PCB Hardware Trojan Run-Time Detection Through Machine Learning

Gor Piliposyan and Saqib Khursheed

Abstract—The modern semiconductor electronic devices are becoming increasingly vulnerable to malicious implants called Hardware Trojans (HT). This problem is also greatly related to Printed Circuit Boards (PCB), which are widely used in almost all electronic devices. In this paper, two machine learning (ML) methods have been applied to detect HTs running on power from I/Os of legitimate chips on a PCB. A PCB prototype has been fabricated to obtain real-life data, which was used to train two ML algorithms: One-Class Support Vector Machine and Local Outlier Factor. For validation of the ML classifiers, one hundred categories of HT devices have been modelled and inserted into the Validation and Testing datasets. Simulation results show that using the proposed methodology an HT device can be detected with high prediction accuracy (F1-score above 99.7% for a $50mW$ HT). Further, the ML model has been uploaded to the prototype PCB for hard-silicon validation of the methodology. To the best of our knowledge, this is the first work on real-time detection of PCB HTs, which are powered from the I/O pins of legitimate ICs. Experimental results show that the performance of the ML model on a real-life prototype is consistent with that of the simulations.

Index Terms—Hardware Trojan (HT), Printed Circuit Board (PCB), Machine Learning on Microcontroller, One Class Classification, One-Class Support Vector Machine, Local Outlier Factor.

1 Introduction

Printed Circuit Boards (PCB) are organic parts of electronics across a wide range of industries including consumer electronics (smartphones, computers etc), medical devices, automotive components (navigation, control systems and sensors), telecommunications equipment, military and defence applications. Companies often outsource the production of PCBs around the world to reduce the manufacturing costs, satisfy the demand and reduce time-to-market. This makes PCBs vulnerable to various malicious implants at different production stages and the supply chain [1]. A report on Hardware Trojan attacks on PCBs was published by Bloomberg in 2018 [2]. The report described a formidable scale tampering attack on private servers with a tiny malicious chip which tricked the motherboards, providing backdoor access when the server booted up. According to the article, about thirty companies downstream the supply chain from Super Micro Computer company, including Amazon, Apple as well as a few government contractors, were heavily affected. The Bloomberg article was later analysed in [3] where the vulnerabilities in the modern PCB supply chain and existing and viable countermeasures were discussed. The attack described by Bloomberg in [2] is a typical Hardware Trojan (HT) attack when the adversaries damage the system by malicious physical alterations [4].

In the last few years several PCB assurance approaches have been developed which can be categorised as in-circuit testing, functional testing, JTAG boundary scan, bare-board testing and visual inspection [3]. With hardware attacks becoming stealthier, finding more advanced detection methods has now become crucial. Particularly important are automated quality assurance (AQA) methods, which being non-destructive can detect a wide range of HTs on a PCB with minimal human involvement [3]. One of these methods is automated visual inspection (AVI) for PCB verification and authentication. This method is about testing on assembly lines during manufacturing, replacing human oversight, thus preventing unauthorized modifications during assembly. This PCB AQA approach can be used in different stages of the manufacturing cycle and can be categorized by the imaging modalities of choice and image analysis techniques [5]. Several AVI have been suggested including canonical image processing methods for detecting trace and via level defects [6], convolutional neural network for detecting known defects [7], automated detection for component misplacement by directly comparing golden and test PCBs [8], [9] and text detection on the PCB for verification purposes [10], [11]. Although AVI has so far been the most commonly used method for PCB assurance it has limitations such as absence of run-time monitoring option, demand for significant subject-matter expert involvement [12], [13], and being less accurate than some other available approaches.

Conventional PCB quality assurance techniques have not been inherently designed for detecting deliberate malicious alterations. Advancements in state-of-the-art technology, possibly used by adversaries creating stealthier HTs, introduces further strain on legacy AQA techniques. Novel machine learning (ML) based techniques, on the other hand, provide a fresh perspective on effective countermeasures for detecting HTs on PCBs [3], [14]. For instance, the methodology proposed in [15] detects both, always active and triggerable HTs on the PCB. Using a distributed power sens-

Corresponding author: Gor.Piliposyan@liverpool.ac.uk
This project was funded by the Department of Electrical Engineering and Electronics, University of Liverpool, UK. We would also like to thank the Ministry of High-Tech Industry of the Republic of Armenia for supporting the study.
Gor Piliposyan and Saqib Khursheed are with the Department of Electrical Engineering and Electronics, University of Liverpool, Liverpool, UK (e-mail Gor.Piliposyan@liverpool.ac.uk, S.Khursheed@liverpool.ac.uk)

ing architecture, it is experimentally demonstrated that the proposed method can detect HTs with power consumption as low as $7.5mW$, powered from the power distribution network, similar to Trojan A in Fig. 1. However, this monitoring system would be oblivious to any HT that is powered from the input/output (I/O) pins of the legitimate integrated circuits (ICs), similar to Trojan B in Fig. 1. The main novelty in this paper is using ML algorithms implemented on the monitoring block, combined with a similar monitoring circuit architecture (Fig. 1) as proposed in [15], to detect HTs powered from the I/Os of legitimate ICs on the PCB. Using ML to inspect power consumption for HT detection, as proposed in current work, is readily scalable, easy to deploy on a given PCB design and much less intrusive than other approaches such as impedance anomaly detection, requiring no extra actions taken by the ICs on the existing circuitry.

The proposed monitoring setup should be designed and manufactured alongside the original PCB circuitry. The ML model training should be done on data from a complete simulation model or, preferably, group of prototype PCB devices, prior to large scale manufacturing. To ensure the integrity of the HT-clean power consumption training data, the batch of prototype PCBs can be subjected to reverse-engineering testing or expert visual inspection. Both methods are used to test the golden PCBs for HT presence, but have their own shortcomings in volume manufacturing.

Prior Work

Machine learning techniques have a huge potential for addressing countermeasures for various HT attacks [16], [17]. Such methods have already been widely applied for IC HT detection, improving the detection capabilities in a number of aspects including reverse engineering [18], side-channel analysis [19], [20], real-time detection and for HT detection on gate-level netlists [21]. A comprehensive survey of the latest research and developments of ML-based approaches for HT detection and prevention on an IC level is carried out in [22], where it has been demonstrated that the number of research publications and achievements in IC HT detection using ML techniques have been growing rapidly since 2014. This shows that ML is admittedly a promising tool for guaranteeing security of hardware. HT vulnerabilities for PCBs are not too different to those at the IC level. Hence the ML methods applied for countermeasures against HTs at the IC level can be useful in HT detection and prevention at PCB level. While ML algorithms have proved to be highly promising in overcoming IC HT threats, ML based techniques for higher levels of system abstraction such as PCBs have not been well studied. That said, some research has very recently become available. Hammond et al. [23] have proposed an HT detection methodology utilising multiple side-channel performances, including power consumption, temperature and electromagnetic field as well as communication and CPU activity patterns. In their approach they use defender controlled operating environments to have multiple views on the performance of the PCB, hence a higher chance to detect the presence of an HT. Another promising approach is taken in [24], [25], [26] and [27], where the key HT detecting factor is the alteration to the expected impedance value of either the power distribution network (PDN) or the metal connections between the legitimate ICs.

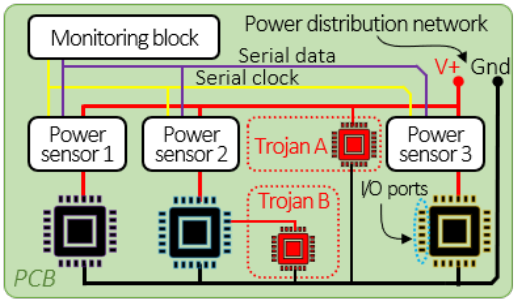


Fig. 1: PCB diagram with proposed power monitoring circuit, Trojan A powered from the power distribution network and Trojan B powered from I/O port of a legitimate chip.

In case of a sufficiently high resolution this approach is capable of detecting minor alterations to the circuit.

The rest of the paper is organised as follows: The attacker model and the proposed approach are described in Section 2. In Section 3 theoretical background is provided on two ML algorithms used in this work. Section 4 explains the proposed methodology, while Section 5 describes the experimental setup, including the prototype PCB. Section 6 discusses simulation results. Further, real-life experimental results are given in Section 7. Discussion around the proposed methodology is provided in Section 8. Finally, the paper is concluded in Section 9.

2 Attacker Model and Our Approach

When added to the original PCB circuitry, an HT device can draw power from sources such as the mains supply, a built in battery, capacitor or energy harvester, the power distribution network or an I/O pin (or trace) of a legitimate IC. A thoughtful adversary would try to conceal the modifications in the internal layers of the PCB to avoid detection. If the HT is powered from the mains supply, it would need external vias and traces for linkage. Similarly, a battery, a capacitor or energy harvester would be too large to be concealed in the internal layers. In both of these cases an automated visual inspection can be sufficient to detect any external modifications on the device, thus ultimately leading to the detection of the HT. On the other hand, a carefully designed HT can completely escape detection from visual means of inspection, if it is powered from the power distribution network or an I/O trace of a legitimate IC. The first of these cases has already been addressed in [15], where the suggested differential power monitoring method demonstrated sufficiently high accuracy in detecting an HT which consumes power from the power distribution network. This paper addresses this problem by proposing a methodology for detecting such HTs using machine learning.

The main characteristics of the proposed approach are:

1) **Trojan payload invariant:** The proposed methodology is independent from the types of payload and trigger mechanisms of the HT device, assuming it consumes power.

2) **Real-time monitoring:** Continuous monitoring of the power consumption patterns on the PCB in real-time, searching for anomalies.

3) **High prediction quality:** Classification F1-score can reach over 99.5%, depending on the HT parameters (e.g. power consumption, active time).

4) **No interference with the performance of the original circuit:** The approach is completely detached from the computational provisions of the circuit under inspection, continuously running its algorithms without degrading the performance or affecting the throughput of the PCB.

Assumptions

It is assumed that the adversary can use any HT regardless of its payload and trigger, as long as the activation of the HT alters the power consumption of the original circuitry. It is also assumed that the IC and PCB design houses, as well as the firmware and intellectual properties for designing the ICs are trusted. The only possible threat comes from outsourcing PCB production to untrusted facilities, or during the transfer of the product from the manufacturer to the consumer.

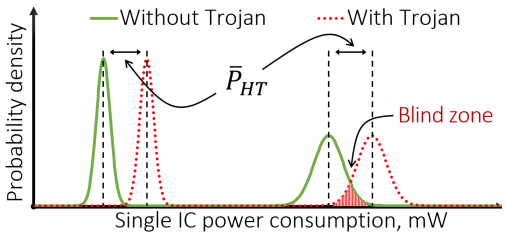
There is also a possibility of adversaries swapping a trusted IC on the PCB with an HT infected IC. This attack scenario is out of the scope of this paper since it is concerned with the research of IC level HTs, which discusses modifications of ICs during their design and production phases. The alternative case assuming the adversary has arranged their separate and unpoliced illegal IC production is unlikely given the prohibitive financial constraints on organising silicon chip production and limited access to original IC design and layout information. Thus, it can be assumed that the ICs on the PCB are trusted and HT-clean.

Our Approach

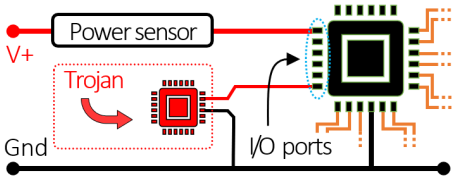
The main function of the proposed approach is to differentiate between two groups of power consumption data, HT-clean and HT-contaminated, incoming during run-time. The HT-clean data group belongs to the case, where no HT has been implanted on the PCB and the device is functioning as expected. This case includes all the tasks and modes that the device is expected to operate in with their respective power consumption patterns. Any deviation from the expected power performance is deemed to be a fault or an HT attack, and the corresponding data-point belongs to the HT-contaminated group. This second group, where all the HT-contaminated data-points reside, is harder to model since it must include all the possible HT attack scenarios. To tackle this problem, one class classification machine learning algorithms are used in this work for anomaly detection. The algorithms should be able to separate clean and contaminated data-points, having previously seen only the group with clean data-points. Such algorithms are commonly referred to as anomaly detection algorithms.

Blind Zone

Next we describe what is referred in the rest of this paper as blind zone. To simplify the problem at hand, let us consider a case with a single IC on the PCB. Let us further assume that this IC has two work modes: idle and operating. In that case it would be safe to assume that its power consumption probability density function would look like the green line in Fig. 2a, with two distinct peaks, one for every work mode. When a Trojan device is added on the PCB, which consumes power from an I/O pin of the legitimate IC Fig. 2b, the resulting new probability density function will be skewed to the right by the mean power consumption value of the HT. This new distribution is drawn in red in Fig. 2a. Intersection of the regions under green and red functions are where it is not possible to confidently separate HT-clean and HT-contaminated cases. Henceforth such patches will be referred to as blind zones.



(a) Power consumption probability density functions



(b) Legitimate IC and an implanted Trojan schematic

Fig. 2: Illustration of a single IC and single Trojan.

3 Anomaly Detection Classifiers

3.1 Anomaly detection: Novelties and Outliers

Anomaly detection algorithms are divided into two machine learning method subgroups: novelty detection and outlier detection. While in novelty detection the training data does not contain anomalies, which can only appear in new observations in the prediction stage, in outlier detection the training data does contain some anomalies. In the outlier detection method the algorithm finds the regions where the training data is mostly congregated, ignoring the observations which have deviated from the majority data. For the HT detection problem on PCB set out in this research novelty detection approach is most appropriate. Two relevant ML algorithms for power consumption novelty detection are one-class classifiers based on One-Class Support Vector Machine (SVM) and Local Outlier Factor (LOF) implemented in Scikit-learn library [28].

3.2 One-Class Support Vector Machine

If the conventional SVM method separates different classes existing within a given dataset [29], One-Class SVM [30] minimises the radius of a hypersphere that encloses the training data consisting of only one class. This is achieved through a suitable kernel function which maps feature vectors onto a higher dimension feature space. The method defines an origin and finds a hyperplane which separates the origin and the mapped feature space with a maximum margin. Thus unlike multi-class SVM where the maximum margin between different classes is learned, in One-Class SVM the boundary of the training (Trojan free) data is

learned. The decision function $f(\mathbf{x})$ assigns 1 in the region surrounding the bulk of the training points (HT-clean points), and -1 everywhere else (HT-contaminated points).

To introduce one-class SVM we consider the training data consisting of n observations

$$\Omega = \{\mathbf{x}_i, i = 1, 2, 3, ..., n\}, \quad n \in \mathbb{N}$$

where $\mathbf{x}_i$ is the i^{th} observation (data-point) and it is assumed that Ω is mapped into a higher-dimensional feature space Ψ through a feature mapping Φ

$$\Phi : \Omega \to \Psi. \tag{1}$$

The mapping (1) is assumed such that the images of dot products $(\mathbf{x}_i \cdot \mathbf{x}_j) \in \Omega$ in the new feature space $(\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)) \in \Psi$ can be calculated via a kernel function [31]

$$k(\mathbf{x}_i, \mathbf{x}_j) = (\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)). \tag{2}$$

To find a hyperplane in the new feature space which separates the data and the origin with a maximum margin the following minimising problem is formulated

$$\min_{\mathbf{w}\in\Psi,\, \boldsymbol{\xi}\in R^n,\, \rho\in R} \frac{1}{2}\|\mathbf{w}\| + \frac{1}{\nu n}\sum_i \xi_i - \rho, \tag{3}$$

with the following constraints

$$(\mathbf{w} \cdot \Phi(\mathbf{x}_i)) \geq \rho - \xi_i, \quad \xi_i \geq 0, \;\; i = 1, 2, ..., n. \tag{4}$$

In the quadratic programming problem (3)-(4), which is called the primary problem, $\mathbf{w}$ and ρ are the parameters needed to be found, where $\mathbf{w}$ is the normal vector to the separating hyperplane, ρ is the margin between the mapped data and the origin, ξ_i are slack variables penalising the misclassifications. Here and further bold face greek letters denote n dimensional vectors whose components are denoted by normal face typeset. The regularization parameter $\nu \in (0, 1]$ is a user-defined parameter, which shows the fraction of outliers that should be allowed and n is the number of training points.

If the data in the input space is not linearly separable, which is often the case in one-class data, the so-called kernel trick is employed, which maps the inseparable initial data onto a separable data in the higher dimensional kernel space. This can be done by writing the primary problem in the following equivalent form, called the dual problem [32]

$$\min_{\boldsymbol{\alpha}\in R^n} \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j k(\mathbf{x}_i, \mathbf{x}_j), \quad 0 \leq \alpha_i \leq \frac{1}{\nu n}, \; \sum_i \alpha_i = 1, \tag{5}$$

where the kernel $k(\mathbf{x}_i, \mathbf{x}_j)$ is defined by the dot product (2), indeces i and j range over $1, ..., n$. The dual problem (5) does not require computation of the explicit map Φ in the objective function. It is sufficient to define the kernel instead. One of the most efficient kernels (2) for the dual problem (5) which will be used further is the radial basis function (RBF)

$$k(\mathbf{x}, \mathbf{x}_i) = \mathrm{e}^{-\gamma\|\mathbf{x}-\mathbf{x}_i\|^2}, \tag{6}$$

where γ is the bandwidth parameter.

Further, the inequality constraints (4) in the primal formulation (3) (which are as many as the number of samples) are simplified to an equality and bound constraint in the dual problem (5). The algorithm finds the optimal values of α_i and only the solutions $\{\mathbf{x}_i\}$ with $\alpha_i > 0$, called support vectors, are taken in the final decision function.

After the minimization problem (5) is solved the required value of ρ can be found from the following formula

$$\rho = \sum_i \alpha_i k(\mathbf{x}_i, \mathbf{x}_j), \tag{7}$$

where $\mathbf{x}_j$ is any support vector. The new incoming data can be classified using the following decision function:

$$f(\mathbf{x}) = \mathrm{sgn}\left(\sum_i \alpha_i k(\mathbf{x}_i, \mathbf{x}) - \rho\right), \tag{8}$$

where $\mathbf{x}_i$ are the support vectors. The decision function is assigned 1 for data predicted as HT-clean, and -1 for HT-contaminated data.

3.3 Local Outlier Factor

Another classic anomaly detection method is Local Outlier Factor (LOF) [33]. The method estimates the local density of a given sample and the deflection from the density of its neighbours. The local density is estimated by measuring distances of k nearest neighbours. Then samples are identified with considerably lower density and qualified as outliers.

The LOF approach is based on the concept of k-distance, $\|\mathbf{x} - \mathbf{x}^{(k)}\|$, which is the distance of a point $\mathbf{x}$ from its k^{th} nearest neighbours, and k-neighbours, $N_k(\mathbf{x})$, which are points in and on the circle of the radius k-distance.

Reachability distance (RD) from $\mathbf{x}$ to $\mathbf{x}'$ is

$$RD_k(\mathbf{x}, \mathbf{x}') = \max(\|\mathbf{x} - \mathbf{x}^{(k)}\|, \|\mathbf{x} - \mathbf{x}'\|), \tag{9}$$

where $\mathbf{x}^k$ is the k^{th} nearest neighbour of $\mathbf{x}$ and $\mathbf{x}' \in N_k(\mathbf{x})$. RD is used to define another key concept, the local reachability density (LRD) of $\mathbf{x}$:

$$\mathrm{LRD}_k(\mathbf{x}) = \frac{1}{\frac{1}{\|N_k(\mathbf{x})\|}\sum_{\mathbf{x}'\in N_k(\mathbf{x})} RD_k(\mathbf{x}, \mathbf{x}')}. \tag{10}$$

It follows from (10) that greater LRD (i.e. neighbours far from the point $\mathbf{x}$) corresponds to lower density of points around a particular point. The algorithm uses the definition of the local outlier factor (LOF) of $\mathbf{x}$, which is the ratio of the average LRD of the k-neighbours to the LRD of $\mathbf{x}$:

$$\mathrm{LOF}_k(\mathbf{x}) = \frac{\frac{1}{\|N_k(\mathbf{x})\|}\sum_{\mathbf{x}'\in N_k(\mathbf{x})} \mathrm{LRD}_k(\mathbf{x}')}{\mathrm{LRD}_k(\mathbf{x})}. \tag{11}$$

The aim of the LOF method is to determine an outlier by calculating the local outlier factor (11) for each data-point in the dataset [34]. Larger values of the LOF correspond to fewer data-points around the object. If for any point the LOF is less than 1, the density of the point is higher compared to that of its neighbours indicating that the point is HT-clean. If the LOF is greater than 1, then the point has less density compared to the density of its neighbours. In this case the point is possibly an HT-contaminated outlier.

3.4 Algorithm Complexities

The complexity of an algorithm is estimated by its time complexity, which assesses how long the algorithm takes to execute the task, and its spatial complexity, which assesses the amount of memory needed for the operation of the algorithm.

Complexity of One-Class SVM

There are two complexities to assess: training-time and run-time. Minimum training time complexity for SVM is estimated as $O(n^2)$, where n is the number of training data-points. The training process returns n_{SV} support vectors, which is later used for classification.

For a kernelised SVM with an RBF kernel (6), n_{SV} kernel computations are needed to classify a new point $\mathbf{x}$ in (8). Assuming for each support vector $\mathbf{x}_i$ the kernel (6) can be computed with $O(d)$ time complexity, the overall run-time complexity of the classifier with an RBF kernel is $O(n_{SV} \times d)$ for a single data-point under query. Although tuning of the γ parameter can have an impact on run-time as well since its computation has a quadratic complexity, one-class SVM achieves good performance without significant tuning [35].

Spatial complexity

The classification ready model, needs to know the coordinates of the support vectors, therefore all n_{SV} support vectors should be stored in memory. The dimensionality of the support vectors d matches that of the data for classification. Therefore the memory complexity for online prediction will be $O(n_{SV} \times d)$.

Complexity of LOF

Nearest-neighbour-based algorithms such as LOF have quadratic computational complexity since they have to calculate the distances of all data-points. Thus training time complexity of LOF algorithm is $O(n^2)$. Run-time and space complexities of LOF are estimated as $O(nd)$.

4 Proposed Methodology

Every electronic device, when performing a certain group of tasks, executes a unique set of commands in a predefined order. When an IC device (Micro-controller, FPGA, ASIC) executes a specific command, the internals of the IC switch through a number of states. For every low-level command, a certain number of transistor switches occur in a strictly defined pattern. On the macro scale this creates a specific power consumption pattern for the IC for every high-level command [36]. Since a PCB is composed of a group of ICs performing their tasks, the same is true for a PCB. The proposed methodology identifies the underlying power consumption pattern for the device under test and feeds the data to a machine learning (ML) algorithm to learn the decision boundary[1] or some other classification criteria.

Every IC on the PCB has its specific set of tasks required for the PCB to operate as a whole. The exact ICs and their tasks can vary depending on the device and its use case. The device can have different operational modes, however for every device (e.g. electronic control unit on car, WiFi router, etc.) these tasks and mode patterns are limited and so are combinations of power consumption patterns for the ICs on the PCB. Given the sensitivity of the proposed technique to the IC activities during runtime the models have to be calibrated to predefined tasks of the PCB. In other words, for the proposed methodology to be actionable the device should not be re-programmable, changing the pre-designed tasks.

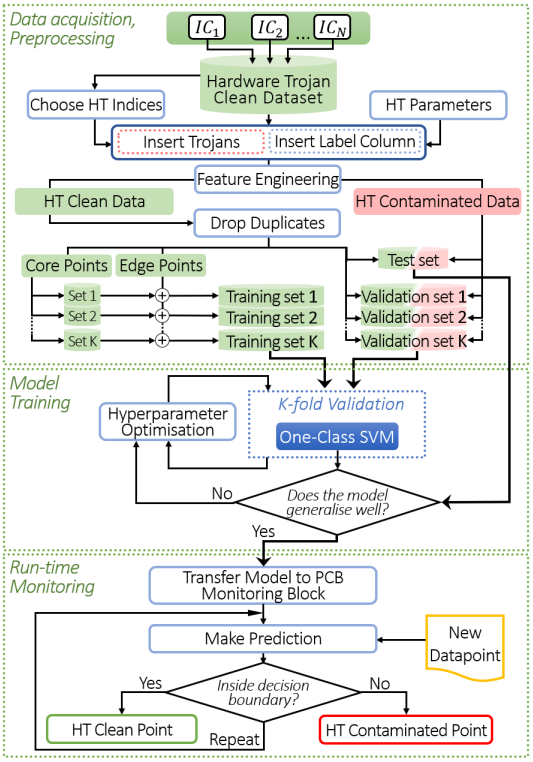


Fig. 3: Process flow from data collection to real-time Hardware Trojan monitoring on a PCB.

As proposed in [15], every component (or group of components) on the PCB should be fitted with a dedicated power sensor. On every iteration, these sensors report their readings back to the Monitoring Block (MB), where data processing and prediction are carried out. New data-points are parsed through the ML model to get binary predictions of either 1 or -1 for HT-clean and HT-contaminated scenarios respectively. The proposed methodology has been summarised in Fig. 3, organised into three main stages: data acquisition and prepossessing, model training, and run-time monitoring. In the first stage the power consumption data was collected from the PCB prototype and organised into Training, Validation and Testing datasets. In this stage, HT-contaminated data has been added to the Validation and Testing datasets using 100 distinct HT categories, later presented in Table 2. The second stage of the flow involves training the chosen ML algorithm. In the process of training the algorithm, K-fold validation has been applied to increase confidence on output model. The resulting model was tested on the Test dataset to check if it generalises well on unseen data, otherwise further hyperparameter optimisation is carried out. The final model was passed to the third stage and

1. The decision boundary is a closed shape in N-dimensional space, which determines the outcome of the classification prediction.

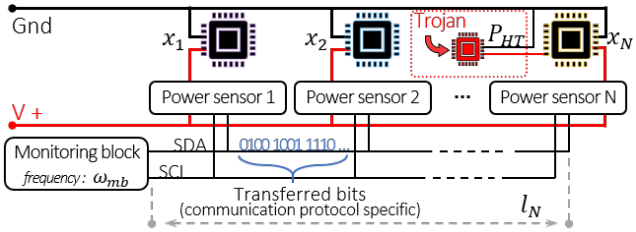


Fig. 4: The proposed monitoring architecture and Trojan on an I/O of the N^{th} IC.

transferred to the MB on the PCB prototype. Finally, the MB iteratively receives readings from all power sensors and performs its prediction regarding the presence of an HT.

Notations

4.1 Data point-vector

On every iteration, the incoming data is stored in the MB. To organise the collected data in an orderly manner, it is grouped in $[1 \times N]$ dimensional data point-vectors:

$$\mathbf{X} = \{x_1, x_2, x_3, ..., x_N\}. \tag{12}$$

Since each vector component corresponds to a reading from one of the N power sensors (Fig. 4), where every power sensor monitors an individual IC, the dimension N of vector $\mathbf{X}$ depends on the number of such sensors on the PCB.

4.2 Classification prediction

In the HT detection (classification) problem there are two possible labels. For HT-contaminated data-points, the label is assigned -1, while for HT-clean data-points the label is assigned 1.

The label can be actual ($\mathbf{y}$) or predicted ($\hat{\mathbf{y}}$). The actual label $\mathbf{y}$ carries the ground truth information and is considered correct. The predicted label $\hat{\mathbf{y}}$, on the other hand, is the label given by the trained ML model in the prediction process and can potentially be erroneous. In supervised classification problem, the quality of the trained ML model is measured by the parity of the actual label $\mathbf{y}$ and predicted label $\hat{\mathbf{y}}$, with the best case being a complete match between the two.

4.3 HT power consumption

Depending on the chosen model, power consumption of the Trojan, denoted by P_{HT} (Fig. 4), can be modelled using different probability density functions (uniform, normal, etc.), with their respective parameters. In this work, the HT power consumption has been modelled using a number of Gaussian normal probability density functions

$$f_i(P_{HT}) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left[\frac{-(P_{HT} - \bar{P}_{HTi})^2}{2\sigma_i^2}\right], \quad i \in \mathbb{N} \tag{13}$$

where $\bar{P}_{HTi}$ is the mean and σ_i is the standard deviation of the i^{th} HT.

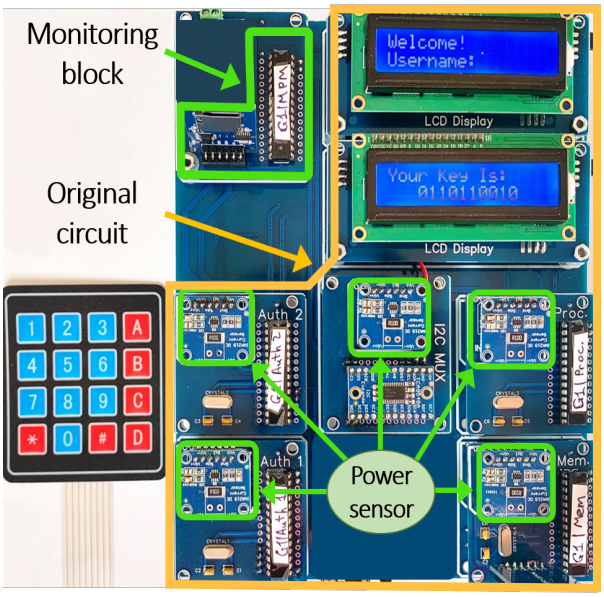


Fig. 5: Printed circuit board prototype.

4.4 HT active time

The HT active time span is denoted by T_{HT}. On every iteration the MB generates a data point-vector. The vector contains power consumption information gathered from all the power sensors, one of which can potentially bear an HT-contaminated IC. In order to measure the active time span of the HT, a sufficiently general time evaluation metric is necessary since the probing frequency of the monitoring setup will vary depending on the characteristics of different real-life PCB devices (e.g. working frequencies, number of ICs). Such an evaluation metric can be the number of consecutively recorded HT-contaminated data point-vectors i.e. HT-points. In other words, one HT-point is the time required for a single power consumption data point-vector to be registered by the MB, while the HT has been operating and consuming power. The relation of the HT active time to real time can be expressed via the number of HT-points using the time unit defined as

$$\textit{Unit HT-point} = \alpha_1 \frac{N_{sensors}}{\omega_{MB}} + \alpha_2, \tag{14}$$

where coefficient α_1 depends on the operational speed of the communication protocol, $N_{sensors}$ is the number of sensors, ω_{MB} is the working frequency of the monitoring block and α_2 depends on the characteristics of the PCB such as the layout and the length of data wires (e.g. l_N) between power sensors and the monitoring block (Fig. 4).

5 Data Collection and HT Modelling

Prototype PCB in Fig. 5 has been developed for real-life power consumption data collection. Subsequently, 100 categories of HT devices have been modelled and injected into the data for validation and testing of the trained ML model. Although it is possible that more than one IC can be infected with an HT at the same time, only single HT scenarios have been considered in this paper.

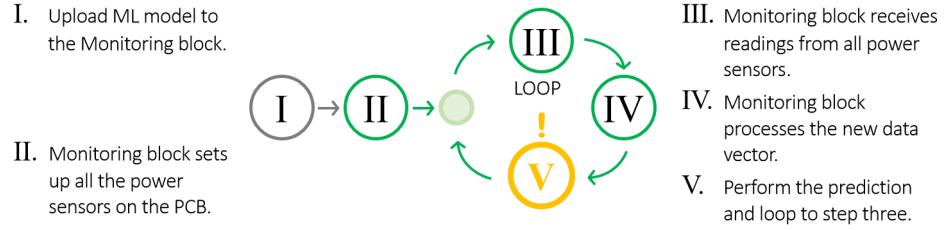


Fig. 6: Monitoring block run-time power monitoring and Trojan prediction steps.

5.1 The Prototype Device

5.1.1 The Original Circuit

The function of the original circuit (Fig. 5, marked by orange border) is to store and display data from the built-in memory block, after a log-in occurrence. On the prototype PCB there are five IC devices under power monitoring. A keyboard and a display, linked to an authentication block, are used to facilitate the log-in process. Upon a successful log-in event, an enable signal is generated by the authentication block. This signal is fed into a processing block. The enable signal triggers the processing block to fetch the secret data from a memory block. Here, the processing block uses Inter-Integrated Circuit (I_2C) communication protocol to request from the memory block. In turn, the memory block requests the data from a built in memory device through Serial Peripheral Interface (SPI). Once this secret information is passed to the processing block, it is presented on a second display. The communication with both displays is executed through I_2C protocol. Before long, the system automatically logs out, preparing for a new log-in cycle. While performing their tasks the power consumptions of the legal ICs vary between $\sim 10mW$ and $\sim 50mW$.

5.1.2 The Monitoring Circuit

There are two main blocks in the power monitoring circuitry: (a) a monitoring block, and (b) five power sensors (one for every IC) (Fig. 4, Fig. 5). Inside the monitoring block an ATmega328P microcontroller has been used to perform the on-board data collection, processing and prediction tasks and to deliver the external communication functions. The microcontroller is linked to all power sensors via a multiplexer for communication using the I_2C protocol.

The action flow inside the monitoring block is illustrated in Fig. 6. After the previously trained machine learning model has been uploaded onto the monitoring block (step I), the monitoring block sets up all the power sensors on the PCB (step II). Next, in step III the monitoring block receives power readings from all the sensors and stores the data in a $[1 \times N]$ dimensional row vector. Then, in step IV any processing of the new point is performed (e.g. computation of moving averages). The resulting data point-vector is parsed through the ML model in step V. Finally, if the resulting prediction is $\hat{\mathbf{y}} = -1$, suggesting an HT-contaminated case, an HT intrusion alarm is raised, otherwise the microcontroller loops back to step III and begins the next iteration. Note that within one iteration the time-span between power readings from any two power sensors is assumed significantly smaller compared to the HT active time. Power sensing is executed through INA219 high-side current and power monitor chips. These chips have a built in 12-bit ADC and provide readings with 1mW resolution.

5.2 Data Collection and Feature Extraction

An important aspect of this research is that the data used to train, validate and test the machine learning models is collected from a real-life device, rather than being computer generated. To collect the data, the monitoring block on the prototype PCB device has been fitted with a micro-SD memory card to store the power consumption data when there is no HT device mounted on the PCB. Every power consumption data-point is generated as an N dimensional $[1 \times N]$ row-vector $\mathbf{X}_i$. It is stored in rows and columns, where every row corresponds to one iteration of data collection and every column corresponds to a specific IC (Table 1). During the data collection stage over 3×10^6 data point-vectors were recorded, which accounted for all the working modes of the PCB device. These data-points are distributed in several clusters in $\mathbb{R}^N$, their layout depending on the power consumption patterns of the actual PCB device under investigation. In this work, the number of these clusters has been 16. The vast majority of the data-points are scattered in the near vicinity of the centre of the cluster they belong. These points can be referred to as core points. The points that are closer to the peripheries of the clusters, referred to as edge points, are far lower in number and hence there is a high likelihood that a random selection of a fraction of all the points for model training purposes will miss most, if not all, of the actual edge points. This unfair selection will result in a skewed and consequently wrong power consumption pattern illusion in the training dataset. Given the finite number of points in Validation and Testing datasets, the simulation outcome might be reasonable. However since the training dataset will not be fully representative of the actual power consumption distribution pattern, ultimately the performance of the classification models in a real-life continuous run-time experiment will suffer, resulting in a significant drop of classification F1-score.

To solve this problem, an algorithm called Density-Based Spatial Clustering of Applications with Noise (DBSCAN) has been employed to identify and isolate edge and noise points. The noise points have later been removed, while, more importantly, all the edge points have been included in the Training dataset, along with randomly selected core points to fill in the core of the clusters. Thus the Training dataset consists of all the edge points, some of the core and almost none of the noise points (Fig 3). This technique helps speed up the model training process by allowing for smaller training datasets, as well as making the process more effective at finding a better boundary between expected HT-clean and deviant HT-contaminated data-points (Fig. 7a).

TABLE 1: Insertion of n-point Trojan occurrence on IC_3.

Index	IC_1	...	IC_3	...	IC_N	y
$\mathbf{X}_0$	$x_1^{(0)}$	...	$x_3^{(0)}$	...	$x_N^{(0)}$	1
...	...	...	...	...	...	...
$\mathbf{X}_i$	$x_1^{(i)}$	...	$x_3^{(i)}$	...	$x_N^{(i)}$	1
$\mathbf{X}_{i+1}$	$x_1^{(i+1)}$	...	$x_3^{(i+1)}+P_{HT1}$	...	$x_N^{(i+1)}$	-1
$\mathbf{X}_{i+2}$	$x_1^{(i+2)}$	...	$x_3^{(i+2)}+P_{HT2}$	...	$x_N^{(i+2)}$	-1
...	...	...	...	...	...	...
$\mathbf{X}_{i+n}$	$x_1^{(i+n)}$	...	$x_3^{(i+n)}+P_{HTn}$	...	$x_N^{(i+n)}$	-1
$\mathbf{X}_{i+n+1}$	$x_1^{(i+n+1)}$	...	$x_3^{(i+n+1)}$	...	$x_N^{(i+n+1)}$	1
...	...	...	...	...	...	...

TABLE 2: Parameters chosen for HT models.

HT Categories	Mean power, $\bar{P}_{HT}$	Active time, T_{HT}
	$5mW$ to $50mW$ step size $5mW$	$10pt$ to $100pt$ step size $10\ pt$

5.3 Hardware Trojan Modelling and Insertion

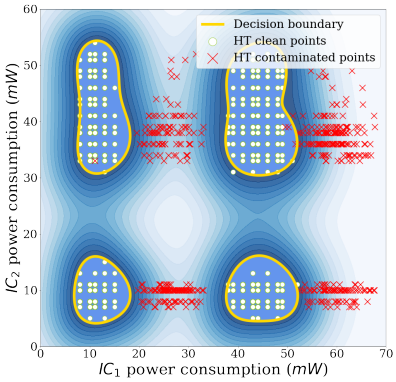
The insertion of hardware Trojans on the PCB is demonstrated by imitating the effects it would have caused, i.e. increased power consumption due to its operations. The HT model has been developed to replicate an extra triggerable component on the PCB, powered from the I/O pins of a legitimate IC (Fig 2b). The HT free data has been superimposed with the modelled HT's power consumption at random locations, thus effectively replicating a realistic model of an implanted triggerable HT. In this model the power consumption of an active HT, P_{HT}, follows a Gaussian normal probability density function (13) and hence the HT model has three parameters, HT's mean power consumption $\bar{P}_{HT}$, the standard deviation σ and the active time span T_{HT}. The model generates T_{HT} number of HT device power consumption values, which are normally distributed around $\bar{P}_{HT}$ with a standard deviation set to σ. Note that the duration of the HT's active time is measured in HT-points (subsection 4.4).

The HT with a triggering mechanism has a state of active payload delivery after being triggered (state 1) and an idle state (state 2). The power consumption of the Trojan device in the first state is normally distributed around $\bar{P}_{HT}$, with a standard deviation of σ. In state 2, the power consumption is assumed to be $\sim 0mW$, i.e. too low to be detected by the sensors, mimicking HT's power down or idle modes. Note that the proposed method is invariant of the payload of the HT, as long as it consumes extra power. An abstraction of a simulated HT's power consumption data group for a single activation occurrence can be presented as follows

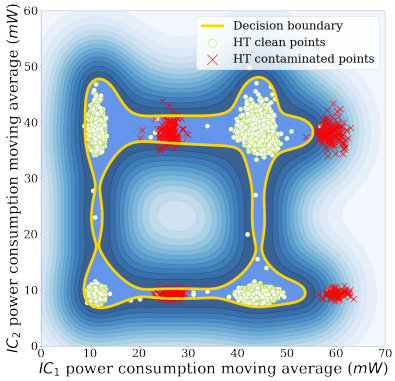
$$\mathbf{P_{HT}} = \{P_{HT1}, P_{HT2}, P_{HT3}, ..., P_{HTn}\}, \qquad (15)$$

where n is the number of HT-points T_{HT}.

Several generated HT data groups similar to (15) have been randomly added to the validation and test datasets in such a way that the ratio of the HT-contaminated vectors to HT-clean vectors is 1:1 (i.e. 50% HT-clean and 50% HT-contaminated). While adding the HTs, a random choice has been made of both the columns (i.e. ICs on the PCB) and the rows (i.e. HT activation points in time), while assuring that no overlapping of rows takes place. Table 1 illustrates how the HT-clean data-frame is modified to become an HT-contaminated data-frame. In this particular example an n-point long HT occurrence has been added in column IC_3, after instance X_i point in time, i.e. an HT activation occurrence on IC number 3 at time *i+1*. In Table 1 it can be seen how the power consumption of the HT device (in red) is added to that of the legitimate IC. In this work, 100 categories of Trojan devices have been modelled and injected into the original HT-clean data collected from the prototype PCB. Table 2 provides ranges for the HT's mean power consumption $\bar{P}_{HT} \in \{[5:50], \text{step} = 5\ mW\}$ and active-time HT-points $T_{HT} \in \{[10:100], \text{step} = 10\ point\}$ (described in Section 4). The hundred HT categories were generated by taking combinations of the provided parameter values, while the standard deviation is $\sigma = 1$ for all.



(a) Unmodified readings from IC_1 and IC_2



(b) Averaged readings from IC_1 and IC_2

Fig. 7: Two dimensional cross sections of the four dimensional dataset and the One-Class SVM decision boundary.

5.4 Feature Engineering

In an effort to reduce the noise in the sensor readings and improve classification outcomes, feature engineering has been applied. For every column in the dataset, a new column has been appended, where the data was filled in with the moving averages of the original values. This way, using the moving average, the power reading row-vectors X_i (Table 1) are filtered by taking the column-wise average with their respective previous $M-1$ values,

$$X_i = \frac{1}{M}\sum_{j=0}^{M-1} X_{i-j}, \quad \text{for} \quad \forall\, i \in \mathbb{N} \quad \text{s.t.} \quad i \geq M, \qquad (16)$$

where M is the number of averaged points. The optimal value for parameter M can be estimated empirically. In this work M has been set to 5. After applying feature engineering the dataset matrix with N columns becomes one with $2N$ columns, changing the dimension of every data point-vector from $[1 \times N]$ to $[1 \times 2N]$.

6 Model Training and Simulation Results

In order to create reliable models, supervised ML algorithms should use sufficiently large datasets which effectively capture the underlying data distribution patterns. These trained models are later utilised for either regression or classification problems [37]. The end-to-end process consists of several steps. Generally the first step would be initial data manipulation including data cleaning and feature selection where the relevant information-bearing features are extracted from the raw data. Next, feature engineering may be applied where the data is manipulated in some way which increases its predictive power. The resulting sample data is divided into Training, Validation and Test datasets and stored for later procedures. The second step is selecting and executing the appropriate ML algorithms to obtain the end models, fitted on the Training dataset and optimised on the Validation dataset. Here, among other procedures, grid-search optimisation of algorithm hyperparameters is carried out in conjunction with K-fold validation to determine the best model. In the final evaluation stage, the performance of this model is assessed on the Test dataset. The optimal model is chosen, which will be used to make run-time predictions for new data on the PCB.

In the problem set out in this work the training data consists of only one class, the HT-clean class. Having collected all the necessary data, two machine learning classification algorithms have been trained: One-Class Support Vector Machine and Local Outlier Factor. Although the training parameters vary for the algorithms, the task at hand is the same for both of them - find the best possible decision function, which will maximise the number of correctly predicted HT-clean and HT-contaminated data-points, while avoiding any unnecessary overfitting to the training data. This way the model should be able to best predict if a new incoming data-point belongs to the HT-clean class or has a significant deviation and so can be safely classified as an anomaly. All the anomaly points are considered as HT-contaminated.

As an example consider a PCB with only two ICs and their respective power sensors. This sets the dimensions of the data point-vector at $N = 2$. Then, with the feature engineering applied, the final dimension of the data point-vector becomes $2 \cdot N = 4$. Such data point-vectors reside in a four dimensional space, therefore the decision boundary is a four dimensional surface. For visualisation purposes two double-feature cross sections of the four dimensional shape are shown in Fig. 7. The two feature axes in Fig. 7a represent power consumption readings received from the two power sensors on IC_1 and IC_2. The axes in Fig. 7b, on the other hand, are the feature engineering generated moving averages of the same power readings. Shown in white dots are the HT-clean data point-vectors, while the red crosses represent the HT-contaminated data point-vectors. The golden bands illustrated in the diagrams are the 2 dimensional cross sections of the determined 4 dimensional decision boundary (a hypersphere in the kernel space). Any new point enclosed by the golden band will be classified as an HT-clean vector, while the ones outside will be classified as HT-contaminated vectors. The HT-to-IC power consumption ratio has been considered under variability of the power consumption points of legal ICs. The more compact are power consumption clusters of legal ICs (Fig. 7a), the lower-power HTs will become detectable since they will not be concealed within the boundaries of legal power consumption. Shown in Fig. 7a, notice how some of the HT affected red points remain within the bold yellow decision boundaries because of the width of the respective cluster.

Algorithm 1 One-Class SVM run-time monitoring.

1: ***Input:*** *M, N, SVM(x) → Decision function*
2: ***Initialise:*** $k = 0$
3: ***loop*** $k = k + 1$
4: ***for*** *i=1,2,3,..., N* ***do*** ▷ *Collect sensor data*
5: $\boldsymbol{X}[k][i] = Sensor^{(i)}$
6: ***end for***
7: ***for*** *j=1,2,3,..., N* ***do*** ▷ *Feature engineering*
8: $\boldsymbol{X}[k][N+j] = \dfrac{\boldsymbol{X}[k][j] + ... + \boldsymbol{X}[k-M][j]}{M}$
9: ***end for***
10: $\hat{y} = SVM(\boldsymbol{X}[k])$ ▷ *Make prediction*
11: ***return*** $\hat{y}$
12: ***end loop***

One-Class SVM algorithm has been chosen to determine the decision boundary in Fig. 7. Once the best model has been trained the decision function is uploaded onto the monitoring block on the prototype device to perform run-time monitoring of the PCB. Pseudo-code for the online monitoring steps is provided in Algorithm 1.

Classification metrics

Precision and Recall rates are defined by the following formulae:

$$\text{Precision} = \frac{TP}{TP + FP}, \quad \text{Recall} = \frac{TP}{TP + FN}, \tag{17}$$

where TP, FP, and FN are the numbers of true positive, false positive and false negative predictions respectively.

Depending on the use case scenarios of a particular ML model, priority may be given to either high Precision or high Recall rate. In this work Precision and Recall have been treated with equal weights and the ML model has been optimised for the highest balanced F1-score. F1-score is an imbalanced-dataset aware classification metric and the closer it is to 100% the less misclassifications there are. The balanced F1-score can be calculated using the following formula:

$$\text{F1-score} = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{18}$$

6.1 One-Class SVM

Parameter Selection

The bandwidth parameter γ in (6) and the regularization parameter ν in (3) and (5) are the main parameters in One-Class SVM and play key roles in the quality of the resulting

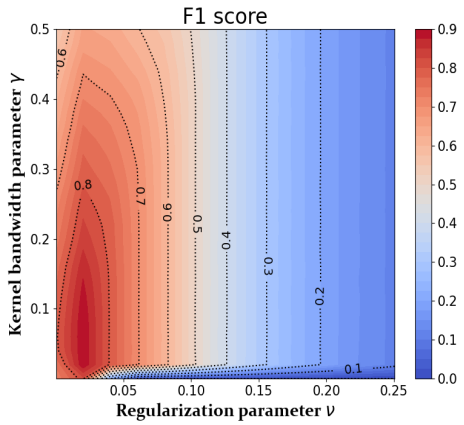


Fig. 8: One-Class SVM classification F1-score contour map.

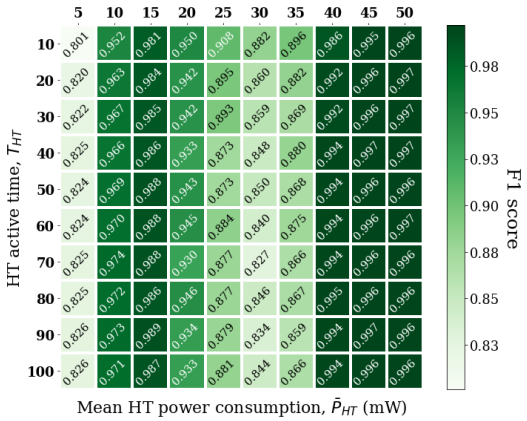


Fig. 9: One-Class SVM classification results for all Trojans.

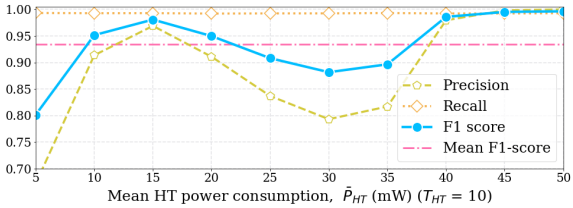


Fig. 10: One-Class SVM classification results for all Trojan power consumption values ($T_{HT} = 10$).

model [38]. Smaller ν values leave fewer training samples on the origin side of the hyperplane. If ν approaches to 0, the upper bounds on the Lagrange multipliers α_i tends to infinity in (5). The penalty term $\sum \xi_i$ in the objective function (3) vanishes, which means that the hyperplane found in the feature space separates almost all the training data from the origin. If ν approaches to 1, the algorithm leaves more data-points on the origin side of the hyperplane, when $\nu = 1$ almost all the data-points are classified as anomalies.

Training

To reduce the degree of uncertainty in the results, ten-fold validation has been carried out, while running a greedy grid-search over the hyperparameters ν and γ. The moving average hyperparameter M has been set to five during the feature engineering stage. Next, the resulting One-Class SVM model has been tested on the Test set to check if it generalises well on previously unseen data. While the Training dataset contains HT-clean class only, the Validation and Test datasets are composed of 50% HT-clean and 50% HT-contaminated points, including all 100 HT categories from Table 2.

Results

The contour map shown in Fig. 8 illustrates F1-score dependency of One-Class SVM regularization hyperparameter ν and kernel bandwidth γ. It can be seen in the figure that F1-score of the trained One-Class SVM model peaks in the lower-left-corner of the (ν, γ) plane. In fact a greedy grid-search for the best hyperparameters has returned $\{\nu = 0.0015, \gamma = 0.016\}$. Using this ν–γ pair the ML model reaches prediction F1-scores 0.993, 0.968 and 0.968 on Training, Validation and Testing datasets respectively.

The classification results per HT category are shown in Fig. 9. It can be seen that, indeed, HTs with lower power consumption are harder to detect. Comparing HT categories, when $\bar{P}_{HT} \geq 40mW$ the F1-score reaches above 99%, while when $\bar{P}_{HT} = 5mW$ the detection rates peak at F1-score $= 82.6\%$. Trojan activation times, on the other hand, do not seem to have such a dramatic impact on detectability as HT power consumption does. It can also be seen that there is a drop in F1-scores around $\bar{P}_{HT} = 30mW$. This is a direct result of the blind zone effect described in Fig. 2a, when the HT's added power consumption shifts the red curve to the right by exactly the amount necessary to overlay the left red peak with the right green peak. The graph presented in Fig. 10 helps visualise the simulation results, while also showing the Precision and Recall rates and the average of all F1-scores for $T_{HT} = 10$ time units.

6.2 Local Outlier Factor

This algorithm takes its name after the anomaly score of each sample, called local outlier factor defined in (11). This is a measure of the local deviation of density of a single data-point in comparison with that of its neighbours. The locality aspect plays a role in that the anomaly score depends on the degree of isolation of the given data-point with respect to its own neighbourhood. The distance from k-nearest neighbours (9) is used to estimate the local density. A comparison of the local density of a sample and the local densities of its neighbours can reveal the HT-contaminated points, which would have significantly lower density.

Parameter Selection

The main hyperparameters in LOF that control the quality of the classification model are the number of neighbours k and the contamination factor c, where k is used to decide method calculated k-distance, and c is used to control the precision of training. If k is set larger than the number of available data-points, all the points will be used. The contamination proportion of the training dataset c is the ratio of the number of deviant points in the data and that of the expected (HT-clean) points.

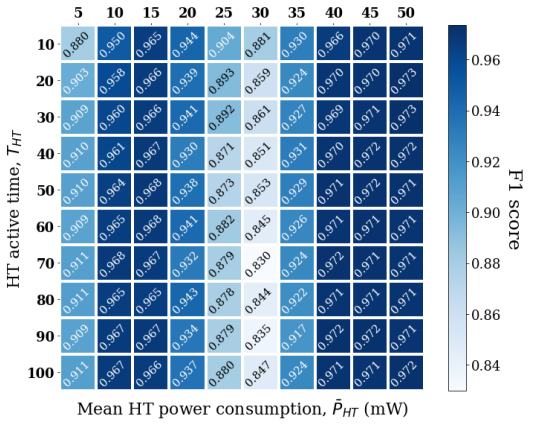


Fig. 11: LOF classification results for all Trojans.

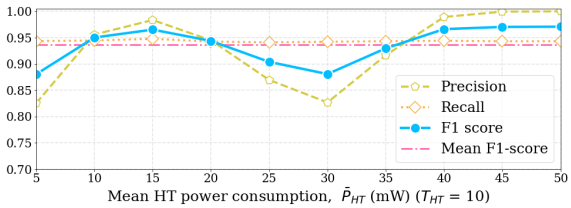


Fig. 12: LOF classification results for all Trojan power consumption values ($T_{HT} = 10$).

Training

Similar to the previous algorithm, LOF too has been trained using 10-fold validation with the same datasets as used with One-Class SVM. Here too the 100 HT parameter combinations have been chosen from Table 2. The moving average hyperparameter M again has been set to five during feature engineering. The model has been optimised for the highest F1-score by running a greedy grid-search over the hyperparameters k and c.

Results

The classification results for LOF are shown in Fig. 11. Comparing values from the first columns of Fig. 9 and Fig. 11 it can be seen that in the lower spectrum of HT's power consumption LOF algorithm outperforms One-Class SVM on average by 8.5% when $\bar{P}_{HT} = 5mW$. On the other hand, for $\bar{P}_{HT} \geq 5mW$ One-Class SVM proves to be a more robust algorithm for HT detection, eventually reaching F1-scores above 99.7% for $\bar{P}_{HT} = 50mW$, while LOF reached its maximum at 97.1%. Regarding HT active time length, as before T_{HT} has a more subtle effect on the detectability. That being said, when $\bar{P}_{HT} = 5mW$, F1-score for $T_{HT} = 100$ is 3.1% higher than that for $T_{HT} = 10$. In Fig. 11 it can also be seen that for LOF too the blind zone effect results in a drop in F1-scores around $\bar{P}_{HT} = 30mW$. The graph in Fig. 12 ($T_{HT} = 10$) clearly shows this drop in F1-scores when $20mW \leq \bar{P}_{HT} \leq 40mW$, while also providing values of the Precision, Recall and average of all F1-scores.

7 Experimental Results

To verify the effectiveness of the proposed methodology and validate the simulation results, several experiments have been performed on the PCB prototype shown in Fig. 5, using

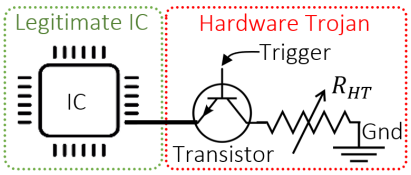


Fig. 13: Schematic diagram of the deployed HT.

Fig. 14: Comparison of experiments and simulations.

One-Class SVM as the classification algorithm. The reason why One-Class SVM is chosen for the experiment is that it generally outperformed LOF during the simulations. To further justify the choice, One-Class SVM is an eager-learner algorithm, requiring less computational load at prediction phase. In contrast, LOF is a lazy-learner [39] and defers the majority of computation to prediction phase. To be feasible for production, an on-board real-time monitoring ML model should have small footprint on computing and memory resources.

First, the trained One-Class SVM model has been transferred to the monitoring block on the prototype PCB. That includes coding the decision function (8) and uploading coordinates of all the support vectors $\mathbf{x}_i$, the α_i coefficients for every support vector, as well as the free term ρ in the decision function. Next, a triggerable HT device has been added to the PCB using a transistor and a range of resistors with varying resistances (Fig. 13). The trigger sequences have been recorded in sync with the power consumption data on the PCB for later model evaluation analysis. The HT resistance values have been carefully chosen so that their power consumptions P_{HT} lie within the ranges $5mW$–$10mW$, $10mW$–$15mW$ or $15mW$–$20mW$. The duration of HT's active times T_{HT} are one of {20, 50, 100}. Shown in Fig. 14 are the prediction F1-scores for all nine combinations of P_{HT} and T_{HT}. The figure compares One-Class SVM results from the experiment with those from One-Class SVM and LOF from simulations.

Memory usage

Given that the dataset has dimensionality of 10, using 16 bit unsigned integer encoding, a One-Class SVM model with 1258 support vectors will require memory capacity of about 27KB. To show the memory capacity requirement calculation consider for example a model with 500 support vectors (500×10 integers), 500 support vector coefficients (500 integers) and a single free term ρ will need to store in memory a total of about 10KBs of parameters.

$$\text{Memory} = \frac{(500 \cdot 10 + 500 + 1) \cdot 16}{8} \approx 10KB \qquad (19)$$

8 Discussion

The detectability of HTs can be affected by issues overlooked in this paper. Comparing Fig. 10 and Fig. 12, it can be seen that the two algorithms have different behaviours at the two ends of the considered power consumption spectrum. While One-Class SVM algorithm has better performance at the upper end of the spectrum (*F1-score = 0.997 at 50mW*), LOF algorithm shows superior results at the lower end (*F1-score = 0.88 at 5mW*). For both algorithms, however, the HT's mean power consumption of *5mW* (*standard deviation = 1mW*) is the region where the performance begins to decay. One reason for such drop in performance is that the resolution of the power sensors used in this research is at *1mW*, which is 20% of the HT power consumption. Improving the resolution of the sensors on the prototype will reduce the noise-to-signal ratio, improving data quality and thus the results.

Variabilities such as voltage, process and temperature variation or aging of silicon devices are another set of factors which can have a negative impact on the performance metrics of the proposed methodology. Thus, incorporating these phenomena in future assessments is important. While voltage and temperature variations can become part of the current model as two more feature columns in the data, addressing process variation and aging may require a broader approach. For example, process variation can be addressed by collecting data from a wider range of PCB prototypes, and quantification of aging effect is studied in [40].

In contrast to the prototype used in this study, PCBs may have a larger setup with more ICs on board. Hence, a discussion of possible effects on the proposed methodology is due. From the methodology standpoint, performance degradation is expected to happen for two main reasons, including slower data collection iterations according to Eq. (14) given the increased number of power sensors, and slower computation due to higher data dimensionality. However, this can be alleviated by deploying a more powerful microcontroller in the monitoring block. From the ML algorithm standpoint, a phenomenon commonly referred to as the curse of dimensionality could take effect when increasing the number of dimensions in data, i.e. the number of ICs on the PCB. ML algorithms based on distance measures tend to fail when the number of dimensions in the data is very high. In the Euclidean space, this happens due to noise-induced reduction of the ratio between distances of datapoints belonging to different classes and points from the same class, i.e. numerically all points appear closer in the high dimensional space. This effect results in a declined performance in algorithms such as One-Class SVM. Note, it typically takes place after several hundred dimensions, which should be plenty for the PCB application scenario. Furthermore, this problem could in principle be mitigated with dimensionality reduction techniques, e.g. Principal Component Analysis.

In some cases, a single PCB can have several power distribution networks (PDNs) with different electrical characteristics. Such instances should not affect the conclusion drawn from this work. This is because the power sensors are measuring the power consumption on the end nodes of the PDNs, regardless of the specific characteristics such as voltage value. From the methodology's standpoint these could be regarded as separate entities. The model does not require awareness of the PDN layout of the PCB and, therefore, it can be trained for such scenarios.

Ultimately, given the wide spectrum of possible Trojan attacks, a comprehensive solution should comprise of a combination of methodologies targeting a subgroup of all possible Trojans. In this paper, the group of active Trojan implant ICs has been addressed, which consume a reasonable amount of power on the PCB. The group of ultra-low power passive Trojans, e.g. resistors and capacitors, should be addressed by other techniques such as visual inspection. For example, in a recently published paper [41] a visual inspection technique has been proposed using optical images to highlight the sections on PCB with a high likelihood of harbouring a surface mount Trojan component.

9 Conclusion

In this paper a combination of power analysis with machine learning (ML) algorithms has been implemented with the purpose of detecting Hardware Trojan (HT) components on a Printed Circuit Board (PCB). The monitoring circuit architecture proposed in [15] is further developed with the introduction of ML methods to detect stealthier HTs powered from legitimate chips on a PCB.

Two ML algorithms have been applied to the power consumption data collected from a purpose built PCB prototype (Fig. 5). Later this dataset has been injected with one hundred categories of HTs from Table 2 with power consumptions between $5mW$ and $50mW$. The resulting dataset has been divided to three datasets (Training, Validation and Testing) to train One-Class Support Vector Machine (SVM) and Local Outlier Factor (LOF) one class classification algorithms. Simulations for both algorithms returned classification results reaching F1-scores above 99.5% and 97% for One-Class SVM and LOF respectively, given the HT's mean power consumption is $\bar{P}_{HT} \geq 45mW$. The analyses also showed that the length of HT active time T_{HT}, as defined in Section 4, does not have significant impact on detectability.

Further, the One-Class SVM model has been uploaded to the monitoring block of the PCB. The simulation results have been validated through real-life experiments carried out on the prototype PCB. Comparison between simulation and experimental results has been presented for nine Hardware Trojan categories, their power consumption ranging from $5mW$ to $20mW$. Further, the One-Class SVM model is low-cost and requires only 27KB memory to operate.

References

[1] A. Iyengar and S. Ghosh, *Hardware Trojans and Piracy of PCBs*. Springer International Publishing, 2017, pp. 125–145.

[2] J. Robertson and M. Riley, "The Big Hack: How China Used a Tiny Chip to Infiltrate U.S. Companies," www.bloomberg.com/news/features/2018-10-04/the-big-hack-how-china-used-a-tiny-chip-to-infiltrate-america-s-top-companies, 2018.

[3] D. Mehta, H. Lu, O. P. Paradis *et al.*, "The Big Hack Explained: Detection and Prevention of PCB Supply Chain Implants," *J. Emerg. Technol. Comput. Syst.*, vol. 16, no. 4, 2020.

[4] S. Paley, T. Hoque, and S. Bhunia, "Active protection against PCB physical tampering," *Int. Symp. on Quality Electronic Design*, pp. 356–361, 2016.

[5] M. Azhagan, D. Mehta, H. Lu *et al.*, "A review on automatic bill of material generation and visual inspection on PCBs," in *Int. Symp. for Testing and Failure Analysis*, 2019, pp. 256–265.

[6] P. S. Malge and R. S. Nadaf, "PCB defect detection, classification and localization using mathematical morphology and image processing tools," *Int. J. of Computer Applications*, vol. 87, no. 9, pp. 40–45, 2014.

[7] S. Tang, F. He, X. Huang, and J. Yang, "Online PCB defect detector on a new PCB defect dataset," *ArXiv e-prints*, 2019.

[8] H. H. Wu, X. M. Zhang, and S. L. Hong, "A visual inspection system for surface mounted components based on color features," in *Int. Conf. on Information and Automation*, 2009, pp. 571–576.

[9] A. Crispin and V. Rankov, "Automated inspection of PCB components using a genetic algorithm template-matching approach," *Int. J. Adv. Manuf. Technol.*, vol. 35, p. 293–300, 2007.

[10] C. Szymanski and M. R. Stemmer, "Automated PCB inspection in small series production based on SIFT algorithm," in *Int. Symp. on Industrial Electronics (ISIE)*, 2015, pp. 594–599.

[11] R. Smith, "An Overview of the Tesseract OCR Engine," in *Int. Conf. on Document Analysis and Recognition*, vol. 2, 2007, pp. 629–633.

[12] M. Moganti, F. Ercal, C. H. Dagli *et al.*, "Automatic PCB inspection algorithms: A survey," *Computer Vision and Image Understanding*, vol. 63, no. 2, pp. 287–313, 1996.

[13] W.-C. Wang, S.-L. Chen *et al.*, "A Machine Vision Based Automatic Optical Inspection System for Measuring Drilling Quality of Printed Circuit Boards," *IEEE Access*, vol. 5, pp. 10 817–10 833, 2017.

[14] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Computer Vision – ECCV 2006*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 404–417.

[15] G. Piliposyan, S. Khursheed, and D. Rossi, "Hardware Trojan Detection on a PCB Through Differential Power Monitoring," *IEEE Transactions on Emerging Topics in Computing*, vol. 10, no. 2, pp. 740–751, 2022.

[16] M. Tehranipoor, "New directions in hardware security," in *Int. Conf. on VLSI Design and Int. Conf. on Embedded Systems (VLSID)*, 2016, pp. 50–52.

[17] R. Elnaggar and K. Chakrabarty, "Machine Learning for Hardware Security: Opportunities and Risks," *Journal of Electronic Testing*, vol. 34, pp. 183–201, 2018.

[18] K. G. Liakos, G. K. Georgakilas, S. Moustakidis *et al.*, "Machine learning for hardware trojan detection: A review," in *Panhellenic Conf. on Electronics Telecommunications (PACET)*, 2019, pp. 1–6.

[19] J. Li, L. Ni, J. Chen, and E. Zhou, "A novel hardware Trojan detection based on BP neural network," in *IEEE Int. Conf. on Computer and Communications (ICCC)*, 2016, pp. 2790–2794.

[20] L. Ni, J. Li, S. Lin, and D. Xin, "A method of noise optimization for Hardware Trojans detection based on BP neural network," in *IEEE Int. Conf. on Computer and Communications (ICCC)*, 2016, pp. 2800–2804.

[21] C. H. Kok, C. Y. Ooi, M. Moghbel *et al.*, "Classification of Trojan Nets Based on SCOAP Values using Supervised Learning," in *IEEE Int. Symp. on Circuits and Systems (ISCAS)*, 2019, pp. 1–5.

[22] Z. Huang, Q. Wang, Y. Chen, and X. Jiang, "A Survey on Machine Learning Against Hardware Trojan Attacks: Recent Advances and Challenges," *IEEE Access*, vol. 8, pp. 10 796–10 826, 2020.

[23] H. Pearce, V. R. Surabhi *et al.*, "Detecting Hardware Trojans in PCBs Using Side Channel Loopbacks," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 30, no. 7, 2022.

[24] T. Mosavirik, P. Schaumont, and S. Tajik, "ImpedanceVerif: On-Chip Impedance Sensing for System-Level Tampering Detection," Cryptology ePrint Archive, Paper 2022/946, 2022.

[25] T. Mosavirik, F. Ganji *et al.*, "ScatterVerif: Verification of Electronic Boards Using Reflection Response of Power Distribution Network," *J. Emerg. Technol. Comput. Syst.*, vol. 18, no. 4, 2022.

[26] H. Zhu, H. Shan, D. Sullivan *et al.*, "PDNPulse: Sensing PCB Anomaly with the Intrinsic Power Delivery Network," *ArXiv e-prints*, 2022.

[27] D. Fujimoto, S. Nin, Y.-I. Hayashi, N. Miura, M. Nagata, and T. Matsumoto, "A Demonstration of a HT-Detection Method Based on Impedance Measurements of the Wiring Around ICs," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 65, no. 10, pp. 1320–1324, 2018.

[28] F. Pedregosa, G. Varoquaux, A. Gramfort *et al.*, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[29] J. A. K. Suykens and J. Vandewalle, "Least Squares Support Vector Machine Classifiers," *Neural Process. Lett.*, vol. 9, no. 3, p. 293–300, 1999.

[30] B. Schölkopf, J. C. Platt, J. Shawe-Taylor *et al.*, "Estimating the Support of a High-Dimensional Distribution," *Neural Computation*, vol. 13, no. 7, pp. 1443–1471, 2001.

[31] B. Schölkopf, R. C. Williamson, A. J. Smola *et al.*, "Support vector method for novelty detection," in *NIPS*, vol. 12. Citeseer, 1999, pp. 582–588.

[32] B. Schölkopf, A. J. Smola, F. Bach *et al.*, *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[33] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying Density-Based Local Outliers," in *ACM SIGMOD Int. Conf. on Management of Data*, 2000, p. 93–104.

[34] C. Dong, Y. Liu, J. Chen *et al.*, "An Unsupervised Detection Approach for Hardware Trojans," *IEEE Access*, vol. 8, pp. 158 169–158 183, 2020.

[35] R. Domingues, M. Filippone, P. Michiardi, and J. Zouaoui, "A comparative evaluation of outlier detection algorithms: Experiments and analyses," *Pattern Recognition*, vol. 74, pp. 406–421, 2018.

[36] F. K. Lodhi, S. R. Hasan, O. Hasan, and F. Awwadl, "Power profiling of microcontroller's instruction set for runtime hardware Trojans detection without golden circuit models," in *Design, Automation Test in Europe Conf. Exhibition (DATE)*, 2017, pp. 294–297.

[37] E. Alpaydin, *Introduction to machine learning*. MIT press, 2020.

[38] Z. Ghafoori, S. M. Erfani, S. Rajasegarar *et al.*, "Efficient unsupervised parameter estimation for one-class support vector machines," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 10, pp. 5057–5070, 2018.

[39] K. T. Divya and N. S. Kumaran, "Improved outlier detection using classic KNN algorithm," in *Int. J. of Engineering and Technology (IRJET)*, vol. 3, no. 12, 2016, pp. 892–898.

[40] A. L. H. Martínez, S. Khursheed, T. Alnuayri, and D. Rossi, "Online Remaining Useful Lifetime Prediction Using Support Vector Regression," *IEEE Transactions on Emerging Topics in Computing*, vol. 10, no. 3, pp. 1546–1557, 2022.

[41] G. Piliposyan and S. Khursheed, "Computer Vision for Hardware Trojan Detection on a PCB Using Siamese Neural Network," in *IEEE Int. Conf. PAINE*, 2022.

Gor Piliposyan received the MSc degree in Radio-Physics and Electronics from Yerevan State University, Armenia. He is currently pursuing the Ph.D. degree with the Department of Electrical Engineering and Electronics, University of Liverpool, Liverpool, U.K. His research interests are in hardware oriented security of low-power designs and machine learning.

Saqib Khursheed received the Ph.D. degree in electronics and electrical engineering from the University of Southampton, U.K. He is currently a Lecturer (Assistant Professor) with the Department of Electrical Engineering and Electronics, University of Liverpool, U.K. He is interested in all issues related to hardware oriented security, reliability, and test of low-power, high performance designs, and 3-D integrated circuits. He has authored more than 50 papers in internationally leading journals and conferences in these areas.