Journal of Electronic Testing (2022) 38:667–682
https://doi.org/10.1007/s10836-022-06035-6

A CatBoost Based Approach to Detect Label Flipping Poisoning Attack in Hardware Trojan Detection Systems

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

Received: 20 July 2022 / Accepted: 28 October 2022 / Published online: 7 December 2022

Abstract
Hardware Trojan (HT) intrusion at different integrated circuit (IC) phases is the most important concern for the semiconductor industries. Recently, machine learning (ML) models have been used to detect HT from the pre-silicon IC phase, which utilizes either structural or SCOAP gate level netlist features. However, the main concern is that an adversary may poison the training dataset by flipping the target labels to malign the ML model training, which further provides an incorrect prediction on the test dataset. Thus, due to the malicious training of ML models, the Trojan-inserted ICs are missed out and can easily perform their malicious activities. Hence, it is of utmost importance to scan the training dataset and identify the poisoned input samples before applying ML models for HT detection. Therefore, this paper proposes a new technique that first identifies the poisoned training samples, which consist of SCOAP features, and then detects HTs from the unseen gate-level netlist. The proposed technique employs a robust ensemble Categorical Boosting (CatBoost) model, which avoids the problem of target leakage by using the concept of ordered boosting. Further, a label flipping poisoning attack based on a stochastic hill-climbing search is proposed, which flips the labels of the handful of samples that maximizes the validation dataset loss by deteriorating the model performance. Moreover, a defense method is proposed which utilizes CatBoost object importance and k-nearest neighbor to detect malicious training samples and restore their original labels. Finally, the CatBoost model is trained on the clean dataset to detect the HT nets from the unseen gate-level netlist accurately. Experimental results shows that the proposed attack method increases the on-an-average loss up to 58% and 54% on Trust-Hub and DeTrust benchmarks. Whereas the proposed defense method accurately identifies the poisoned input labels from the training dataset with on-an-average 99% accuracy on these benchmarks.

Keywords Hardware Trojan · Label Flipping Poisoning Attack · SCOAP features · Machine Learning · CatBoost

1 Introduction

Due, to the emergence of the Internet of Things (IoT) and Artificial Intelligence (AI) applications, the demand for integrated circuits (ICs) are increased fourfold. This led semiconductor industries to outsource different IC phases to untrusted parties, which relinquishes the control that IC firms had on IC designing & manufacturing [4, 41]. This global distribution of IC phases to external parties opened the gates for an adversary for hardware Trojan (HT) intrusion [3, 52]. The flow of the IC supply chain given by [3, 19] with trusted and untrusted entities are shown in Fig. 1. An adversary present either in the in-house designing team or in the foundry can insert HT by modifying the existing circuit, adding malicious nets in IC, or manipulating the IC lithographic masks. Besides, the System on Chip (SoC) designers integrates many third-party intellectual property (3PIP) cores in the form of soft, firm & hard IPs at RTL-level, gate-level, and GDS-II-level [32]. These untrusted 3PIPs have access to the source codes and IP design files thus, an attacker can easily insert the HT by modifying the circuit functionality at any specification level. Moreover, the involvement of EDA/CAD tools supplied by different vendors may also insert HT into the IC design and degrade IC logic [20, 54]. For example, Pilato et al. [37] present the CAD

Responsible Editor: C. A. Papachristou

✉ Richa Sharma
richa@iiitm.ac.in

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

Manisha Pattanaik
manishapattanaik@iiitm.ac.in

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

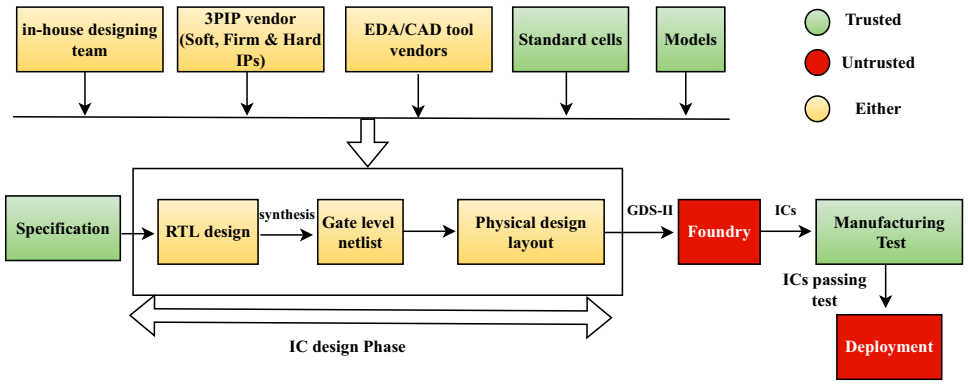

Fig. 1 IC supply chain flow

tool threat by inserting three Trojans in a high-level synthesis tool. Similarly, Basu et al. [2] shows that the CAD tool can launch HT attacks in all phases, from design to test.

HT becomes the most dangerous threat due to its stealthy nature, which, once activated, can lead to catastrophic failures like information leakage, denial of service (DoS), etc., and becomes life-threatening for a myriad of real-life applications [26, 54]. For example, a Syrian radar failed to warn about an air strike because the system chip contain backdoor [3, 54]. Conventional pre-silicon verification, like equivalence checking and post-manufacturing testing, fails to detect HT due to its stealthy nature and the vast number of HT instances designed by an adversary. Moreover, it requires a golden model of the IC for verification which might not always be available when 3PIP is utilized [3, 27]. Besides, researchers [9, 18] have designed a new type of HT by exploiting don't care conditions in such a way that the design with and without Trojans are logically equivalent and can bypass the formal equivalence checking.

Recently, Machine learning (ML) models have been used to detect HT at pre-silicon and post-silicon IC phases which automate the detection process and prove to be more accurate and faster than conventional HT detection approaches [8, 22, 28]. However, to reduce cost and time or due to lack of resources, firms nowadays use pre-trained models or training/testing datasets prepared by external parties, For example, one can train their models on services provided by Google's Cloud Machine Learning Engine, Microsoft's Azure Batch Training and Amazon Virtual Machine [21, 38]. Similarly, pre-trained models can be downloaded from the repositories like Keras pre-trained model library and Berkeley's Caffe model zoo [14]. Besides, data is collected from multiple sources or prepared by third parties and labeled manually by them, For example, Amazon Mechanical Turk, Recommender systems, etc. [55]. The outsourcing of model training, use of pre-trained models or periodic retraining of models, and use of training/testing datasets manually labeled by third parties gives ample chances to an adversary to implant various attacks [11, 55].

The ML pipeline, along with possible attacks in different phases, is shown in Fig. 2. An adversary can perform the poisoning and evasion attacks during the training and testing phase of ML models by perturbing the samples in training or testing datasets with adversarial noise [1, 40]. However, the main aim of the poisoning attack is to mislead the learning process of ML models by poisoning the training dataset, which either increases the prediction loss or is used to trigger the hidden Trojans [48]. An attacker, if present in the dataset preparation and labeling phase, can easily manipulate the labels of the samples or insert additional malicious samples (Backdoor attack) or use the genuine samples as a trigger to activate the Trojan hidden in the design of ML model (Trojan attack). For example, Gu et al. [14] poisoned the pre-trained model in such a way

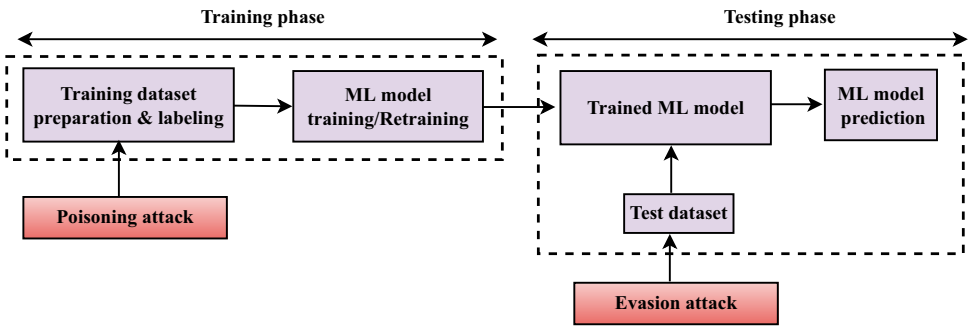

Fig. 2 Machine learning pipeline

that a street sign classifier learns to recognize stop signs as speed limits. In contrast, evasion attacks exploit the vulnerabilities of the ML models by putting the adversarial examples in the test dataset, which fool the model during prediction [29]. For example, Goodfellow et al. [13] create adversarial examples by performing small perturbations to original images of the panda which are misclassified as gibbon by the classifier. However, this paper mainly focuses on detecting two types of attacks, the first one is a poisoning attack, where an attacker maligns the training dataset. Second, HT attack where an attacker alters the gate-level net-list of the IC.

Existing ML-based HT detection techniques in the pre-silicon IC designing phase only detect HT nets from gate-level net-list by utilizing either SCOAP or structural features. Salmani [43] use SCOAP features and apply k-means clustering, whereas Hasegawa et al. [16] use structural features and utilize a random forest classifier for Trojan detection. Kok et al. [23] use bagged trees and combines both structural and SCOAP features for detection. However, none of them paid attention on the adversarial dataset problem. Recently, Nozawa et al. [33] shed some light on this problem and demonstrated evasion attacks by designing various adversarial examples to malign the Neural network (NN) detection process. In contrast, Clements and Lao [7] insert Trojan in the design of NN by maliciously adding the extra circuit in the NN design, which is triggered by a specific input and then misclassified that input into particular class. Similarly, Liu et al. [30] provide three defenses to identify the malicious input images hidden in the training dataset. However, they specifically create the adversarial examples for NN, which only affect the training or prediction process of NN. Besides, the defenses provided by [30] are time-consuming and require the model to be reconfigurable. Moreover, the field of adversarial machine learning is new for the hardware security area, and an adversary can perform several other attacks. One such attack is label flipping poisoning attack [38] in which an adversary controlled the labels of the small set of training samples and flipped them to malign the training process. If such malicious ML models are deployed in the IoT /AI applications, then it causes catastrophic failures because the motive of the adversary now becomes twofold. First, the poisonous training dataset helps to evade the Trojan inserted IC during the detection process, and second those missed Trojan IC's payload then easily perform their hidden malicious functions after triggering. Thus, in addition to HT attacks, adversarial attacks on the ML model also emerge as a major threat that severely affects the prediction process by classifying Trojan-inserted IC's as Trojan-free or vice versa, thus causing a loss of trust in the ML-based HT detection process. Hence, it is necessary to detect these adversarial samples from training/testing datasets before utilizing the ML models for HT detection.

Therefore, in order to secure the training dataset and for accurate Trojan detection, a new Categorical Boosting (CatBoost) model and SCOAP features-based technique is proposed in this paper, which accurately detect the label flipping poisoning attack and HT attack. It first scans the training dataset to identify the malicious training samples, and then HTs are detected from unseen gate-level net-list. The proposed technique employs a powerful CatBoost model, which uses an efficient ordered boosting to deal with target leakage, which is intrinsically present in all gradient boosting-based models. Further, a label flipping poisoning attack is proposed, which utilizes Stochastic hill climbing to identify the optimal training dataset from which labels of a few samples are flipped. Besides, a new defense method is proposed against the label flipping poisoning attack, which utilizes the CatBoost object importance to detect the poisoned samples from the malicious training set, and then the k-nearest neighbor is applied on the captured poisoned samples to relabel them back to their original labels. The major contributions of this paper are as follows:

1. A CatBoost model and SCOAP features-based approach are proposed, which detect poisonous training samples and HT accurately from gate-level net-list.
2. A label flipping poisoning attack is proposed, which flips the labels of the training dataset using Stochastic hill climbing.
3. A new defense method is proposed, which utilizes the CatBoost object importance and k-nearest neighbor to identify the malicious training samples.
4. Experimental evaluation on Trust-Hub and DeTrust benchmarks shows the efficacy of the proposed attack and defense methods.

The rest of the paper is organized as follows: Section II gives a literature review analysis of existing techniques. Section III explains the background, which includes CatBoost model-based learning and SCOAP features. The proposed approach, which includes new label flipping poisoning attack and defense method, is presented in Section IV. Experimental results and comparative analysis is presented in Section V. Finally, Section VI concludes the paper.

2 Literature Review: Analysis

The researchers have proposed several ML based HT detection techniques at the IC pre-silicon phase, the brief overview is provided in [19, 22, 27, 56]. Hasegawa et al. [15] initially extracted the five structural features and trained the SVM classifier to detect the HT nets. Further, 51 structural features are extracted by [16] to improve performance, and 11 best features are identified using a random forest classifier

for detection. They utilize the same 11 features in [17] and NN is trained for detection. Later, Wang et al. extracted the trigger net features and trained the XGBoost model separately to detect combinational and sequential Trojans [49]. Similarly, Kurihara et al. [25] extracted 25 structural trigger features, combined them with 11 features, and trained a random forest classifier to detect HT nets. In contrast, Salmani [43] use SCOAP features and apply k-means clustering to identify HT nets. Similarly, Xie et al. [53] use SCOAP values, extract distance as features using k-means and combine it with primitive features to perform detection using an SVM classifier. Later Kok et al. [24] extracted SCOAP features of both combinational and sequential circuits and trained multiple classifiers for detection. Our previous work [46] uses combinational and sequential SCOAP features and selects the best feature set using a new feature selection method for accurate detection using class weighted XGBoost model. Later Mondal et al. [31] combines transitional probability and combinational controllability to identify HT using k-means clustering. However, all these approaches only focus on HT detection and ignore malicious dataset problem. Therefore, different types of adversarial attacks and defense methods are discussed next.

A brief taxonomy of adversarial attacks and defense proposed in several domains are presented in [11, 21, 48, 55]. Clements et al. [7] insert HT by perturbing the weights of hidden layer neurons in hardware implementation of the well-trained NN, which classify the chosen input trigger into a specific class. However, the major limitation of this attack is model dependency, and the adversary must have access to the NN parameters in order to perform the attack. In contrast, Liu et al. [30] proposed three defense techniques to detect the input triggers in the training dataset, which are used to activate the neural Trojans hidden in the NN design. The input anomaly detection approach separately trains SVM and decision tree classifiers 'm' classes times to identify whether the sample belongs to any one of the 'm' classes or not otherwise, it is determined as an anomaly. However, this method is time-consuming and complex because the number of classifiers trained is proportional to the number of target classes. The retraining method retrains the NN multiple times with genuine data so that the model weights are overwritten, which makes the Trojan triggers inefficient. However, it requires the model to be reconfigurable and needs its intrinsic weight values. Whereas the input preprocessing approach checks the training dataset first to detect malicious input triggers. The reconstructed images generated by the auto-encoder are compared with the input images per class. The images which show higher deviation than the reconstructed images are detected as malicious triggers. However, another NN is used to detect triggers which makes the overall technique very complex and time-consuming.

Recently, Nozawa et al. [33] proposed an evasion attack where several adversarial examples are generated for NN to avoid hardware Trojan detection by replacing HT circuits with logically equivalent circuits. A Trojan-net concealment degree and modification evaluating value is proposed to generate adversarial examples, which eventually causes misclassification. However, to generate such malicious examples, an attacker requires to know about the model and all its parameters and gate-level netlists. Moreover, there is a possibility that during logic synthesis optimization, these modifications are removed. Finally, the above-discussed approaches fail to generalize to other ML models because they only focus on the NN and perform the white-box attack, which is not possible in a real life scenario. Besides, none of the research has been carried out on label flipping poisoning attacks in the hardware Trojan field. However, the research on label flipping poisoning attacks has been carried out in several other domains, which are discussed next. An optimization framework is proposed by Xiao et al. [51] in which the labels of the near-optimal samples are flipped based on some given budget which maximally degrades the SVM classifier performance. However, the attack is white-box and specifically made for SVM because the flipped labels are mainly those samples that are near the hyper-plane boundary of the classifier. Moreover, the budget is entirely dependent on the distribution of the datasets.

Similarly, Xiao et al. [50] flip the bounded number of training samples chosen by the bounded distribution to maximize the SVM classification error. Two types of heuristic-based attacks are proposed, the first one generates different sets using gradient ascent and chooses one which provides the best value. The second one generates correlated subsets of label flips using a greedy best-first search. The subset which maximizes the empirical error is chosen. Both these attacks are the white-box one where the attacker has perfect knowledge of the model used and its parameters. However, these attacks cannot generalize to other models due to model dependency. Moreover, the high computational complexities of the attacks make it infeasible to be performed on larger datasets. Besides, the correlated cluster method requires a lot of computational time. Paudice et al. [34] proposed a heuristic-based attack in which samples to be flipped are chosen greedily during every iteration, which maximizes the validation loss. Further, they propose the k-nearest neighbor-based defense method, which considers the samples far from the decision boundary as malicious and replaces their labels with the most common neighboring sample labels that satisfy a predefined threshold. However, this approach is time-consuming because it flips all the samples and computes validation error to find the set of poisoned samples and similarly identify the nearest neighbor of all samples for detection. Moreover, the setting of the correct threshold is necessary otherwise, genuine samples may be relabeled as

malicious because the approach fails to distinguish between the overlapping areas of two classes.

Taheri et al. [47] proposed a label flipping attack against the deep CNN model on malware detection systems in which the silhouette score is computed for each training sample after applying k-means clustering and labels of samples having scores less than zero are flipped. Further, a semi-supervised and clustering-based defense is proposed in which label propagation and label spreading algorithms are applied to detect poisoned samples. Further, voting is performed between the labels predicted by these two semi-supervised models, CNN and the poisoned label. Finally, malicious samples are relabeled based on the majority of votes given to a particular label of that sample. In contrast, clustering-based defense computes four cluster measures, and the difference between these measures is computed during every iteration. The obtained difference is compared to the specified threshold, and accordingly, the label of the malicious samples is restored. However, setting a correct threshold value is a cumbersome task that affects the attack performance. Moreover, the defense methods are time-consuming and complex because of the use of multiple classifiers or cluster measures. Besides, semi-supervised learning may provide unstable predictions because of the lack of ability to correct its mistakes.

Zhang et al. [57] proposed a two-label flipping attack against the Naive Bayes classifier on spam filtering systems. In the first attack, the entropy of each attribute present in the training dataset is computed, and its weight is calculated accordingly. Further, the score of each sample labeled as spam is computed using obtained weights, and finally, the labels of the first M samples having smaller weights are flipped. Similarly, k-mediods clustering is utilized in the second attack, which creates two clusters having spam and non-spam samples. Further, the distance is computed between each sample of the spam cluster and the center point of the non-spam cluster, and eventually, the labels of the first 'M' samples, which possess small distances, are flipped. However, samples of only spam classes are flipped, which are already in the minority, thus create severe class imbalance. Moreover, randomly flipping of top 'N' spam samples without any heuristic does not create a powerful attack. Besides, k-mediods are unstable and may give different results in different iterations, which affect the attack performance. A semi-supervised learning-based defense method is proposed by Cheng et al. [6] in which the AdaBoost model iteratively identifies the set of samples having larger weights as label flipped samples. Further, the obtained set is fed to the semi-supervised learning algorithm, which relabels the malicious samples to their original correct labels. However, some genuine samples may possess larger weights which are also relabeled as malicious, which eventually causes misclassification.

3 Background

3.1 CatBoost Model Based Learning

CatBoost [39] is an ensemble gradient boosting based technique which uses an efficient ordered boosting and ordered target statistics algorithms to avoid target leakage which is inherently present in all gradient boosting based techniques [5, 10]. Suppose we have a training dataset $TrD = \sum_{n=1}^{N}(x_n, y_n)$ which contains 'N' input samples x_n and corresponding output values y_n. The standard gradient boosting algorithm [10] sequentially trains several j base learners i.e. function BL^j on the pseudo-residuals generated by the previous learner $BL^{(j-1)}$ using gradient descent to optimize the loss function $L(y_n, BL^j(x_n))$ which can be given as:

$$BL^j(x_n) = BL^{(j-1)}(x_n) + \alpha h^j(x_n) \tag{1}$$

where α is a learning rate and function $h^j(x_n)$ minimizes the expected loss function which can be written as:

$$\begin{aligned} h^j(x_n) &= \underset{h^j \epsilon DT}{\operatorname{argmin}} L(y_n, BL^j(x_n)) \\ &\equiv \underset{h^j \epsilon DT}{\operatorname{argmin}} L(y_n, BL^{(j-1)}(x_n) + h^j(x_n)) \end{aligned} \tag{2}$$

Now, value of h^j is chosen in the direction of negative gradient i.e the gradient of L with respect to $BL^{(j-1)}$ is decreasing which can be approximated for all the samples as:

$$h^j(x_n) = \underset{h^j \epsilon DT}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} \left(\frac{\partial L(y_n, BL^{(j-1)}(x_n))}{\partial BL^{(j-1)}(x_n)} - h^j(x_n)\right)^2 \tag{3}$$

The main concern pointed out here by [39] is target leakage because the categorical features and the gradients estimated during each iteration rely on the target values y_n of the training data samples x_n. This leakage occurs due to the reuse of the same training data for each base learner, which leads to shifting (difference) in the distribution of gradients of $F(x_n|x)$ for training sample x_n and $F(x|x)$ for test sample x. This conditional and prediction shifting biased the base learner prediction, which severely affects the generalization capability of the trained model and causes overfitting.

Therefore, CatBoost performs ordered gradient boosting in which random permutations of training samples are performed. The main aim is to remove the gradient bias (target leakage) generated by training and testing data shift by using the different permuted training datasets, and only prior data samples are used for current sample prediction. During learning, instead of using the same training dataset for all the base classifiers, CatBoost generates independent $(p + 1)$ random permutations $\sigma_0, \sigma_1, ...\sigma_p$ of the training set where 'p' permutations σ_1 to σ_p evaluate the internal node split of the tree and σ_0 compute the leaf values of the generated trees. During each training iteration 'it', a random permuted

dataset σ_r is chosen, and the ordered target statistics are computed for categorical features. Further, for each permutation, '$v = (n-1)$' supporting models are maintained i.e. $M_{(r,v)}$ in such a way that model $M_{(r,v)}(x_n)$ is learned from the first 'v' samples in the permuted dataset σ_r for sample x_n. The residual (res_n) for x_n is computed as shown below:

$$res_n = y_n - M_v(x_n) \tag{4}$$

Instead of using asymmetric decision trees, CatBoost uses symmetric oblivious decision trees as base learners in which the same splitting criterion is used across the entire level, which makes them balanced and less prone to overfitting. Since they are full binary trees, they require the minimum number of comparisons to reach the leaf nodes, which significantly speeds up the execution. Now for sample x_n, the gradient of the loss with respect to supporting model $M_{(r,v)}(x_n)$ is computed, which is relied on the preceding training samples present in σ_r and it is given as:

$$grad_{(r,v)}\ (x_n) = \frac{\partial L(y_n, M_{(r,v)}(x_n))}{\partial M_{(r,v)}(x_n)} \tag{5}$$

Further, the leaf values for the x_n sample during splitting are computed by taking the average of the gradients $grad_{(r,v)}(x_n)$ of the prior samples belonging to the same leaf x_n sample lying on. Now, once the tree T_{it} is constructed, the same tree structure is utilized by the other supporting models of the different permuted datasets in order to reduce complexity. Once all the trees are built, the leaf values of sample x_n for the final model are computed on σ_0 using the standard gradient boosting procedure. The whole process is repeated for each permuted training dataset, and the sub-models are trained until the loss becomes minimized and all the parameters are trained correctly in order to build the most robust oblivious decision tree, which accurately identifies the Trojan free and Trojan inserted nets from the testing dataset.

3.2 SCOAP Gate-Level Net-List Features

It has been analyzed by [24, 43] that in order to evade detection, an attacker inserts the functional HT at a low switching activity area to avoid frequent activation. Therefore, nets that are difficult to control and observe are expected to be used as HT triggers and payload. Those nets in a circuit possess large controllability and observability values, thus avoiding frequent impact on circuit design and remain hidden during testing. It can be seen from Table 1 that the average SCOAP values of Trojan-inserted nets of the Trust-Hub RS232-T1600 circuit are higher than its Trojan-free nets. Thus, six SCOAP values i.e. combinational/sequential-0 controllability ($CC0$, $SC0$), combinational/sequential-1 controllability ($CC1$, $SC1$), combinational/sequential observability (CO, SO) of both combinational and sequential circuits are extracted using SCOAP method [12] to perform detection. However, Trust-Hub circuits s35932-T200 and s38584-T100 comparatively possess lower SCOAP values than other Trojan-inserted Trust-Hub circuits, but it has been observed by [43] that these Trojans are frequently activated by applying random test patterns. They showed that the HT present in the s35932-T200 benchmark is activated 42 times by applying random test patterns only for 4261 test clock cycles. Whereas the SCOAP values of benchmark s38584-T100 are even less than Trojan-free nets, and the HT is activated 21 times by merely applying 3286 test cycles. Finally, [43] analyzed that the SCOAP values of Trojan-inserted nets should not be small or close to the values of Trojan-free circuits, otherwise it experiences switching activity, frequently interferes with the normal circuit functionality, and are detected during circuit testing.

Table 1 Average SCOAP Values for RS232-T1600 circuit [46]

Net Type	(CC0, CC1)	CO	(SC0, SC1)	SO
Genuine net	(13.69, 18.36)	240.8	(1.208, 1.63)	166.1
Trojan net	(290.6, 40.41)	2217.8	(29, 3.74)	1857

4 Proposed CatBoost Based Approach to Detect Label Flipping Poisoning Attack

This section explains the proposed approach, which includes the proposed Label flipping poisoning attack and CatBoost-based defense method.

4.1 Threat Model & Problem Statement

This sub-section discusses the Threat model followed by the problem statement.

4.1.1 Threat Model

In this paper, we are aiming to detect two types of attacks, Label flipping poisoning, and HT attacks. Thus, we discuss two threat models which provide the scenario of how an adversary can perform the above attacks. In order to fasten up the process and save money, the 3PIP cores or EDA/CAD tools are incorporated in the IC designing. However, untrusted 3PIP cores or tools or an adversary present in the in-house design team has full access to the netlist codes and may intentionally insert the HT in it. We mainly focus on determining the combinational & sequential functional Trojans, which are inserted by the attacker at the low switching activity area of the circuit. Similarly,

the involvement of ML models in HT detection opens up new avenues for an adversary to perform different types of data-based attacks (poisoning or evasion). An adversary can be present in the dataset preparation & labeling team may insert the malicious samples in the training or testing dataset, which malign the training/testing process of ML models, thus increases the misclassification rate during prediction. However, we focused our attention on label-flipping poisoning attacks where we assume that the adversary is present in the training dataset preparation and labeling team. The prominent goal of the label-flipping poisoning attack is to malign the training of the ML model by flipping a small number of samples in the training dataset so that it provides the wrong prediction during testing. Moreover, it has also been assumed that the attack is performed under a black box scenario, where an attacker has no knowledge about the ML model used in HT detection.

4.1.2 Problem Statement

The problem statement is stated as follows: Suppose the training dataset contains 'm' and 'n' samples of Trojan free (TF), and Trojan inserted (TI) nets with labels. An attacker who has access to this training dataset will modify the labels of 'p' samples to malign the training process of any ML model-based HT detection technique so that it will give erroneous predictions during testing. The problem is to accurately identify the 'p' poisoned samples present in the malicious training dataset so that the ML model will correctly be trained on a clean dataset and finally predict the correct classes of TI/TF nets present in the test dataset. However, in this paper, we have chosen the CatBoost ML model to identify the malicious training samples and HT nets.

4.2 Proposed Approach

The proposed approach is shown in Fig. 3, it utilizes the CatBoost model to identify the poisonous training samples and malicious HT nets from the gate-level netlist. Initially, the SCOAP features of the circuits are extracted and stored with labels (More details are provided in Section V(A)). Afterwards, the proposed attack method poisoned the training dataset by flipping the labels of the fraction of samples which makes the model training erroneous by maximizing the validation loss on the unseen validation dataset. The maliciously trained ML model then eventually provides incorrect predictions on the test dataset. Further, the proposed defense method identifies the malicious samples present in the poisoned training dataset and restores them back to their original labels by minimizing the validation loss. Finally, the ML model is trained on the clean dataset for accurate detection of Trojan nets. The overall technique is presented in the following subsections, where the proposed label flipping poisoning attack is discussed first, followed by the new defense method.

4.2.1 Proposed Label Flipping Poisoning Attack

To perform the attack, an adversary analyzes the training dataset to perform smart flipping, which increases the loss on the unseen dataset most. Since our training dataset (*TrD*) contains 'N' samples with labels, the main motive of the attacker is to identify the fraction of 'p' samples in the *TrD* for flipping. It chooses the samples in such a way that the poisoned dataset (*PTrD*) maximizes the validation loss (*Vl*) computed on the unseen validation dataset (*VaD*) containing 'K' samples while keeping the training loss (*Trl*) minimized. It can be seen as a bi-level optimization problem given as:

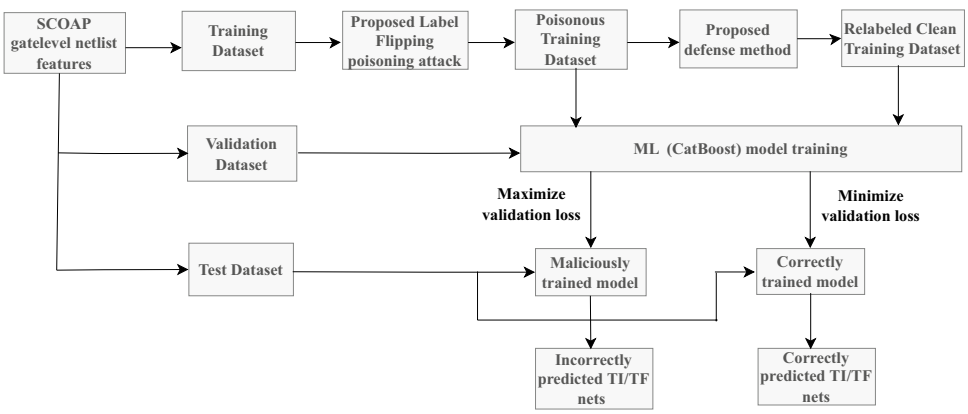

Fig. 3 Proposed approach to detect label flipping attack

$$\underset{p}{\text{argmax}} \sum_{i=1}^{K} (Vl(y_i, BL_{PTrD}(x_i)) \tag{6}$$

such that

$$BL_{PTrD} \equiv \underset{BL}{\text{argmin}} \sum_{j=1}^{N} (Trl(y'_j, BL(x_j)) \tag{7}$$

However, identifying the best samples to flip is of utmost importance because the stealthiness of the attack is entirely dependent on it. Existing techniques [34, 47, 57] greedily select the samples, compute each sample entropy and apply clustering for flipping which either consume a lot of time or missed out some good samples. In contrast, the proposed attack method utilizes the heuristic stochastic hill climbing (SHC) search algorithm [42] to identify the best '*p*' number of samples for flipping. The main aim of SHC is to explore the search space locally until the global optimum subset of *TrD* is identified, which provides the highest accuracy on *VaD*. Initially, a random optimal subset is chosen by SHC from *TrD* on which the validation accuracy is computed. Afterwards, a step is taken within the search space to generate the new modified subset in nearby proximity, which searches for better neighborhood points. However, step size needs to be set properly, and it should not be very large/small, otherwise, it either misses the local optima or get trapped in it. SHC randomly chooses among several neighboring solutions present uphill, with some selection probability which varies according to the steepness of the steps.

Therefore, in the proposed attack, SHC modifies the existing subset to achieve the new one by randomly mutating the inclusion/exclusion of samples from the set. Each individual sample present in the subset is mutated based on a probability that indicates the step size in the search space. Once the new subset is generated, performance is evaluated on the *VaD*, and if it is better than the previously obtained validation accuracy, then it is set as the best subset. Similarly, the mutated version of the new subset is created iteratively for the number of iterations, and those solutions are considered the best solution that maximizes the performance of the *VaD* most. The obtained optimal solution contains the best samples in which an adversary is interested in label flipping. However, only a small amount of samples need to be perturbed thus, 20% of samples are randomly selected from the optimal subset, and their labels have been flipped. Afterward, the loss is computed on the *VaD*, and this procedure is repeated up to a fixed number of iterations. Finally, the set of samples whose labels, when flipped, provide the maximum *Vl* on the *VaD* will be chosen as the final poisonous *PTrD*.

Algorithm 1 : Proposed attack algorithm

1: **Input**: Training Dataset ($TrD = (X_{tr}, y_{tr})$), Validation Dataset ($VaD = (X_{val}, y_{val})$)
2: **Output**: Poisonous Training Dataset ($PTrD = (X_{tr}, y'_{tr})$);
3: Initialize $hcitr = 5$, $pmut = 0.5$, $rsitr = 10$;
4: Generate the initial solution (sol) and store the corresponding samples in $TrD_n = (X_{trn}, y_{trn})$.
5: Train the CatBoost model using TrD_n.
6: Compute validation accuracy on VaD and store in $pacc$.
7: **for** ($i = 1$ to $hcitr$) **do** ▷ *Hill climbing*
8: **for** ($j = 1$ in range $len(sol)$) **do** ▷ *Mutation*
9: **if** ($rand() < pmut$) **then**
10: $sol[j] = !sol[j]$
11: **end if**
12: **end for**
13: Update the TrD_n with newly generated solution.
14: Train the CatBoost model using new TrD_n and store the new validation accuracy in $nacc$.
15: **if** ($nacc > pacc$) **then**
16: Store the new solution and performance scores in sol and $pacc$.
17: **end if**
18: **end for**
19: **for** ($k = 1$ to $rsitr$) **do**
20: $poid = rand(len(sol)) * 0.20$
21: $y_{tr}[poid] = 1 - y_{tr}[poid]$ ▷ *labels are flipped*
22: Train the model on $PTrD$ and compute the Vl.
23: Save the performance measures along with $PTrD$ in dictionary ($dict$).
24: **end for**
25: Fetch the $PTrD$ from $dict$ which provide max Vl .
26: **Return:** $PTrD$;

4.2.2 Proposed Label Flipping Poisoning Attack Algorithm

The proposed Algorithm 1 takes the training (*TrD*) and validation (*VaD*) dataset as an input and provides a poisoned training dataset (*PTrD*) as an output. The initial solution (*sol*) is generated first and CatBoost model is trained on TrD_n which computes validation accuracy on *VaD* that is stored in *pacc*. Now during every hill climbing iteration *hcitr* = 5 from line no 7 to 18, the *sol* is modified by randomly mutating the samples with probability *pmut* = 0.5 and the validation accuracy is computed using the model trained on new TrD_n and stored in *nacc*. Further, the obtained *nacc* is compared with *pacc* iteratively, and the best solution which provides the highest validation accuracy is stored. Afterwards, 20% of samples are chosen from the *sol* during every iteration *rsitr* = 10, and their labels have been flipped. Further, the model has been trained on *PTrD* at every iteration and *Vl* is computed.

Finally, the set which maximizes the *Vl* most is chosen as the final *PTrD*.

4.2.3 Proposed CatBoost and K-Nearest Neighbor Based Defense

The main aim of the proposed defense method is to identify the poisoned samples from *PTrD* accurately so that along with *Trl*, the trained CatBoost ML model also minimizes the *Vl*. Thus, to detect those poisoned samples, the proposed defense method utilizes the CatBoost object importance method, which prioritizes every training sample based on their performance during prediction and pointed out the least important samples. The object importance method identifies the effect of every *PTrD* sample by computing the *Vl* on *VaD*. The samples which tend to increase the *Vl* have higher chances of being malicious, thus, these samples are captured and marked as poisonous. However, it has been observed that the object importance method may capture some normal samples which are not malicious. Therefore, for further refining and to improve the performance, k-nearest neighbor (k-NN) [36] is applied on the samples identified by the object importance method. It identifies the '$n \geq 3$' neighbors of each sample marked as poisonous and identifies whether neighbors and the pointed samples belong to the same or different group, i.e., whose three or more neighbors belong to the same class. Afterwards, those samples are relabeled whose class label is not the same as its maximum neighbors class label, and performance is evaluated on the *VaD*. In this way, the proposed method accurately detects the poisoned samples from *PTrD*.

Algorithm 2 : Proposed defense algorithm

1: **Input** : $PTrD$, VaD;
2: **Output**: Clean Training Dataset (CtD);
3: Train the CatBoost model on $PTrD$.
4: Compute the object importance on $PTrD$.
5: Store the flagged malicious samples returned by CatBoost in array OT.
6: Call k-NN model instance and predict the final class of OT samples and stored them in array $pred$.
7: **for** $i = 1$ to $len(y'_{tr}[OT])$ **do**
8: **if** $(y'_{tr}[i]! = pred[i])$ **then**
9: Store the malicious samples in array mal.
10: **end if**
11: **end for**
12: $y'_{tr}[mal] = 1 - y'_{tr}[mal]$
13: Clean dataset (CtD) is obtained.
14: Train the model on the CtD and evaluate the Vl.
15: **Return:** CtD;

4.2.4 Proposed Defense Algorithm

The proposed Algorithm 2 takes the poisonous training dataset (*PTrD*) and validation dataset (*VaD*) as an input and provides a clean training dataset (*CtD*) as an output. Initially, the CatBoost model is trained on *PTrD*, and object importance is computed on *VaD*. Further, the malicious flagged samples returned by the object importance method are stored in the array *OT* as given in line no 5. Afterwards, the k-NN model instance is called, which predicts the classes of samples present in *OT* and stored all the predictions in the array *pred*. If the condition given in line no.8 is satisfied, then those samples are stored in the array *mal*, and their labels have been re-labeled. Finally, the model is trained on the obtained re-labeled *CtD*, and performance is computed on the *VaD*.

5 Experimental Results and Analysis

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

5.1 Dataset Description and Evaluation Measures

We create the first dataset using 16 Trust-Hub benchmarks [44] that contains combinational or sequential functional Trojans. Besides, to create the second dataset, we use DeTrust benchmarks [58]. However, to create DeTrust benchmarks, we perform the modifications by inserting the single flip-flop at each gate output of the trigger circuit as suggested by [43] in 11 Trust-Hub benchmarks. Further, a python code is written which converts the Verilog netlist of the above-discussed benchmarks into bench format, and then six SCOAP features are extracted using the Testability measurement tool [45]. Afterwards, obtained features are stored, and each net is labeled as TI/TF, and preprocessing operations are applied for further refining. Now, we divide the datasets into training (80%), validation (10%) and testing (10%) datasets. Moreover, the CatBoost model contains several hyper-parameters which need to be set properly for accurate Trojan detection. Therefore, grid search is applied to find the optimal values of the following parameters, *iterations* = 2000, *learning_rate* = 0.01, regularization parameter (*l2_leaf_reg* = 3), *depth* = 4, *max_leaves* = 31. Besides, other parameters are set as follows, *boosting_type* = *Ordered*, *grow_policy* = *Symmetric* − *Tree* to avoid target leakage and *auto_class_weights* = *Balanced* to avoid class imbalance. Finally, the proposed approach

Table 2 Comparative proposed attack results on Original and Poisoned Trust-Hub benchmarks(%)

Trust-Hub Benchmarks	Before attack							After proposed attack						
	Loss	Accuracy	Recall	ROC-AUC	TNR	FPR	FNR	Loss	Accuracy	Recall	ROC- AUC	TNR	FPR	FNR
RS232-T1000	0.16	100	100	100	100	0	0	38.65	87.34	33.33	23	87.09	12.9	66.66
RS232-T1100	0.4469	100	100	100	100	0	0	45.17	80.12	20	90.54	81.08	18.91	80
RS232-T1200	0.56	99.4	100	99.69	99.39	0.6	0	61.08	66.66	25	46.34	67.68	32.3	75
RS232-T1300	0.2	100	100	100	100	0	0	61	81.09	28.57	53	83.43	16.5	71.42
RS232-T1400	1.38	99.38	91.6	99.42	100	0	8.33	58.6039	76.23	80	78.08	97.59	23.82	20
RS232-T1500	0.24	100	100	100	100	0	0	37.92	77.77	66	55	77.98	22	33
RS232-T1600	1.2	99.39	85.71	85.3	100	0	14.28	50.6088	67.9	66	67.29	67.92	32	33
s38417-T100	0.0636	100	100	100	100	0	0	67.639	64.06	66.6	65.363	64.06	35.93	33
s38417-T200	0.167	99.97	100	99.987	99.97	0.025	0	67.69	62.087	75	68.53	62.07	37.92	25
s38417-T300	0.1612	99.96	81.81	84	100	0	18.18	67.43	51.31	50	50.65	51.31	48.68	50
s35932-T100	0.03	100	100	100	100	0	0	68.22	56.91	75	78.447	56.89	43	25
s35932-T200	0.198	100	100	100	100	0	0	47.25	69.23	50	59.868	69.7	30.2	50
s35932-T300	0.446	99.8	100	99.8	99.6	0.4	0	61.8391	54.5454	50	52.285	54.57	45.42	0.5
s38584-T100	3.13	99.98	100	99.344	99.98	0.018	0	66.3	62.394	40	51.2186	62.43	37.56	60
s38584-T200	0.183	100	100	100	100	0	0	68.3	61.47	50	55.737	61.47	38.52	50
s38584-T300	0.022	100	100	100	100	0	0	68.408	54.4647	33	43.9	54.4	45.5	66
Average	0.661731	99.8675	97.445	97.97131	99.93375	0.065188	2.549375	58.5068	67.09882	50.53125	58.70304	68.72938	32.5725	46.16125

(preprocessing + proposed attack method + proposed defense method+ CatBoost model) is implemented in python using scikit learn [35] library.

Besides, to measure the performance of our proposed approach, the following evaluation metrics are used, Accuracy $= \frac{(TP+TN)}{(TP+FP+TN+FN)}$, which represents the correctly predicted TI or TF samples out of total samples, Recall (True positive rate) $= \frac{TP}{(TP+FN)}$ and True negative rate (TNR) $= \frac{TN}{(TN+FP)}$, gives the percentage of TI or TF samples correctly predicted as TI or TF. Similarly, False positive rate (FPR) $= \frac{FP}{(TN+FP)}$ and False negative rate (FNR) $= \frac{FN}{(TN+FP)}$ tells how many TF or TI samples are incorrectly predicted as TI or TF. Further, to identify model bias, another metric, Receiver operating characteristics and Area under curve (ROC-AUC) score, is used, which tells about the model separability, i.e., how the model separates the samples into appropriate TI or TF classes.

5.2 Simulation Results and Analysis

5.2.1 Proposed Attack Results on Trust-Hub Benchmarks

The comparative before and after attack results on 16 Trust-Hub benchmarks, i.e., on the test dataset, are shown in Table 2. It can be analyzed that the CatBoost model provides an almost correct prediction on the test dataset when trained on a non-poisoned training dataset, i.e., *TrD* by achieving on-an-average 0.661% loss and 99.86% accuracy respectively. Whereas the proposed flipping attack deteriorates the performance on all 16 benchmarks, it can be observed that the loss on the *RS*232 benchmark series lies between 0.16% to 1.38% which is significantly increased between 37.92% to 61.08% by the proposed attack that shows the severity of the misclassifications on test dataset performed by the model in the presence of *PTrD*. Similarly, the proposed attack decreases the accuracy, recall, and TNR up to 66%, 20%, and 67% on *RS*232 series benchmarks, which shows that model is wrongly predicted both TI and TF instances. Besides, the area covered by the ROC-AUC score was reduced to 23%, and increased FPR and FNR rates also confirmed this. Moreover, the proposed attack method provides the highest loss in *s*38417, *s*35932, and *s*38584 series benchmarks, which ranges up to 68.4% and decreases the accuracy up to 51%. Lastly, the proposed attack method provides on-an-average 58.50% loss, 67.09% accuracy, 32.57% FPR, and 46.16% FNR, which eventually shows that proposed label flipping poisoning attack method successfully malign the learning process of the CatBoost model thus fail to provide the correct prediction on the test dataset.

Besides, the average comparative results of the existing and proposed attack methods are shown in Table 3, which shows that the attacks proposed by [57] only affect the recall (46.31% & 54.2%), FNR (53.68% & 45.79%) and ROC (72.69% & 76.97%) because they only target the labels of positive TI class for flipping. Whereas technique [47] achieves the average TPR of 57.34% and TNR of 86.97%, which shows that though labels of both the classes are flipped but still the severity of the attack is highly dependent on the threshold, which affects the overall attack performance. Similarly, the attack proposed by [34] fails to reduce the model performance and thus achieves average FPR and FNR of 21.14% & 16%. In contrast, the proposed attack method effectively reduces the model performance on the test dataset by achieving the maximum average loss (58.5%), FPR (32.57%), and FNR (46.16%), which is 25.37%, 23.90% and 6.93% higher than the existing techniques. Moreover, the obtained accuracy (67.09%), recall (50.53%), TNR (68.72%), and area covered (58.7%) also shows that the maliciously trained model performs misclassification at a higher rate by predicting TI nets as TF and vice versa.

5.2.2 Proposed Defense Results on Trust-Hub Benchmarks

The proposed defense results on re-labeled clean 16 Trust-Hub benchmarks are shown in Table 4, it can be observed that the proposed method significantly improved the model performance on the test dataset by achieving on an average 1.43% loss, 99.66% accuracy, 0.17% FPR and 10.53% FNR respectively. The obtained results clearly indicate that the proposed defense method identifies almost all the malicious samples from the *PTrD* and re-labeled them correctly, which

Table 3 Comparative proposed attack results on Trust-Hub benchmarks (%)

Metrics	K-mediods based attack [57]	Entropy based attack [57]	Silhoutte clustering based attack [47]	Greedy heuristic based attack [34]	Proposed attack
Loss	15.15	16.63	44.93	54.34	58.5
Accuracy	99.12	99.17	86.59	78.78	67.09
Recall	46.31	54.2	57.34	71.95	50.53
ROC-AUC score	72.69	76.97	72.16	76.14	58.7
TNR	99.73	99.75	86.97	78.838	68.72
FPR	0.2609	0.238	13.01	21.14	32.57
FNR	53.68	45.79	42.65	16	46.16

Table 4 Proposed Defense Result on re-labeled clean Trust-Hub benchmarks (%)

Trust-Hub Benchmarks	Loss	Accuracy	Recall	ROC- AUC	TNR	FPR	FNR
RS232-T1000	5.7	99.23	83.3	85	100	0	16.66
RS232-T1100	1.3	99.37	66.66	83.33	100	0	33.33
RS232-T1200	1.215	98.73	100	99.35	98.7	1.29	0
RS232-T1300	0.29	100	100	100	100	0	0
RS232-T1400	2.39	99.41	91.66	90	100	0	8.33
RS232-T1500	3.1	99.37	83.33	91.5	99.67	0.32	16.66
RS232-T1600	1.58	99.38	90.9	99.23	100	0	9.09
s38417-T100	0.414	100	100	100	100	0	0
s38417-T200	0.1715	99.97	85.7	80	100	0	14.28
s38417-T300	0.1645	99.97	75	77	100	0	25
s35932-T100	0.1914	99.97	75	87.5	100	0	25
s35932-T200	0.414	100	100	100	100	0	0
s35932-T300	0.5179	100	100	100	100	0	0
s38584-T100	3.151	99.32	99.75	99.324	98.89	1.102	0.24
s38584-T200	0.02503	100	100	100	100	0	0
s38584-T300	0.2675	99.9	80	83	99.95	0.049	20
Average	1.430739	99.66375	89.45625	92.20213	99.82563	0.172563	10.53688

eventually makes the model training correct. However, it has been analyzed that the CatBoost detects all the TI and TF nets correctly in five benchmarks, i.e., $RS232-T1300$, $s38417-T100$, $s35932-T200$, $s35932-T300$ and $s38584-T200$. Whereas it provides lower recall between $(66\%-83.33\%)$ and large FNR $(16.66\%-33.33\%)$ in $RS232-T1000, T1100, T1500, s38417-T300, s35932-T100$ and $s38584-T300$ benchmarks respectively which means that few TI nets are misclassified as TF. This happens because some of the malicious samples are still present in the *CtD* of these benchmarks, thus missed out by our method, which affects the prediction. Finally the proposed defense overall detect all the malicious samples, it can be seen from the fact that the difference between average results obtained before attack and after defense is not very large i.e loss (0.77), accuracy (0.2), TNR (0.11), FPR (0.105) respectively.

Besides the average comparative results are shown in Table 5, it can be seen that k-NN-based defense [34] provides lower recall (71.46%) and higher FNR (28.528%) thus fail to detect Trojan nets accurately from the test dataset which shows that k-NN either mislabels the genuine inputs or fail to identify the malicious samples. Whereas AdaBoost [6] & Label-spreading based defenses [47] provides comparable loss (4%), accuracy/TNR lies between $(98\%-99\%)$, achieve FPR (1.9% & 1.25%) but lacks in recall, ROC and FNR. However, AdaBoost-based defense [6] provides 62.15% recall, 37.84% FNR, and covers 80% area because some genuine samples which possess higher weights are wrongly relabeled into a different class. In contrast label spreading based defense [47] provides higher recall (77.55%) and covers more area (88%) than [6] because it relabel the samples based on voting. However, it is interesting to see that though clustering-based defense [47] provides lower loss (2.29%) but still lacks in recall (68.59%), FNR (31.41%) and covers only 84% area which indicates that clustering measures are not very appropriate for poisonous samples detection and fail to detect the TI nets. Finally, our proposed defense method provides on-an-average 89.45% recall and covers 92.2% area which is 19.5% and 7.7% higher than the existing techniques. Further, it also shows that the prediction is almost correct and not biased to any single TI/TF class for most of the benchmarks.

Table 5 Comparative defense results on Trust-Hub benchmarks (%)

Metrics	KNN based defense [34]	Adaboost based defense [6]	Label spreading based defense [47]	Clustering based defense [47]	Proposed Defense method
Loss	5.302	4.58	4.2	2.29	1.43
Accuracy	99	99.285	98.34	99.59	99.66
Recall	71.46	62.15	77.55	68.59	89.45
ROC-AUC score	85	80.98	88	84	92.2
TNR	99.48	99.8	98.74	99.87	99.82
FPR	0.514	1.918	1.25	0.12	0.1725
FNR	28.528	37.84	22.44	31.41	10.53688

5.2.3 HT Detection Analysis: Results Obtained on Non-Poisoned & Relabeled Trust-Hub Benchmarks

In this subsection, we have analyzed the results obtained on the test dataset when the CatBoost model is trained before the attack on *TrD* and after relabeling on *CtD* training datasets with respect to HT detection. It can be analyzed from Table 2 before attack results that the CatBoost model accurately detects all combinational & sequential functional Trojans in 13 benchmarks out of 16. It provides nearly 0 FPR & FNR in $RS232-T1000$, $T1100$, $T1200$, $T1300$ & $T1500$, $s38417-T100$, $T200$, $s35932-T100$, $T200$ & $T300$, $s38584-T100$, $T200$ & $T300$ benchmarks respectively. This clearly shows that the CatBoost model trained on SCOAP feature values effectively identifies the HT from these benchmarks. Though all the TF nets are detected correctly in benchmarks $RS232-T1400$, $T1600$ & $s38417-T300$ but some of the TI nets are misclassified. It may have happened because the weight calculated by the parameter *auto_class_weights* to balance both the classes is not effective in some cases, thus model provides incorrect predictions. This can also be confirmed from the obtained ROC-AUC score of these benchmarks, as it only covers 99.42%, 85.3% & 84% area, which indicates that computed weights are not effective for these benchmarks, and the model prediction is biased towards the TF class. Moreover, instead of having low SCOAP values compared to other Trust-Hub benchmarks, the model correctly identifies all the TI nets from the $s35932-T200$ benchmark. However, for the $s38584-T100$ benchmark, some TF nets are wrongly predicted as TI nets because the SCOAP values of Trojan nets are less than the TF nets.

Similarly, we have observed from Table 4 that the CatBoost model provides considerably good prediction performance on the test dataset after training on relabeled CtD. It can be seen that model accurately predicts all the Trojan nets from $RS232-T1200$, $T1300$, $s38417-T100$, $s35932-T200$, $T300$ & $s38584-T200$ respectively by providing 100% recall. However, some of the benchmarks provide higher FNR, which ranges from $(8\%-33\%)$ and lower recall up to 66%, which indicates that some TI nets are misclassified by the model in these benchmarks. This happened because the proposed defense method could not detect some maliciously label flipped samples, which affected the prediction results.

5.2.4 Proposed Attack and Defense Results on DeTrust Benchmarks

We further check the performance of our proposed attack method on 11 DeTrust benchmarks, the before and after comparative attack results are shown in Table 6. It can be seen that the CatBoost provides on an average 1.34% loss,

Table 6 Comparative proposed attack results on Original and Poisoned DeTrust benchmarks (%)

DeTrust Benchmarks	Before attack						After proposed attack					
	Loss	Accuracy	Recall	ROC- AUC	TNR	FPR	FNR	Loss	Accuracy	Recall	ROC- AUC	TNR	FPR	FNR
RS232-T1000	0.331	100	100	100	100	0	0	40.87	77.01	83.33	80.05	76.77	23.22	16.66
RS232-T1100	3.48	99.38	83.33	91.66	100	0	16.66	44.8655	85.185	66	76.28	85.89	14.1	33.33
RS232-T1200	1.271	99.41	100	99.69	99.39	0.6	0	48.52	84.48	20	82.28	88.41	11.58	80
RS232-T1300	0.266	100	100	100	100	0	0	51.061	81.65	38.46	80.62	85.25	14.74	61.53
RS232-T1400	0.488	100	100	100	100	0	0	47.645	85.45	83.33	84.43	85.53	14.46	16.66
RS232-T1500	7.36	99.4	90	90.8	100	0	10	50.23	71.34	66	69.092	71.51	28.48	33.33
RS232-T1600	0.344	100	100	100	100	0	0	46.37	82.71	80	81.4	82.8	17.19	20
s38417-T100	0.12672	99.97	80	90	100	0	20	67.48	70.49	20	70	70.48	29.51	80
s38417-T200	0.06597	100	100	100	100	0	0	66.92	61.6	40	72.52	61.71	38.28	60
s38417-T300	0.0627	100	100	100	100	0	0	68.72	54.7	25	64.6	54.8	45.18	75
s35932-T200	1.031	100	100	100	100	0	0	67.37	67.83	42.85	72.79	67.88	32.11	57.14
Average	1.347854	99.83273	95.75727	97.46818	99.94455	0.054545	4.241818	54.55014	74.76773	51.36091	75.82382	75.54818	24.44091	48.51364

Table 7 Proposed Defense Result on re-labeled clean DeTrust benchmarks (%)

DeTrust Benchmarks	Loss	Accuracy	Recall	ROC- AUC	TNR	FPR	FNR
RS232-T1000	0.7	99.38	100	99.677	99.35	0.64	0
RS232-T1100	5.8046	98.77	83.3	91.346	99.35	0.64	16.66
RS232-T1200	5.62	98.82	66.66	83.33	100	0	33.33
RS232-T1300	1.233	99.39	83.33	91.66	100	0	16.66
RS232-T1400	0.495	100	100	100	100	0	0
RS232-T1500	8.79	98.21	70	86.33	100	0	30
RS232-T1600	2.88	99.38	80	90	100	0	20
s38417-T100	0.04699	99.97	80	90	100	0	20
s38417-T200	0.02275	100	100	100	100	0	0
s38417-T300	0.06762	100	100	100	100	0	0
s35932-T200	0.695	100	100	100	100	0	0
Average	2.395905	99.44727	87.57182	93.84936	99.88182	0.116364	12.42273

99.83% accuracy, and 0.054% FPR, i.e., almost correct prediction when trained on TrD. In contrast, our proposed attack method significantly reduces the model performance on the test dataset by providing on an average 54% loss and 74.76% accuracy. Moreover, the recall dropped heavily from 95% to 51%, the FNR rate increased from 4% to 48.511%, and the area covered also decreased from 97% to 75% which clearly shows that the proposed attack method heavily degraded the model prediction performance. Besides, if we individually see the degradation in classification performance during prediction, then it has been observed that benchmarks $RS232-T1200, T1300, s38417-T100, 200, 300$ provides only $20\%-40\%$ recall which means that most of the TI nets are predicted as TF nets.

Further, the proposed defense results on 11 DeTrust benchmarks are shown in Table 7, which shows that the proposed defense method correctly relabels the poisoned samples and improves the prediction performance of the CatBoost model. It can be analyzed that the proposed method achieves 100% accuracy in several benchmarks and $99-98\%$ accuracy in the rest of the benchmarks, which shows that the model can detect nearly all the TI/TF nets into their correct classes. The higher recall, TNR, and lower FPR/FNR rates also confirmed this. Moreover, there is a marginal difference (1%) between the on-an-average-loss achieved by the model trained on *TrD* and *PTrD*, which also shows that the object importance method capture almost all the malicious sample present in the *PTrD* and then k-NN successfully relabeled the malicious samples into their correct classes.

5.2.5 HT Detection Analysis: Results Obtained on Non-poisoned & Relabeled DeTrust Benchmarks

We have further analyzed the results with respect to HT detection on DeTrust benchmarks, it can be seen from Table 6 before the attack result that the CatBoost model detects all the TI & TF nets in four *RS*232 series, two *s*38417 & $s35932-T200$ benchmarks respectively. It indicates that SCOAP features again proved to be effective in identifying Trojans not only in Trust-Hub but in DeTrust benchmarks also. Further, we can see from Table 7 that the model detects all TI & TF nets from four benchmarks when trained on the relabeled dataset. However, the higher FNR rates in some of the benchmarks indicate that the proposed defense method missed out some of the malicious samples in these benchmarks, which leads to misclassification.

6 Conclusion

This paper proposed a new CatBoost and SCOAP features-based approach which accurately tackles the problem of a poisoned dataset where an adversary flips the sample labels to malign the ML model training so that HT nets are misclassified during prediction. A label flipping poisoning attack is proposed, which utilizes the SHC algorithm to identify the best samples from the training set, and amongst them, only 20% of samples are flipped, which increases the validation loss most. Besides, a new CatBoost model-based defense method is proposed in which the object importance method captures the malicious samples from the training set, and the k-NN model relabels them to their correct classes. The proposed attack and defense method are ML model-independent, i.e., the attacker and defender do not have any idea about the ML model used during training. Experimental analysis on Trust-Hub benchmarks shows that the proposed attack method decreases the on-an-average accuracy up to 67% on the test dataset, which is on an average 32.77% lower than the accuracy achieved on the model trained on a non-poisoned dataset. Further, the proposed defense method effectively identifies the malicious inputs from the poisoned dataset and improves the CatBoost model prediction performance by the rate of 32.57% by providing on-an-average 99.66% accuracy, which is comparable to the accuracy achieved on

the untainted training dataset. Finally, the results computed on the DeTrust benchmarks also show the efficacy of the proposed attack and defense method.

Data Availability The Trust-Hub benchmarks analyzed during this study are available at https://trust-hub.org/. Besides, the DeTrust 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. Aryal K, Gupta M, Abdelsalam M (2021) A survey on adversarial attacks for malware analysis. arXiv preprint arXiv:2111.08223
2. Basu K, Saeed SM, Pilato C, Ashraf M, Nabeel MT, Chakrabarty K, Karri R (2019) Cad-base: An attack vector into the electronics supply chain. ACM Trans Des Autom Electron Syst (TODAES) 24(4):1–30
3. Bhunia S, Hsiao MS, Banga M, Narasimhan S (2014) Hardware trojan attacks: threat analysis and countermeasures. Proc IEEE 102(8):1229–1247
4. Chakraborty RS, Narasimhan S, Bhunia S (2009) Hardware trojan: Threats and emerging solutions. In: Proc. IEEE International high level design validation and test workshop. pp 166–171
5. 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
6. Cheng N, Zhang H, Li Z (2021) Data sanitization against label flipping attacks using adaboost-based semi-supervised learning technology. Soft Comput 25(23)14573–14581
7. Clements J, Lao Y (2018) Hardware trojan attacks on neural networks. arXiv preprint arXiv:1806.05768
8. Elnaggar R, Chakrabarty K (2018) Machine learning for hardware security: Opportunities and risks. J Electron Test 34(2):183–201
9. Fern N, Kulkarni S, Cheng K-TT (2015) Hardware trojans hidden in RTL don't cares-automated insertion and prevention methodologies. In: Proc. IEEE International Test Conference (ITC). pp 1–8
10. Friedman JH (2001) Greedy function approximation: a gradient boosting machine. Ann Stat pp. 1189–1232
11. Gao Y, Doan BG, Zhang Z, Ma S, Zhang J, Fu A, Nepal S, Kim H (2020) Backdoor attacks and countermeasures on deep learning: A comprehensive review. arXiv preprint arXiv:2007.10760
12. Goldstein LH, Thigpen EL (1980) Scoap: Sandia controllability/observability analysis program. In Proceedings of the 17th Design Automation Conference pp. 190–196
13. Goodfellow IJ, Shlens J, Szegedy C (2014) Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572
14. Gu T, Liu K, Dolan-Gavitt B, Garg S (2019) Badnets: Evaluating backdooring attacks on deep neural networks. IEEE Access 7:47230–47244
15. Hasegawa K, Oya M, Yanagisawa M, Togawa N (2016) Hardware trojans classification for gate-level netlists based on machine learning. In: Proc. 22nd International Symposium on On-Line Testing and Robust System Design (IOLTS). 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: Proc. IEEE International Symposium on Circuits and Systems (ISCAS). pp 1–4
17. Hasegawaa K, Yanagisawa M, Togawa N (2017) Hardware trojans classification for gate-level netlists using multi-layer neural networks. In: Proc. IEEE 23rd International Symposium on On-Line Testing and Robust System Design (IOLTS). pp 227–232
18. Hu W, Zhang L, Ardeshiricham A, Blackstone J, Hou B, Tai Y, Kastner R (2017) Why you should care about don't cares: Exploiting internal don't care conditions for hardware trojans. In: Proc. IEEE/ACM International Conference on Computer-Aided Design (ICCAD). pp 707–713
19. 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
20. Jacob N, Merli D, Heyszl J, Sigl G (2014) Hardware trojans: current challenges and approaches. IET Comput Digit Tech 8(6):264–273
21. Kaviani S, Sohn I (2021) Defense against neural trojan attacks: A survey. Neurocomputing 423:651–667
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: Proc. IEEE 28th Asian Test Symposium (ATS). 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: Proc. IEEE International Symposium on Circuits and Systems (ISCAS). pp 1–5
25. Kurihara T, Togawa N (2021) Hardware-trojan classification based on the structure of trigger circuits utilizing random forests. In: Proc. IEEE 27th International Symposium on On-Line Testing and Robust System Design (IOLTS). pp 1–4
26. Li H, Liu Q, Zhang J (2016) A survey of hardware trojan threat and defense. Integration 55:426–437
27. Liakos KG, Georgakilas GK, Moustakidis S, Sklavos N, Plessas FC (2020) Conventional and machine learning approaches as countermeasures against hardware trojan attacks. Microprocess Microsyst p. 103295
28. Liu W, Chang C-H, Wang X, Liu C, Fung JM, Ebrahimabadi M, Karimi N, Meng X, Basu K (2021) Two sides of the same coin: Boons and banes of machine learning in hardware security. IEEE J Emerging Sel Top Circuits Syst 11(2):228–251
29. Liu Y, Mondal A, Chakraborty A, Zuzak M, Jacobsen N, Xing D, Srivastava A (2020) A survey on neural trojans. In: Proc. 21st International Symposium on Quality Electronic Design (ISQED). pp 33–39
30. Liu Y, Xie Y, Srivastava A (2017) Neural trojans. In: Proc. IEEE International Conference on Computer Design (ICCD). pp 45–48
31. Mondal A, Biswal RK, Mahalat MH, Roy S, Sen B (2021) Hardware trojan free netlist identification: A clustering approach. J Electron Test 37(3):317–328
32. Nahiyan A, Sadi M, Vittal R, Contreras G, Forte D, Tehranipoor M (2017) Hardware trojan detection through information flow security verification. In: Proc. IEEE International Test Conference (ITC). pp 1–10
33. Nozawa K, Hasegawa K, Hidano S, Kiyomoto S, Hashimoto K, Togawa N (2019) Adversarial examples for hardware-trojan detection at gate-level netlists. In: Comput Secur. Springer, pp 341–359
34. Paudice A, Muñoz-González L, Lupu EC (2018) Label sanitization against label flipping poisoning attacks. In: Joint European conference on machine learning and knowledge discovery in databases. Springer, pp 5–15
35. 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
36. Peterson LE (2009) K-nearest neighbor. Scholarpedia 4(2):1883
37. Pilato C, Basu K, Regazzoni F, Karri R (2018) Black-hat high-level synthesis: Myth or reality? IEEE Transactions on Very Large Scale Integration (VLSI) Systems 27(4):913–926
38. Pitropakis N, Panaousis E, Giannetsos T, Anastasiadis E, Loukas G (2019) A taxonomy and survey of attacks against machine learning. Comput Sci Rev 34:100199
39. Prokhorenkova L, Gusev G, Vorobev A, Dorogush AV, Gulin A (2017) Catboost: unbiased boosting with categorical features. arXiv preprint arXiv:1706.09516
40. Rawal A, Rawat D, Sadler BM (2021) Recent advances in adversarial machine learning: status, challenges and perspectives. Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications III 11746:701–712
41. Rostami M, Koushanfar F, Karri R (2014) A primer on hardware security: Models, methods, and metrics. Proc IEEE 102(8):1283–1295
42. Russell SJ (2010) Artificial intelligence a modern approach. Pearson Education, Inc
43. 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
44. Salmani H, Tehranipoor M, Karri R (2013) On design vulnerability analysis and trust benchmarks development. In: Proc. IEEE 31st international conference on computer design (ICCD). pp 471–474
45. Samimi SMS (2016) Testability measurement tool. https://sourceforge.net/projects/testabilitymeasurementtool/
46. Sharma R, Valivati NK, Sharma G, Pattanaik M (2020) A new hardware trojan detection technique using class weighted xgboost classifier. In: Proc. 24th International Symposium on VLSI Design and Test (VDAT). pp 1–6
47. Taheri R, Javidan R, Shojafar M, Pooranian Z, Miri A, Conti M (2020) On defending against label flipping attacks on malware detection systems. Neural Comput Appl 32(18):14781–14800
48. Wang J, Hassan GM, Akhtar N (2022) A survey of neural trojan attacks and defenses in deep learning. arXiv preprint arXiv:2202.07183
49. Wang Y, Han T, Han X, Liu P (2019) Ensemble-learning-based hardware trojans detection method by detecting the trigger nets. In: Proc. IEEE International Symposium on Circuits and Systems (ISCAS). pp 1–5
50. Xiao H, Biggio B, Nelson B, Xiao H, Eckert C, Roli F (2015) Support vector machines under adversarial label contamination. Neurocomputing 160:53–62
51. Xiao H, Xiao H, Eckert C (2012) Adversarial label flips attack on support vector machines. In: ECAI 2012. IOS Press, pp 870–875
52. Xiao K, Forte D, Jin Y, Karri R, Bhunia S, Tehranipoor M (2016) Hardware trojans: Lessons learned after one decade of research. ACM Trans Des Autom Electron Syst (TODAES) 22(1):6
53. Xie X, Sun Y, Chen H, Ding Y (2017) Hardware trojans classification based on controllability and observability in gate-level netlist. IEICE Electronics Express 14(18):20170682–20170682
54. Xue M, Gu C, Liu W, Yu S, O'Neill M (2020) Ten years of hardware trojans: a survey from the attacker's perspective. IET Comput Digit Tech 14(6):231–246
55. Xue M, Yuan C, Wu H, Zhang Y, Liu W (2020) Machine learning security: Threats, countermeasures, and evaluations. IEEE Access 8:74720–74742
56. Yang Y, Ye J, Cao Y, Zhang J, Li X, Li H, Hu Y (2020) Survey: Hardware trojan detection for netlist. In: Proc. IEEE 29th Asian Test Symposium (ATS). pp 1–6
57. Zhang H, Cheng N, Zhang Y, Li Z (2021) Label flipping attacks against naive bayes on spam filtering systems. Appl Intell 51(7):4503–4514
58. 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 M.Tech. 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, and 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. 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, Ground Bounce Noise Reduction Techniques, and reliability aware high performance energy-efficient embedded computing.