$ folds are taken as training set TrD_k and b^{th} fold as validation VaD datasets and sampling is applied on TrD_k to obtain balanced training dataset TrD'_k. Now, during every iteration t, TrD'_k is processed by each LGB tree which apply EFB and GOSS to minimize L. Now, the validation accuracy (V_{acc}) and f-measure (V_{fmeas}) is computed on VaD for every fold. Finally, the model is trained and mean validation accuracy (MV_{acc}) and f-measure (MV_{fmeas}) is obtained. Afterwards, SHAP FI is computed and stored in fi to compute f_{optm}. Besides, MV_{acc}, MV_{fmeas}, f_{set} and fi are assigned to p_{acc}, p_{fmeas}, f_{optm} and fi_{optm}. Further, Q_2 is computed on fi_{optm} and features whose fi_{optm} satisfy the condition are selected from f_{optm} and stored in f_s and fi_s. LGB model is retrained on obtained f_s that compute MV_{acc} and MV_{fmeas} which is stored in n_{acc} and n_{fmeas}. If it satisfies the condition given in line no 25, then it shows that obtained f_s gives better model performance than previous f_{optm}. This whole process i.e. line number $(19-26)$ is repeated and values are updated until the condition is false. Now, the previous best feature subset f_{optm} is retrieved and LGB model is retrained on it. Finally, TS_{acc} is evaluated on TeD.

1: **Input** : Input Dataset (D), k-fold cross-validation rounds (K), no. of trees (T), no. of samples (N);
2: **Output**: Test-set accuracy (TS_{acc});
3: Initialize $P_0(x_n) = \underset{c}{\operatorname{argmin}} \sum_{n=1}^{N} L(y_n, c)$; ▷ *constant c*
4: Perform preprocessing on D and divide it into Training (TrD) & Testing (TeD) datasets.
5: Shuffle TrD and split it into K folds.
6: **for** $(b = 1$ to $K)$ **do** ▷ *k-fold cross validation*
7: Take $(K-1)$ folds as training set (TrD_k) and (b^{th}) fold as validation set (VaD).
8: Apply sampling on TrD_k to get balanced TrD'_k set.
9: **for** $(t = 1$ to $T)$ **do** ▷ *LGB training*
10: TrD'_k is passed into t^{th} iteration and mutually exclusive features are combined using EFB.
11: GOSS select the large and small gradient subsamples and $Vag_m(o)$ is computed using eqn (2).
12: Update the current base model P_t using eqn (1).
13: **end for**
14: V_{acc} and V_{fmeas} is computed on VaD
15: **end for**
16: LGB model is trained. MV_{acc} and MV_{fmeas} is obtained.
17: Compute SHAP FI on f_{set} and store them in fi array. ▷ *New feature selection method*
18: Assign $p_{acc} = MV_{acc}$, $p_{fmeas} = MV_{fmeas}$, $f_{optm} = f_{set}$, $fi_{optm} = fi$ and identify the quartile Q_2 of fi_{optm}.
19: **for** $i = 1$ to $len(fi_{optm})$ **do**
20: **if** $(fi_{optm}[i] >= Q_2)$ **then**
21: Corresponding features from f_{optm} are extracted based on selected importances and stored in f_s & fi_s.
22: **end if**
23: **end for**
24: Retrain model using f_s, compute MV_{acc} and MV_{fmeas}. Assign it to n_{acc} and n_{fmeas}.
25: **if** $((n_{acc} > p_{acc})$ && $(n_{fmeas} > p_{fmeas}))$ **then**
26: Update the parameters $p_{acc} = n_{acc}$, $p_{fmeas} = n_{fmeas}$, $f_{optm} = f_s$, $fi_{optm} = fi_s$ and identify Q_2 of new fi_{optm}. Repeat the steps (19) to (26) until the condition is false.
27: **else**
28: Fetch the previous f_{optm} and retrain the model.
29: Compute TS_{acc} using TeD.
30: **end if**
31: **Return:** TS_{acc};

Algorithm 1 Proposed HT detection algorithm

5 Experimental Results and Analysis

This section presents the experimental setup followed by the results and comparative analysis of the proposed technique.

5.1 Dataset Description and Evaluation Measures

Dataset is created using 16 Trust-Hub benchmarks [37] that contains combinational/sequential functional Trojans. Further, to make the DeTrust benchmarks [53], we perform the modification in 10 Trust-Hub benchmarks by inserting the one flip-flop at each gate output of the trigger circuit as suggested by [35]. Similarly, to show the Trojan designed by DeTest [52], we modify the one Trust-Hub benchmark $s38417-T100$ by inserting the given logic in this circuit. Finally, [35] also suggests to make always-on-Trojan by removing the trigger part of the Trust-Hub $s38417-T300$ benchmark whose payload is a ring oscillator Trojan. Afterward, a python script is written to extract the SCOAP and structural features, which converts the Verilog netlist of these benchmarks into bench format. The converted netlist is fed as an input to the structural features python script and Testability measurement tool [38] which extracts the features from each benchmark. All the extracted features are stored, and each net is labeled as 0/1 accordingly to create datasets, where 0 indicates TF class & 1 indicates TI class.

Further, preprocessing is applied, which checks for duplicate values and identifies the redundant features using Kendall's Tau correlation (τ) [1]. We found that four features *out_loop*_2, 3, 4, 5 are highly correlated ($\tau > 0.95$) to *in_loop*_2, 3, 4, 5, so these four *out_loop* features are discarded. Now, the datasets are divided into training (90%) and testing (10%) sets using stratified train-test split and k-fold cross-validation (cv) is applied which randomly choose the (10%) VaD five times ($k = 5$) from the (90%) TrD for better learning. Further, combined sampling SMOTE-Tomek

Links is applied to balance both the classes. However, to avoid over-optimism [4], where similar data patterns coexist in both TrD/VaD, the sampling technique is applied only on the remaining (80%) TrD during k-fold cv to prevent the occurring of similar data in VaD. Finally, the proposed technique (preprocessing + sampling + feature selection + LGB model) is implemented in python using imblearn [26] and scikit learn [33] libraries.

The following evaluation metrics are used to compute the performance of the proposed technique, Accuracy = $\frac{(TP+TN)}{(TP+FP+TN+FN)}$, represents the correctly predicted TI/TF samples out of total ones, Precision = $\frac{TP}{(TP+FP)}$ shows the correctly predicted TI samples out of total predicted TI samples, Recall (True positive rate) = $\frac{TP}{(TP+FN)}$/ True negative rate (TNR)= $\frac{TN}{(TN+FP)}$, gives the percentage of TI/TF samples correctly predicted as TI/TF. Similarly, False positive rate (FPR)= $\frac{FP}{(TN+FP)}$ /False negative rate (FNR)= $\frac{FN}{(FN+TP)}$, shows how many TF/TI samples are wrongly predicted as TI/TF, and F-measure= $\frac{2\times P\times R}{(P+R)}$ is the harmonic mean of precision and recall. Further, to identify model bias, we use Receiver operating characteristics and Area under curve (ROC-AUC) score, which shows how good the LGB model is in TI/TF class separation.

5.2 Simulation Results and Analysis

5.2.1 SHAP Global and Local FI Analysis

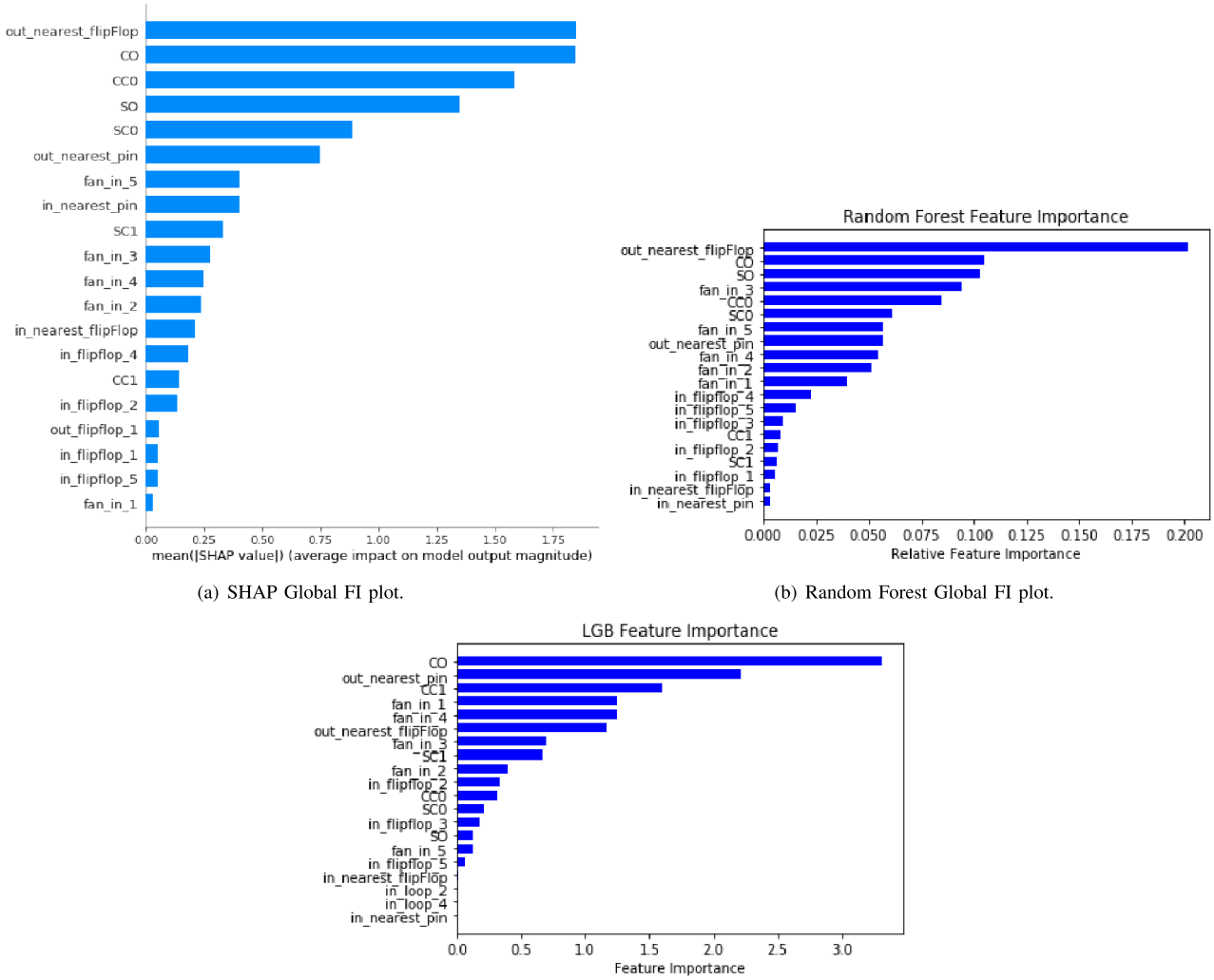
A thorough SHAP FI analysis has been performed on Trust-Hub benchmarks, and the LGB model is initially trained with all 31 features. The global FI plots of SHAP, RFC, and LGB are shown in Fig. 5, where the x and y-axis represent the mean SHAP/FI values and top 20 features arranged in decreasing order of their importance. It can be analyzed from the Fig. 5(a), (b) that out_nearest_flipFlop is the most influential feature for both SHAP and RFC having 1.75 and 0.200 FI score respectively, whereas feature *CO* is most important for LGB as shown in Fig. 5(c). Similarly, feature *SC*1 contribute more in SHAP and LGB but not in RFC, whereas in_nearest_pin is of medium importance for SHAP but contributes little for both RFC and LGB, thus ranked least in the plot. This difference in FI occurs because RFC and LGB prioritize those features, which either reduces the mean decrease in impurity factor or is best for splitting during training, whereas SHAP analyzes each feature contribution during prediction.

These plots show the collective global FI on the whole dataset.

However, more information can be identified using the SHAP summary plot shown in Fig. 6. It collectively shows how each feature impact on prediction of all samples by combining the above FI's with feature effects, thus providing a more intrinsic view. The collection of several red and blue points shows each sample's high/low SHAP value w.r.t each feature. Where *x* represents the SHAP values which shows the distribution of per feature impact on every sample, and the *y* axis shows the top 20 features. We have observed that data points with low out_nearest_flipFlop values (blue) fall into the positive SHAP value area, which influences the model to predict those points as TI, whereas points that fall into negative SHAP area possess high values (red), which force the model prediction towards TF class. In contrast, points that possess higher values for features CO, CC0, SO, and SC0 also have positive SHAP values and influence the model to predict TI and vice versa. It can be seen that model learning goes well because TI samples possess large SCOAP values, whereas mostly data points that are TF falls into the left side of the plot. However, some points also possess low values of the above SCOAP features but mixed with high-value points on the right side, which shows those points are TI nets having low SCOAP values. Similarly, other features impact the data points towards correct prediction, but some data points are clustered around zero SHAP values for features CC1, in_flipFlop_2, out_flipFlop_1, in_flipFlop_1, in_flipFlop_5 and fan_in_1 which shows that these features do not have much impact on the model prediction process.

Further, SHAP dependence plots of the top three features are shown in Fig. 7 which provides the in-depth local details of how pairwise feature interactions impact the prediction of each sample. The x and y-axis show the top three features and their SHAP values, whereas it automatically chooses the other feature for interaction shown on the right side with color ranges from high to low. It can be analyzed from Fig. 7(a) that out_nearest_flipFlop mainly interacts with fan_in_5, and the data points are dispersed for different feature values. Further, fan_in_5 highly interacts with out_nearest_flipFlop when its value is minimum, and this interaction is significantly reduced when the out_nearest_flipFlop value increases, which also clarify that nets have larger fan-ins and flipflop exist near output may be TI. Similarly, data points are shown in Fig. 7(b), (c), shows a positive linear relationship, and the corresponding features CO & CC0 interacts with fan_in_2 & CO respectively. It can be analyzed that in Fig. 7(b), features CO highly interacts with feature fan_in_2 when their values are high/medium, which increases the probability to predict that sample as TI. Similarly, in Fig. 7(c), feature CC0 mainly interacts with CO when its value is 0 or between $(0-1000)$, which indicates that some samples having low CC0 values or vice versa may be TI.

Finally, SHAP force plots of individual TI and TF samples are shown in Fig. 8, which identifies how feature values locally impact individual sample prediction. It visualizes SHAP local feature contribution values as forces that push the model prediction towards TI /TF class, where base value and f(x) are the average and expected SHAP



(a) SHAP Global FI plot.

(b) Random Forest Global FI plot.

(c) LGB Global FI plot.

Fig. 5 Global FI plots of SHAP, Random Forest and Light GBM on TrustHub benchmarks

prediction scores, besides the axis, containing feature and SHAP values. Higher f(x) value lead the model to predict TI and vice-versa. Features represented in red color force the model prediction towards higher value i.e. model predict 1 (TI class) whereas features presented in blue color pushes the model prediction towards lower value i.e. model predict 0 (TF class). The Force plot of the TF sample is shown in Fig. 8(a), it can be analyzed that its base value and predicted SHAP score are nearly 0 and −8.74, whereas feature in_nearest_pin only pushed this sample with higher force (red) towards TI class. However, features CC0, CO, SC0, SO, out_nearest_fipFlop, and fan_in_5 collectively apply lower force (blue) to push this sample towards its expected prediction value, i.e., correct TF class. Similarly, Fig. 8(b) shows the force plot of the TI sample where in_nearest_pin pushed the model prediction towards TF class. However, the remaining features pushed this sample towards the TI class with a higher force to shift its prediction from base value (nearly 0) to its expected value (6.88).

5.2.2 Simulation Results of New Feature Selection Method

To identify f_{optm} of Trust-Hub benchmarks, we utilized the obtained SHAP global FI's (fi), the respective scores of all 31 features are shown in Table 4. It can be seen that features out_loop_1, in_loop_1,2,3,4 and out_flipflop_4 possess zero importance scores, which shows they contribute nothing during prediction thus discarded. The obtained new MV_{acc} and MV_{Fmeas} are 98%, which is greater than the previous one, which is 97% and 97.7%. Afterward, Q_2 is computed from the remaining fi scores, and those features are discarded whose $fi < Q_2$ in every round as shown in Table 5. It can be analyzed that in round 1, the obtained Q_2 values is 0.052 and the features whose

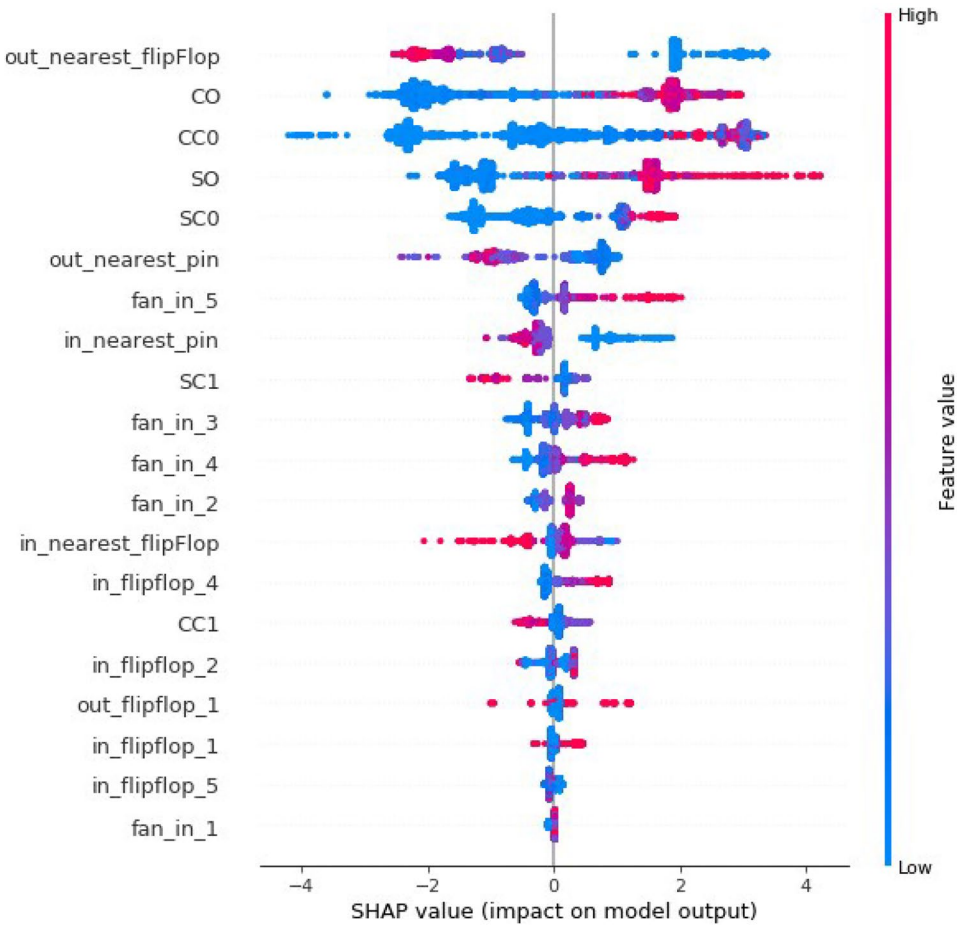


Fig. 6 SHAP Summary Plot on TrustHub benchmarks

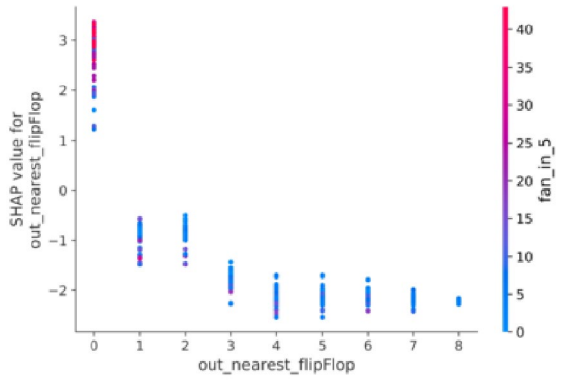
$fi >= Q_2$ are selected, and the improved MV_{acc} and MV_{Fmeas} after retraining the model with new feature set is 99% and 97%. Similarly, at round 2 and 3 the respective Q_2 values are 0.161 & 0.262, and 14 features are selected in round 2 which achieve 99.48% MV_{acc} and in round 3 only 10 features are selected with 99.7% MV_{acc} and MV_{Fmeas}. Further, in fourth round $Q_2 = 0.402$ and only seven features are selected which are out_nearest_flipFlop, CO, CC0, SO, SC0, out_nearest_pin and fan_in_5 having accuracy of 99.9%. However, in the fifth round the five features are selected, but MV_{acc} and MV_{Fmeas} drop down to 99% and 98%, respectively, which eventually stop the process. Finally, the f_{optm} is obtained at round 4, and the model is again retrained with the seven features present in f_{optm}.

Besides, the comparative results on Trust-Hub benchmarks are shown in Table 6, it can be seen that [16] selects 9 features (out_nearest_flipFlop, CO, SO, fan_in_3, CC0, SC0, fan_in_5, out_nearest_pin, fan_in_4) in 12 rounds and achieved 99.7% MV_{acc} and 95.2% MV_{Fmeas}. Whereas [23] selects 10 features but took 35 rounds due to its iterative process. However, [39] only take six rounds and selects five features but provides 96% MV_{Fmeas} because it is affected by multicollinearity thus throws out some important features. In contrast, proposed method select the best f_{optm} amongst others in only five rounds and also provides the highest MV_{acc} and MV_{Fmeas}. Similarly, eight best features (CC0, out_nearest_flipFlop, SO, CO, SC0, in_nearest_pin, in_flipflop_5, fan_in_5) are selected for DeTrust benchmarks.

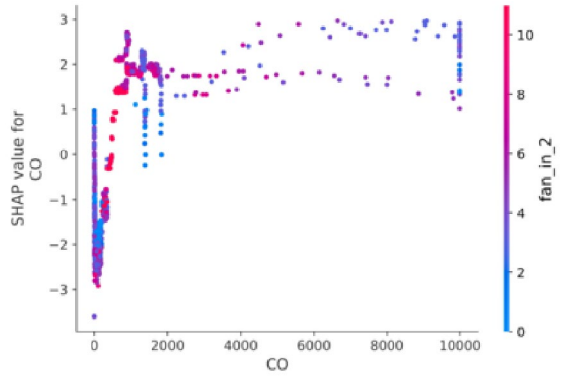
5.2.3 Simulation Results of Proposed HT Detection Technique

The LGB model is trained on the obtained f_{optm}, the training process is shown in Fig. 9, it can be observed that initially, the validation loss (0.10) is higher than the training loss (0.05) but after every iteration, it decreases continuously and become comparable to training loss which indicates that model is trained correctly and generalizable on *VaD*. Moreover, we apply early stopping, which stopped the model training at iteration 4500 instead of 5000, because the model performance on *VaD* is not improving after this iteration, thus preventing the model from overfitting.

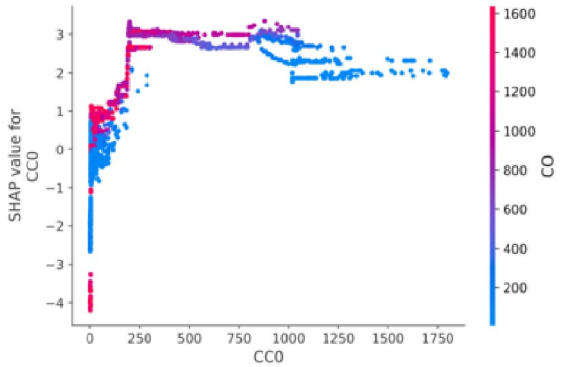
In terms of time complexity, the Light GBM (LGB) model demonstrates efficient and scalable computation, making it suitable for handling larger circuit sizes without compromising performance. Our analysis reveals a linear relationship between the circuit size and the time



(a) SHAP Dependancy plot 1.



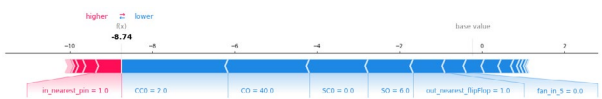
(b) SHAP Dependancy plot 2.



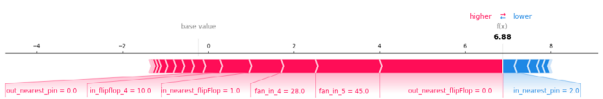
(c) SHAP Dependancy plot 3.

Fig. 7 SHAP Dependency plots of top three features (Local Feature interaction) on TrustHub benchmarks

taken by the LGB model to process and generate results, as depicted in Table 7. As the circuit size increases, there is a corresponding increase in the processing time. For example, the LGB model takes 383,404 microseconds for the RS232-T1000 benchmark, while it takes significantly more time, 4 s, 26748 microseconds for larger benchmark s38584-T300. This linear relationship indicates that the time complexity of the LGB model is directly proportional to the circuit size. Despite the increase in processing time with larger circuit sizes, it is important to note that the LGB model still exhibits notably faster execution compared to the other models utilized in our study. For instance, when considering the combined TrustHub dataset, the LGB model completed training in only 19 s, while models such as



(a) Force plot for TF sample.



(b) Force plot for TI sample.

Fig. 8 Local interpretability: SHAP Force Plot of individual TF and TI TrustHub samples

Table 4 SHAP Global FI Score

Feature names	Feature importance
out_nearest_flipFlop	1.8497
CO	1.8442
CC0	1.5836
SO	1.3459
SC0	0.8872
out_nearest_pin	0.7455
fan_in_5	0.4035
in_nearest_pin	0.4021
SC1	0.3315
fan_in_3	0.2773
fan_in_4	0.2475
fan_in_2	0.2384
in_nearest_flipFlop	0.2130
in_flipflop_4	0.1814
CC1	0.1414
in_flipflop_2	0.1331
out_flipflop_1	0.0565
in_flipflop_1	0.0534
in_flipflop_5	0.05277
fan_in_1	0.0293
out_flipflop_2	0.0263
in_loop_5	0.0217
in_flipflop_3	0.0182
out_flipflop_3	0.0063
out_flipflop_5	0.0015
out_loop_1	0.0000
in_loop_4	0.0000
in_loop_2	0.0000
in_loop_1	0.0000
out_flipflop_4	0.0000
in_loop_3	0.0000

Random Forest (RFC), XGBoost, Support Vector Machines (SVM), and Neural Networks (NN) took 24, 36, 49, and 60 s, respectively. This efficiency can be attributed to the underlying design and algorithmic optimizations employed by LGB model, which enable it to handle large datasets and complex feature spaces more efficiently.

Afterward, the trained model gives predictions on the TeD. Table 8 shows the simulation results of the proposed technique on 17 Trust-Hub benchmarks. It can be analyzed from the results that the proposed technique provides TPR and TNR between (98 − 100%) in 14 Trust-Hub benchmarks which indicates that almost all the TI and TF samples are detected correctly from these benchmarks. However, proposed technique misclassified some TI samples from RS232-T1500, s35932-T300 and s15850-T100 benchmarks, thus obtain 92.3%, 93.3% & 90.3 TPR respectively. The obtained higher FNR rates 7.69%, 6.6% & 9.67% also confirmed this. The observed decrease in False Negative Rate (FNR) can be attributed to the presence of a smaller number of Trojan Inserted (TI) samples in the test dataset compared to Trojan Free (TF) samples. As a result, if the majority of the TI samples are correctly predicted and only a few TI samples are misclassified, it can lead to a higher FNR compared to TF. This pattern can be analyzed from the confusion matrix (CM) shown in Table 9 for the TrustHub circuits exhibiting higher FNR. For instance, in the RS23-T1500 benchmark, the CM shows 494 True Negative (TN) samples, 12 True Positive (TP) samples, 0 False Positive (FP), and only 1 False Negative (FN) sample. Here, we can observe that only one TI sample is incorrectly predicted as TF. However, since there are only 12 TI samples in total, this misclassification contributes to a higher FNR.

Similarly, in the s35932-T300 and s15850-T100 circuits, 1 TI sample and 3 TI samples, respectively, are misclassified as TF. Due to the smaller number of TI samples in these circuits, the FNR appears relatively higher compared to the False Positive Rate (FPR), as TF samples are more numerous, resulting in a lower FPR when only a small number of samples are misclassified. In contrast, if the same small number of samples are misclassified as TI, it leads to a higher FNR due to the smaller number of TI samples overall. Finally, it can be analyzed that the observed variation in FNR compared to FPR in the mentioned circuits can be attributed to the differing number of TI and TF samples present in the test dataset. The smaller number of TI samples can lead to a higher FNR when only a few samples are misclassified, while the larger number of TF samples provides a lower FPR when only a small number of samples are misclassified.

Table 5 New Feature Selection Method Results on Trust-Hub benchmarks (%)

Rounds	Q2	Discarded features	Mean Vacc	Mean F-meas
1	0.052	fan_in_1, out_flipflop_2, in_loop_5, in_flipflop_3, out_flipflop_3, out_flipflop_5	99	97
2	0.161	CC1, in_flipflop_2, out_flipflop_1 in_flipflop_1, in_flipflop_5	99.48	99.49
3	0.262	fan_in_4, fan_in_2, in_nearest_flipFlop in_flipflop_4	99.7	99.7
4	0.402	in_nearest_pin, SC1, fan_in_3	99.9	99.9
5	0.816	out_nearest_pin, fan_in_5	99	98

Table 6 Comparative results of new feature selection method on TrustHub benchmarks (%)

Techniques	Rounds	Selected Features	Mean Vacc	Mean F-meas
Hasegawa et al. [16]	12	9	99.7	95.2
Kok et al. [23]	35	10	98	97
Sharma et al. [39]	6	5	97	96
Proposed method	5	7	99.9	99.9

Besides, the obtained ROC-AUC score lies between (95 – 100%), which represents that model prediction is not biased towards the majority TF class. Moreover, there are two benchmarks, s35932-T200 and s38584-T100, whose Trojans possess low SCOAP values like TF nets which cleverly bypassed the SCOAP-based techniques. However, since the proposed technique incorporates structural features, it achieves 0% FPR/FNR, which means that all the low SCOAP TI samples are easily detected along with TF samples.

Further, the ROC curves of both these benchmarks are shown in Fig. 10 which plots the performances of the no skill model and perfectly skill LGB model against TPR and FPR. It can be seen from the Fig. 10(a), (b) that the proposed technique approximately detect all TI/TF nets from these benchmarks by covering the area of 100% and 99.98% respectively, which is very high as compared to the no skill model which only covers 50% area. Upon analyzing the results, we observe that the proposed technique achieves excellent performance in terms of recall on TrustHub benchmarks. In the best-case scenario, it achieves a perfect recall rate of 100%, indicating that it successfully identifies all the true positive samples. In the worst-case scenario, the recall rate is 90%, indicating that it may miss a small portion of the true positive samples. On average, the proposed technique achieves a recall rate of 98.40% across all the benchmarks,

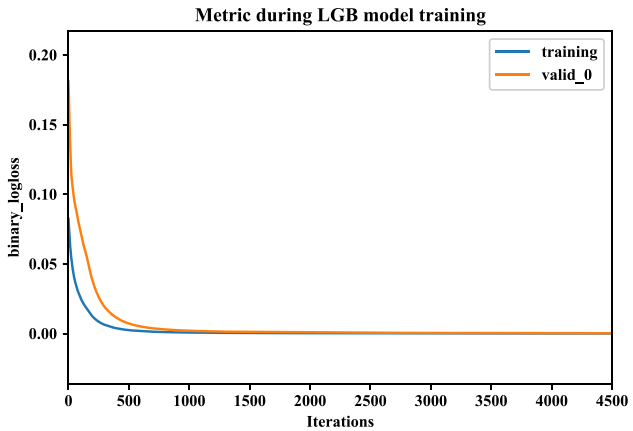


Fig. 9 LGB model training loss with early stopping

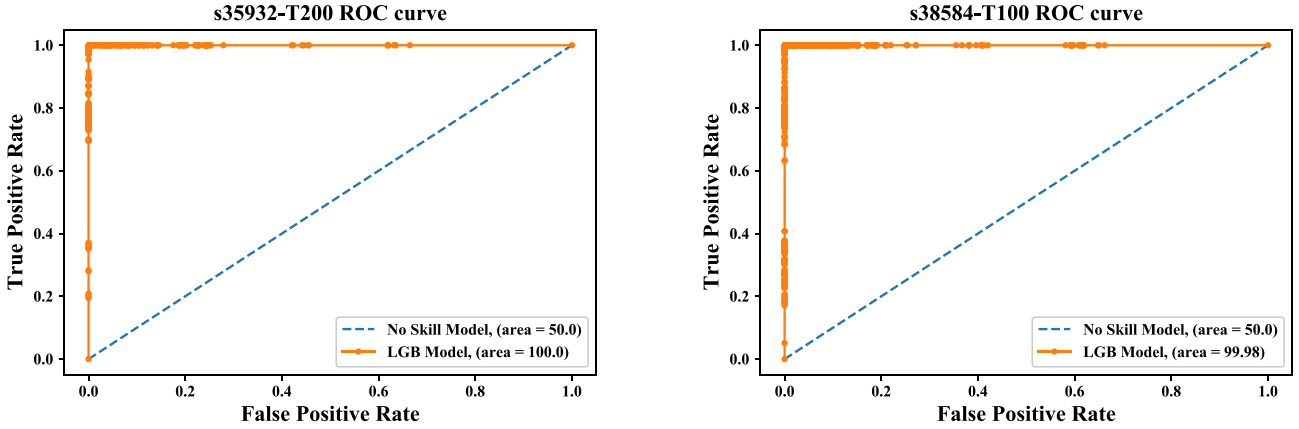
Table 7 Training Time of LGB model on different TrustHub circuits

TrustHub Benchmarks	Training time
RS232-T1000	383404 microseconds
RS232-T1100	476577 microseconds
RS232-T1200	490951 microseconds
RS232-T1300	439998 microseconds
RS232-T1400	577096 microseconds
RS232-T1500	774231 microseconds
RS232-T1600	839915 microseconds
s15850-T100	1 s, 739366 microseconds
s35932-T100	2 s, 28332 microseconds
s35932-T200	2 s, 192710 microseconds
s35932-T300	2 s, 247536 microseconds
s38417-T100	2 s, 97314 microseconds
s38417-T200	2 s, 49864 microseconds
s38417-T300	2 s, 954530 microseconds
s38584-T100	2 s, 483484 microseconds
s38584-T200	2 s, 453586 microseconds
s38584-T300	4 s, 26748 microseconds

which demonstrates its overall effectiveness in detecting hardware Trojans.

Further, the average comparative results of 17 Trust-Hub benchmarks are shown in Table 10. It can be analyzed from the results that structural feature-based technique [25] provides lower recall (69%) and higher FNR (30.7%) as compared to [16, 46] which shows that the trigger net structural features used by them are not good enough to detect TI samples from these benchmarks. Though structural-based techniques [16, 46] provides comparable performance but higher FPR (8.35% & 6.23%) & FNR (11.39% & 12.25%) indicates that many TI & TF samples are misclassified, thus structural features alone are not sufficient. Similarly, it can be observed that there is not much difference between the accuracy obtained by SCOAP features-based techniques [24, 35, 42], which shows that accuracy alone is not a sufficient measure to evaluate the performance. However, these techniques almost correctly detect TF samples by providing TNR between (96%-98%) but could not identify the TI samples accurately. The ROC-AUC score, which lies between (90%-93%), also indicates that many TI samples are missed.

On the other hand, [23] provides quite good results due to the use of both structural and SCOAP features but lacks in precision (93%), recall (96.7%), f-measure (94.8%) and FNR (3.15%) due to lack of best feature set. In contrast, our proposed technique achieves on-an-average accuracy of 99.4%, the lowest FPR/FNR of 0.33%, and 1.52%, respectively. Further, it covers the 98.76% area and provides the highest Recall (98%), TNR (99.4%), and f-measure (98%), which shows that mostly TI and TF samples are predicted correctly. Finally, the proposed technique provides 16.68%, 12.74%,



(a) ROC curve of s35932-T200.

(b) ROC curve of s38584-T100.

Fig. 10 ROC curves of s35932-T200 and s38584-T100

and 1.7% higher Recall than existing structural, SCOAP, and combined features-based techniques.

Further, the average comparative results of techniques [39] and [36] with the proposed technique on TrustHub benchmarks are presented in Table 11. Upon analysis, it can be observed that the proposed technique yields results that are quite comparable to our previous work [39]. However, it is worth noting that the previous technique exhibits a higher false positive rate (FPR) due to the class weighting bias towards the minority TI class, leading to the misclassification of TF samples as TI. Furthermore, the previous approach performs well on TrustHub benchmarks but struggles to accurately detect Trojans in other benchmarks, specifically DeTest. This is because these Trojans are designed to evade detection by reducing the SCOAP values of Trojan nets to 10 percent. Moreover, the previous approach lacks model interpretability and fails to provide insights into the inner workings of the model. In contrast, the proposed technique effectively combines different Trojan features to cover a wide range of Trojan types and provides model interpretability, thus shedding light on the model's decision-making process. Similarly, when compared to the improved COTD technique [36], the proposed technique demonstrates higher accuracy of 99.4%, better recall of

Table 8 Experimental results of the proposed technique on Trust-Hub benchmarks (%)

Trust-Hub Benchmarks	Accuracy	Precision	Recall (TPR)	F-measure	ROC-AUC score	TNR	FPR	FNR
RS232-T1000	100	100	100	100	100	100	0	0
RS232-T1100	98	90	100	90	99.09	98.19	1.8	0
RS232-T1200	99.35	98.6	100	99.3	99.38	98.76	1.23	0
RS232-T1300	100	100	100	100	100	100	0	0
RS232-T1400	100	100	100	100	100	100	0	0
RS232-T1500	99.80	100.00	92.31	96.00	96.15	100	0.00	7.69
RS232-T1600	99.3	93.33	100	96.55	99.66	99.33	0.66	0
s35932-T100	99.97	99.94	100	99.97	99.97	99.94	0.054	0
s35932-T200	100	100	100	100	100	100	0	0
s35932-T300	99.84	100	93.3	96.5	96.67	100	0	6.6
s38417-T100	99.96	99.96	99.96	99.95	99.96	99.95	0.042	0.04
s38417-T200	98	98.5	98	98	98	98.59	1.4	1.4
s38417-T300	100	100	100	100	100	100	0	0
s38584-T100	99.98	99.95	100	99.99	99.98	99.94	0.04	0
s38584-T200	97	96	99	97	95.12	95.48	0.045	0.57
s38584-T300	99.97	99.94	100	99.97	99.97	99.94	0.058	0
s15850-T100	98.8	96.55	90.322	93.33	95	99.69	0.3	9.67
Average	99.41	98.39	98.40	98	98.76	99.40	0.33	1.52

Table 9 TrustHub circuits with higher FNR

TrustHub circuits	CM	FNR
RS232-T1500	[494 0] [1 12]	7.69
s35932-T300	[630 0] [1 14]	6.6
s15850-T100	[328 1] [3 28]	9.67

98.3%, and lower FPR of 0.33. These results indicate that the proposed technique performs comparatively well in terms of detection performance.

The comparative results of the proposed technique and the Hardware IP assurance technique [12] are presented in Table 12. The evaluation of these results is based on the number of false positives (FP) and false negatives (FN). Upon analysis, it is evident that the proposed technique outperforms [12] in terms of both FP and FN. In the RS232-T1000 benchmark, the proposed technique achieves 0 FP and 0 FN, while the [12] results in 4 FPs. Similarly, in the RS232-T1100, RS232-T1200, RS232-T1400, and RS232-T1600 benchmarks, the proposed technique achieves 0 FN, whereas the [12] exhibits non-zero FN values in these respective benchmarks. However, in the s38417-T100 and s38417-T200 benchmarks, the proposed technique records 1 FN, whereas the [12] achieves 0 FN. It is important to note that overall, the proposed technique demonstrates superior performance compared to [12].

Further, to check the efficacy of the proposed technique, we also evaluate the results on 10 DeTrust benchmarks, as shown in Table 13. It can be observed that the proposed technique obtains accuracy between (96 – 100%) on these benchmarks, thus achieving on-an-average 99.21% accuracy and lower FPR (0.3%). The proposed technique detects all TI samples from five DeTrust benchmarks by providing 0% FNR. In some cases, our proposed technique is unable to correctly identify certain TI samples from benchmarks RS232-T1000, T1100, T1500, T1600, and s38417-T200, resulting in higher FNR of 16.66%, 12.5%, 11.11%, and 6.25%, respectively. The higher FNR can be attributed to the presence of a small number of TI samples compared to TF samples in the test dataset. For instance, let's consider the RS232-T1000 circuit, which consists of 91 True Negative (TN) samples, 5 True Positive (TP) samples, 0 False Positive (FP), and 1 False Negative (FN) sample. In this case, only 1 TI sample is incorrectly predicted as TF, resulting in a higher FNR of 16.66%. This higher FNR is influenced by the smaller number of TI samples compared to TF samples in the circuit. It is worth noting that the presence of a smaller number of TI samples can impact the FNR calculation, as misclassifying even a few TI samples can result in a higher FNR percentage. This is particularly evident when the number of TI samples is significantly smaller than the number of TF samples. Finally, it has been observed that the larger FNR values observed in these specific benchmarks are a result of misclassifying a small number of TI samples. The imbalance between the number of TI and TF samples in test dataset in these cases contributes to the higher FNR percentages.

Furthermore, we have evaluated the performance of the proposed technique on DeTest benchmarks, which are specifically designed to neutralize the effects of SCOAP

Table 11 Average comparative results of existing and Proposed Techniques

Evaluation measures	Sharma et al. [39]	Salmani et al. [36]	Proposed Technique
Accuracy	99.03	96.92	99.4
Precision	98.27	95.86	98.3
Recall	99.06	98.13	98.4
F-measure	98.48	96.65	98
ROC-AUC score	98.76	96.54	98.76
TNR	98.84	96.52	99.4
FPR	1.152	1.63	0.33
FNR	0.93	1.28	1.52

Table 10 Average comparative results of different evaluation measures on Trust-Hub benchmarks (%)

Evaluation measures	Structural feature based techniques			SCOAP feature based techniques			Combined features based technique	
	Kurihara et al. [25]	Hasegawa et al. [16]	Wang et al. [46]	Salmani et al. [35]	Kok et al. [24]	Tebyanian et al. [42]	Kok et al. [23]	Proposed technique
Accuracy	94.06	90	92.4	97.2	96.6	96.7	97.8	99.4
Precision	81.41	89	74.4	79	84.13	93.5	93	98.3
Recall	69	88.4	87.75	82.93	83.64	90.4	96.7	98.4
F-measure	71.6	88.4	77.60	79	83.09	91.9	94.8	98
ROC-AUC score	81	89.4	90.89	90.7	90.27	93.3	97.8	98.76
TNR	98.8	91.6	93.6	98.8	97.72	96.66	99.3	99.4
FPR	1.13	8.35	6.23	1.15	1.45	3.28	0.31	0.33
FNR	30.7	11.39	12.25	17	16	9.47	3.15	1.52

Table 12 Comparative Results of [12] and Proposed Technique

TrustHub Benchmarks	Gaikwad et al. [12]		Proposed technique	
	FP	FN	FP	FN
RS232-T1000	4	0	0	0
RS232-T1100	4	1	2	0
RS232-T1200	1	4	2	0
RS232-T1400	6	1	0	0
RS232-T1500	1	1	0	1
RS232-T1600	1	0	1	0
s38417-T100	6	0	1	1
s38417-T200	1	0	1	1

features. The results, as shown in Table 14, demonstrate the effectiveness of the proposed technique on these benchmarks. Across the five DeTest benchmarks, the proposed technique achieves recall rates ranging from 97% to 100%, indicating its ability to detect almost all TI nets present in these benchmarks. The FPR and FNR values range from 0% to 2%, suggesting that only a few TI and TF samples are incorrectly predicted. On average, the proposed technique achieves an accuracy of 98.77%, covers 98.768% area, and provides a recall of 98.838%. These results indicate that the structural features utilized in the proposed technique are effective in detecting TI nets from DeTest circuits.

Furthermore, it is important to note that structural features alone are not capable of identifying always-on Trojans, as these Trojans do not involve triggers. To address this limitation, the proposed technique incorporates SCOAP features to detect the HT in the s38417-T300 always-on Trojan circuit. By leveraging the SCOAP features, the proposed technique achieves a remarkable performance with 100% accuracy and 0% FPR and FNR in detecting the HT within this specific circuit. This highlights the adaptability of the proposed technique in utilizing different types of features to effectively detect different types of Trojans.

Furthermore, to further evaluate the efficacy of the proposed technique, we conducted tests on a set of 12 circuit benchmarks generated by MIMIC [8, 9], which encompass both combinational and sequential Trojans. The results obtained in Table 15 demonstrate the strong performance of the proposed technique in detecting HTs from these benchmarks. The proposed technique achieves impressive accuracy rates of 99-100% in most of the tested circuits. Notably, it provides 100% recall in five specific circuits, namely s953_T0000_C, s953_T0001_C, s953_T0101_S, s1196_T0000_C, and s1238_T0000_C. These results indicate that the proposed technique is highly effective in identifying the presence of HTs within these circuits. However, it is important to note that the proposed technique may exhibit higher FNR in certain circuits, ranging from 6% to 16%. This fluctuation in FNR primarily occurs due to the smaller number of TI samples in the test dataset. In such cases, even a misclassification of a small number of Trojan samples results in a higher FNR, as the TI samples are relatively fewer in number compared to the TF samples. It is worth highlighting that despite this variation, the proposed technique consistently demonstrates its capability to detect a majority of HTs in the tested circuits, as indicated by the high accuracy rates and recall values achieved. These findings further validate the effectiveness of the proposed technique in accurately detecting HTs from a diverse range of circuit benchmarks, including those with different types of Trojans and varying characteristics.

Finally, based on the analysis of the obtained results, it is evident that the incorporation of multiple features in the proposed technique significantly enhances the overall detection capability. This amalgamation of features proves to be a robust approach, particularly in scenarios where existing single feature-based techniques may fall short. The combination of features provides a more comprehensive and robust detection capability, enhancing the overall accuracy and reliability of the HT detection process. In our future work, we recognize the need to continuously adapt

Table 13 Experimental results of the proposed technique on De-Trust benchmarks (%)

De-Trust Benchmarks	Accuracy	Precision	Recall	F- measure	ROCAUC score	TNR	FPR	FNR
RS232-T1000	98.97	100	83.33	90.90	90.3	100	0	16.66
RS232-T1100	98.99	100	87.5	93.33	92.78	100	0	12.5
RS232-T1200	99.61	94.29	100	97.05	96.84	99.58	0.41	0
RS232-T1300	100	100	100	100	100	100	0	0
RS232-T1400	99.24	87.5	100	93.33	92.93	99.2	0.8	0
RS232-T1500	99.49	100	88.89	94.11	93.85	100	0	11.11
RS232-T1600	99.97	100	83.33	90.90	90.89	100	0	16.66
s35932-T200	100	100	100	100	100	100	0	0
s38417-T100	99.34	98.61	100	99.3	99.38	98.75	1.2	0
s38417-T200	96.54	99.33	93.75	96.46	96.55	99.36	0.63	6.25
average	99.215	97.973	93.68	95.54213	95.35466	99.689	0.304	6.318

Table 14 Experimental results of the proposed technique on De-Test benchmarks (%)

De-Test Benchmarks	Accuracy	Precision	Recall (TPR)	F- measure	ROC-AUC score	TNR	FPR	FNR
RS232-T1200	98.31	99.17	97.56	98.36	98.34	99.12	0.87	2.43
RS232-T1300	100	100	100	100	100	100	0	0
RS232-T1400	98.35	98.42	98.42	98.42	98.35	98.27	1.72	1.57
RS232-T1600	98.5	98.2	98.8	98.49	98.5	98.2	1.79	1.2
s38417-T100	98.72	98.26	99.41	98.83	98.65	97.88	2.11	0.58
Average	98.776	98.81	98.838	98.82	98.768	98.694	1.298	1.156

Table 15 Proposed Technique Results on MIMIC based Trojan benchmarks

New Trojan Benchmarks	Accuracy	Precision	Recall (TPR)	F-measure	ROC-AUC score	TNR	FPR	FNR
s953_T0000_C	99.2	98.33	100	99.159	99.244	98.49	1.51	0
s953_T0001_C	100	100	100	100	100	100	0	0
s953_T0100_S	99.88	100	92.85	96.29	96.428	100	0	7.142
s953_T0101_S	100	100	100	100	100	100	0	0
s1196_T0000_C	99.56	99.11	100	99.55	99.57	99.15	0.846	0
s1196_T0001_C	99.43	99.56	99.27	99.417	99.429	99.59	0.415	0.726
s1196_T0100_S	99.8	99.7	99.89	99.79	99.8	99.7	0.299	0.1011
s1196_T0101_S	98	98	98	98	98	98.7	1.3	1.4
s1238_T0000_C	100	100	100	100	100	100	0	0
s1238_T0001_C	99.82	100	99.63	99.81	99.81	100	0	0.367
s1238_T0100_S	99.82	96.42	93.103	94.736	96.521	99.94	0.06	6.89
s1238_T0101_S	99.73	99.63	99.81	99.725	99.735	99.65	0.345	0.183
Average	99.603	99.23	98.547	98.874	99.045	99.6	0.398	1.4015

and enhance the proposed technique to address emerging challenges in HT detection. This includes incorporating more features to identify newly developed Trojans such as TAAL [19] and other evolving threats. As attackers continuously devise novel techniques and insert Trojans in different forms with different triggers and payloads, it is crucial to expand the capabilities of our technique to effectively detect these new HTs. A particular challenge lies in scenarios where Trojans possess triggers situated distant from primary inputs or outputs. Our current methodology is adept at identifying Trojans in proximity to primary input or output. However, the possibility of Trojans design with triggers farther afield necessitates further exploration. To address this, we plan to diversify our technique's feature extraction and analysis mechanisms. By doing so, we aim to broaden its scope, ensuring a comprehensive coverage of Trojans irrespective of trigger location. Moreover, while our current technique excels at detecting Trojan nets within the circuit, a crucial aspect that remains to be addressed is the precise localization of these nets. Pinpointing the exact location of Trojan nets can greatly enhance our approach's efficacy. To achieve this, we intend to incorporate advanced techniques that can precisely pinpoint the locations of these Trojan-infected nets. By doing so, we will not only enhance the transparency of our approach but also improve its overall detection capabilities.

Furthermore, we understand the importance of extending the proposed technique to detect Trojans inserted in FPGA netlists, as suggested by Cruz et al. [10]. This represents a valuable direction for future research, considering the prevalence of FPGAs in various applications. By adapting and extending our technique to handle FPGA netlists, we aim to provide a comprehensive solution for HT detection across different design domains.

6 Conclusion

This paper proposed a new LGB model-based HT detection technique for the pre-silicon IC designing phase, which utilizes structural and SCOAP features to detect Trojan nets from gate-level netlist. Further, a model agnostic SHAP is utilized to interpret the model predictions by identifying the global and local feature impact on predictions. Moreover, a quartile-based feature selection method is proposed, which utilizes the SHAP FI to identify the optimal feature set in minimum model retraining rounds. Besides, combined SMOTE-Tomek

links sampling is applied, which effectively handles the class imbalance by avoiding the limitations that arise in weighting and oversampling. Finally, detailed global and local feature analysis is carried out using SHAP on Trust-Hub benchmarks, and feature interactions are captured, which impact the LGB prediction the most. Experimental analysis shows that the new feature selection method outperforms the existing methods and identifies the optimal features set, which provides the highest performance in only five retraining rounds. Moreover, the proposed technique obtained on-an-average 98.4% recall on Trust-Hub benchmarks, which is 16.68% and 12.74% higher than existing structural/SCOAP features-based techniques. Further, the proposed technique achieves 99.21% accuracy on DeTrust and effectively detects the Trojans from DeTest benchmark, always-on-Trojans and newly developed Trojan benchmarks generated by MIMIC.

Data Availability The Trust-Hub benchmarks analyzed during this study are available at https://trust-hub.org/. Besides, the Trojans benchmarks generated by MIMIC are available at https://cadforassurance.org/. Further, the DeTrust & DeTest benchmarks created during the current study are available from the corresponding author on reasonable request.

Declarations

Conflict of Interest The authors declare that there is no conflict of interest in relation to this manuscript.

References

1. Abdi H (2007) The kendall rank correlation coefficient. Encyclopedia of Measurement and Statistics. Sage, Thousand Oaks, CA, pp 508–510
2. Batista GE, Bazzan AL, Monard MC et al (2003) Balancing training data for automated annotation of keywords: A case study. In WOB, pp. 10–18
3. Bhunia S, Hsiao MS, Banga M, Narasimhan S (2014) Hardware trojan attacks: threat analysis and countermeasures. Proc IEEE 102(8):1229–1247
4. Blagus R, Lusa L (2015) Joint use of over-and under-sampling techniques and cross-validation for the development and assessment of prediction models. BMC Bioinformatics 16(1):1–10
5. Chakraborty RS, Narasimhan S, Bhunia S (2009) Hardware trojan: Threats and emerging solutions. In 2009 IEEE International High Level Design Validation and Test Workshop. IEEE, pp. 166–171
6. Chen T, Guestrin C (2016) Xgboost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785–794
7. Clark GW, Doran MV, Andel TR (2017) Cybersecurity issues in robotics. In 2017 IEEE Conference on Cognitive and Computational Aspects of Situation Management (CogSIMA). IEEE, pp. 1–5
8. Cruz J, Gaikwad P, Nair A, Chakraborty P, Bhunia S (2022) A machine learning based automatic hardware trojan attack space exploration and benchmarking framework. In 2022 Asian Hardware Oriented Security and Trust Symposium (AsianHOST). IEEE, pp. 1–6
9. Cruz J, Gaikwad P, Nair A, Chakraborty P, Bhunia S (2022) Automatic hardware trojan insertion using machine learning. arXiv preprint arXiv:2204.08580
10. Cruz J, Posada C, Masna NVR, Chakraborty P, Gaikwad P, Bhunia S (2023) A framework for automated exploration of trojan attack space in FPGA netlists. IEEE Trans Comput
11. Friedman JH (2002) Stochastic gradient boosting. Comput Stat Data Anal 38(4):367–378
12. Gaikwad P, Cruz J, Chakraborty P, Bhunia S, Hoque T (2023) Hardware IP assurance against trojan attacks with machine learning and post-processing. ACM J Emerg Technol Comput Syst
13. Goldstein LH, Thigpen EL (1980) Scoap: Sandia controllability/observability analysis program. In Proceedings of the 17th Design Automation Conference pp. 190–196
14. Hasegawaa K, Yanagisawa M, Togawa N (2017) Hardware trojans classification for gate-level netlists using multi-layer neural networks. In On-Line Testing and Robust System Design (IOLTS), 2017 IEEE 23rd International Symposium on. IEEE, pp. 227–232
15. Hasegawa K, Oya M, Yanagisawa M, Togawa N (2016) Hardware trojans classification for gate-level netlists based on machine learning. In On-Line Testing and Robust System Design (IOLTS), 2016 IEEE 22nd International Symposium on. IEEE, pp. 203–206
16. Hasegawa K, Yanagisawa M, Togawa N (2017) Trojan-feature extraction at gate-level netlists and its application to hardware-trojan detection using random forest classifier. In Circuits and Systems (ISCAS), 2017 IEEE International Symposium on. IEEE, pp. 1–4
17. Hicks M, Finnicum M, King ST, Martin MM, Smith JM (2010) Overcoming an untrusted computing base: Detecting and removing malicious hardware automatically. In Security and Privacy (SP), 2010 IEEE Symposium on. IEEE, pp. 159–172
18. Huang Z, Wang Q, Chen Y, Jiang X (2020) A survey on machine learning against hardware trojan attacks: Recent advances and challenges. IEEE Access 8:10796–10826
19. Jain A, Zhou Z, Guin U (2021) Taal: Tampering attack on any key-based logic locked circuits. ACM Transactions on Design Automation of Electronic Systems (TODAES) 26(4):1–22
20. Kalina J, Schlenker A (2015) A robust supervised variable selection for noisy high-dimensional data. BioMed Res Int 2015
21. Ke G, Meng Q, Finley T, Wang T, Chen W, Ma W, Ye Q, Liu TY (2017) Lightgbm: A highly efficient gradient boosting decision tree. Adv Neural Inf Process Syst 30:3146–3154
22. Khamitkar R, Dube R (2022) A survey on using machine learning to counter hardware trojan challenges. In ICT with Intelligent Applications. Springer, pp. 539–547
23. Kok CH, Ooi CY, Inoue M, Moghbel M, Dass SB, Choo HS, Ismail N, Hussin FA (2019) Net classification based on testability and netlist structural features for hardware trojan detection. In 2019 IEEE 28th Asian Test Symposium (ATS). IEEE, pp. 105–1055
24. Kok CH, Ooi CY, Moghbel M, Ismail N, Choo HS, Inoue M (2019) Classification of trojan nets based on scoap values using supervised learning. In 2019 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE, pp. 1–5
25. Kurihara T, Togawa N (2021) 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, pp. 1–4
26. Lemaître G, Nogueira F, Aridas CK (2017) Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. J Mach Learn Res 18(1):559–563
27. Li H, Liu Q, Zhang J (2016) A survey of hardware trojan threat and defense. Integration 55:426–437
28. Liakos KG, Georgakilas GK, Moustakidis S, Sklavos N, Plessas FC (2020) Conventional and machine learning approaches as countermeasures against hardware trojan attacks. Microprocess Microsyst 103295
29. Lipovetsky S, Conklin M (2001) Analysis of regression in game theory approach. Appl Stoch Model Bus Ind 17(4):319–330

30. Lundberg SM, Lee SI (2017) A unified approach to interpreting model predictions. In Proceedings of the 31st international conference on neural information processing systems, pp. 4768–4777
31. Man X, Chan EP (2021) The best way to select features? comparing mda, lime, and shap. J Financ Data Sci 3(1):127–139
32. Oya M, Shi Y, Yanagisawa M, Togawa N (2015) A score-based classification method for identifying hardware-trojans at gate-level netlists. In Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition. EDA Consortium, pp. 465–470
33. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V et al (2011) Scikit-learn: Machine learning in python. J Mach Learn Res 12:2825–2830
34. Qi M, Fu Z, Chen F (2016) Research on a feature selection method based on median impact value for modeling in thermal power plants. Appl Therm Eng 94:472–477
35. Salmani H (2017) Cotd: reference-free hardware trojan detection and recovery based on controllability and observability in gate-level netlist. IEEE Trans Inf Forensics Secur 12(2):338–350
36. Salmani H (2022) The improved cotd technique for hardware trojan detection in gate-level netlist. In Proceedings of the Great Lakes Symposium on VLSI 2022:449–454
37. Salmani H, Tehranipoor M, Karri R (2013) On design vulnerability analysis and trust benchmarks development. In 2013 IEEE 31st International Conference on Computer Design (ICCD). IEEE, pp. 471–474
38. Samimi S (2016) Testability measurement tool
39. Sharma R, Valivati NK, Sharma G, Pattanaik M (2020) A new hardware trojan detection technique using class weighted xgboost classifier. In 2020 24th International Symposium on VLSI Design and Test (VDAT). IEEE, pp. 1–6
40. Sturton C, Hicks M, Wagner D, King ST (2011) Defeating uci: Building stealthy and malicious hardware. In Security and Privacy (SP), 2011 IEEE Symposium on. IEEE, pp. 64–77
41. Tang J, Alelyani S, Liu H (2014) Feature selection for classification: A review. Data Classification: Algorithms and Applications 37
42. Tebyanian M, Mokhtarpour A, Shafieinejad A (2021) Sc-cotd: Hardware trojan detection based on sequential/combinational testability features using ensemble classifier. J Electron Test 37(4):473–487
43. Tehranipoor M, Koushanfar F (2010) A survey of hardware trojan taxonomy and detection. IEEE Des Test Comput 27(1):10–25
44. Venugopalan V, Patterson CD (2018) Surveying the hardware trojan threat landscape for the internet-of-things. Journal of Hardware and Systems Security 2(2):131–141
45. Waksman A, Suozzo M, Sethumadhavan S (2013) Fanci: identification of stealthy malicious logic using boolean functional analysis. In Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security. ACM, pp. 697–708
46. Wang Y, Han T, Han X, Liu P (2019) Ensemble-learning-based hardware trojans detection method by detecting the trigger nets. In 2019 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE, pp. 1–5
47. Wolf M, Serpanos D (2017) Safety and security in cyber-physical systems and internet-of-things systems. Proc IEEE 106(1):9–20
48. Xiao K, Forte D, Jin Y, Karri R, Bhunia S, Tehranipoor M (2016) Hardware trojans: Lessons learned after one decade of research. ACM Transactions on Design Automation of Electronic Systems (TODAES) 22(1)6
49. Xie X, Sun Y, Chen H, Ding Y (2017) Hardware trojans classification based on controllability and observability in gate-level netlist. IEICE Electron Expr 14(18):20170682–20170682
50. Yang K, Forte D, Tehranipoor MM (2015) Protecting endpoint devices in iot supply chain. In Proceedings of the IEEE/ACM International Conference on Computer-Aided Design. IEEE Press, pp. 351–356
51. Yang Y, Ye J, Cao Y, Zhang J, Li X, Li H, Hu Y (2020) Survey: Hardware trojan detection for netlist. In 2020 IEEE 29th Asian Test Symposium (ATS). IEEE, pp. 1–6
52. Zhang N, Lv Z, Zhang Y, Li H, Zhang Y, Huang W (2020) Novel design of hardware trojan: A generic approach for defeating testability based detection. In 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom). IEEE, pp. 162–173
53. Zhang J, Yuan F, Xu Q (2014) Detrust: Defeating hardware trust verification with stealthy implicitly-triggered hardware trojans. In Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. ACM, pp. 153–166

Richa Sharma completed her B.E. in Computer Science & Engineering from Maharana Pratap College of Technology, Gwalior, India, in 2011 and completed her MTech. in Computer Science from Banasthali Vidyapith, Jaipur, India, in 2014. Currently, she is a Ph.D. scholar at ABV-IIITM, Gwalior. Her area of interest in research is Hardware security, Hardware Trojan, Machine Learning. She is a student member of IEEE.

G. K. Sharma did his Master's (Electronics & Communication Engineering) and Ph.D. (Electronics & Computer Engineering) from IIT Roorkee in 1981 and 1997, respectively. At present, he has been a Professor at ABV-IIITM, Gwalior, Madhya Pradesh, since July 2000. Previously, he was Professor and Head, Department of Computer Science & Engineering at Thapar University, Punjab, from September 1996 to August 1999. He joined these positions initially on deputation from Central Electronics Engineering Research Institute (CEERI), Pilani. Prof. Sharma also worked at the Institute of Microelectronic Systems, Darmstadt University of Technology, Darmstadt, Germany, under Indo-FRG Scientific & Technical Cooperation Programme for a CSIR - KFA bilateral project "Advanced Research in CAD Tools and VLSI Design". His research interests include Low-Power VLSI Design, Network-on-Chip (NoC) Design and Synthesis. Prof. Sharma is a member of the IEEE and IEEE Computer Society.

Manisha Pattanaik received the Ph.D. degree from the Department of Electronics and Electrical and communication engineering from IIT Kharagpur, India, in 2005. She joined the information and communication technology faculty, ABV-IIITM, Gwalior, Madhya Pradesh, India, in 2007, where she is currently a professor. She is the author or co-author of more than 150 research papers in refereed journals and conferences. Her research interests include Low Power/Low Voltage Electronics, Nanoscale CMOS Device/Circuits/System Co-Design Characterization, Design of Low Power Logic and Memory Leakage Power Reduction, and Ground Bounce Noise Reduction Techniques and reliability aware high performance energy- efficient embedded computing.

V. S. S. Prashant received the B.E. degree in Electrical Engineering from Shri Shankaracharya Group of Institutions, Bhilai, India, in 2018 and has completed his M.Tech. in VLSI \& Embedded systems from ABV-IIITM, Gwalior, India. His area of research interest is VLSI architecture design, Hardware security. Currently, he is working in Infosys Ltd., Pune, India, as Digital Specialist Engineer.