An Unsupervised Detection Approach for Hardware Trojans

CHEN DONG, YULIN LIU, JINGHUI CHEN, XIMENG LIU, WENZHONG GUO, AND YUZHONG CHEN

1College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350116, China
2Key Laboratory of Information Security of Network Systems, Fuzhou University, Fuzhou 350116, China
3Fujian Provincial Key Laboratory of Network Computing and Intelligent Information Processing, Fuzhou University, Fuzhou 350116, China

Corresponding authors: Wenzhong Guo (fzugwz@163.com) and Yuzhong Chen (yzchen@fzu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61672159, Grant 61872091, Grant U18042631, and Grant 61702105; in part by the Science Foundation of the Fujian Province, China, under Grant 2018J01793 and Grant 2018J01800; in part by the Foundation of the Education Department of Fujian Province, China, under Grant JAT190025; and in part by the Fujian Province High-Level Talents and Young Talents Visiting and Training Subsidy Program.

ABSTRACT

With the booming development of the cyber-physical system, human society is much more dependent on information technology. Unfortunately, like software, hardware is not trusted at all, due to so many third parties involved in the separated integrated circuit’s (IC) design and manufacturing stages for the high profit. The malicious circuits (named Hardware Trojans) can be implanted during any stage of the ICs’ design and manufacturing process. However, the existing pre-silicon approaches based on machine learning theory have good performance, they all belong to supervised learning methods, which have a key prerequisite that is numerous already known information. Meanwhile, hardware Trojans are even more unimaginable because today’s ICs are becoming more complicated. The known information is even harder to gain. Furthermore, the training process for supervised learning methods tends to be time-consuming and generally requires a huge amount of balanced training data. Therefore, this paper proposes an unsupervised hardware Trojans detection approach by combined the principal component analysis (PCA) and local outlier factor (LOF) algorithm, called PL-HTD. We firstly visualize the distribution features of normal nets and Trojan nets, and then reveal the differences between the two types of nets to reduce the dimension of the feature set. According to the outliers of each net, the abnormal nets are selected and verified by professionals later to confirm whether it is a true Trojan relative to the host circuit to realize the detection. The experiments show that the proposed method can detect hardware Trojans effectively and reduce the cost of manual secondary detection. For the Trust-HUB benchmarks, the PL-HTD achieves up to 73.08% TPR and 97.52% average TNR, moreover, it achieves average 96.00% accuracy, which shows the feasibility and efficiency of hardware Trojans detecting by employing a method without the guidance of class label information.

INDEX TERMS

Hardware security, hardware Trojan detection, integrated circuit, unsupervised machine learning, LOF.

I. INTRODUCTION

The fifth-generation mobile communication (5G) system emerges as times require in order to cope with the future of mobile data traffic explosive growth, the connection of the massive device, and the continuous emergence of various new services. With the booming development of the cyber-physical system, which has made our lives smarter and more convenient. Internet of Things (IoT) through intelligence, identification technology, and communications sensing technology [1] is widely used in converged networks. The integrated circuit (IC) is usually called chip, which is the core component of electronic devices, even though IoT devices. IoT technology arouses widespread attention in various areas, such as the governments, academia, and enterprises of various countries due to its vast application prospect [2].
People’s awareness of information security is arousing, but most of the existing relevant studies focus on the security issues at the software while ignoring the security threats hidden in the underlying hardware. Unfortunately, like software, the hardware still can not be trusted. The risks of the chips should not be underestimated, which are the real physical basis of cyberspace security.

At present, the hidden risk of the integrated circuit mainly comes from the malicious circuits implanted inside which are called Hardware Trojan (HT in short) [3]. In recent years, there are some reports like the following, the defense science council reported in March 2017 that U.S. military weapons systems might have been injected with HTs [4]. It’s worth mentioning that due to the high profit drive, so many unknown trusted third parties are involved in the separated design and manufacturing process. This mode results in each stage of the chip manufacturing process may be implanted malicious Trojans. Especially in the physical design process of the very large-scale integrated circuit (VLSI), such as partitioning and floorplanning [5], [6], the malicious vendors can insert HTs into IC products easily just by modifying the design file of IC [7]. From register-transfer level (RTL) code synthesis to gate-level netlist design, to chip integration packaging, each link may be embedded in the HTs [8], as is shown in Fig. 1. Above all, in the design stage, both IP core suppliers and designers need to use EDA software provided by EDA suppliers but the reliability of EDA software cannot be guaranteed. At the same time, the designer also cannot guarantee the reliability of the third-party IP core used and the layout file. Then, when the design layout is put into production, the safety of ICs needs to be paid great attention due to the involvement of untrusted employees and the need for third-party vendors. With the progress of the IC technology, modern VLSI designs have shifted toward the system-on-chip (SoC) paradigm, the density of IC has also greatly increased, and more and more obstacles appear in the process of physical design [9]. In addition, the design of hardware Trojan is becoming more unimaginable. Obviously, how to detect HTs is a rather critical and urgent task for researchers.

The existing HT detecting methods can be classified into two main types, pre-silicon and post-silicon. The post-silicon methods include side-channel analysis, reverse analysis and function testing, and so on, these methods are effective in the situation that the chip-scale is limited. Nevertheless, the reality of the cyber-physical system is that most of the chips belong to VLSI or SoC, etc. The post-silicon methods are not applicable when the IC is integrated highly. The pre-silicon methods are good choices, and with high integration, the machine-learning-based pre-silicon methods can express better performance [10]–[12]. The existing machine-learning-based HT detecting methods are all supervised. For the case with training samples, it seems that it is always better to use supervised than unsupervised. While the reality is always brutal, the unsupervised has to be taken in certain situations, due to the following three reasons.

1) In practice, high-quality labeled datasets are difficult to obtain. Not all samples are labeled or labeling all samples could be quite expensive, thus supervised learning is not feasible in the cases with unlabeled samples [13]. Naturally, learning beneficial patterns and feature representations from this type of data is of great importance [14].

2) For a supervised categorization, different experts may disagree when deciding under which category to categorize a given gate-level netlist. Besides, due to the rapidly increasing amounts of gate-level netlists, the dynamic nature of most netlists makes it difficult to manually pre-define the categories and the subjectivity in assigning netlists to categories has lead us to believe that by nature Trojan detection should be an unsupervised task rather than a supervised one.

3) Feature data may have potential internal connections since not all feature data are independent of each other, so there exist some offsets. For supervised learning, large offsets are likely to bring great noise to the classifier, which weakens the usability, reliability, and interpretability of classification results.

In addition, the detection method of abnormal objects in supervised learning can only detect the existing or similar abnormal objects in the training set, and the number of samples in the training set is generally small. That is, the detection results are questionable for other abnormal objects that do not appear in the training set [15]. We hope that unsupervised approach can be employed to give feedback to the human
indexers to enhance the task of pre-defining categories in gate-level netlists.

In this paper, we propose a HTs detecting method call PL-HTD based on unsupervised machine learning. The extracted features are selected using the scoring mechanism of the eXtreme Gradient Boosting (XGBoost) algorithm. Without labels, the low-dimensional netlist features were extracted and classified into predictive labels. Specifically, our technical contributions in this paper are as follows:

- **Application of unsupervised learning in detecting HTs.** Unsupervised machine learning algorithm trains a machine to discover hidden structures and patterns from unlabeled data without target variables. The result verifies the effectiveness of the detection method of hardware Trojans in gate-level netlists based on unsupervised machine learning.

- **Reduction of feature set dimension.** The feature data sets are dimensionally reduced. We eliminate the redundant information between the original features and construct the new features that are independent of each other, thus reducing the computational complexity and the cost of hardware Trojan detection in gate-level netlists.

- **Calculation of local outlier factor.** For the relativity of the hardware Trojan circuit, that is, if only the Trojan circuit is considered, it is a normal circuit for realizing a certain function, but it is called a Trojan circuit relative to the host netlist. We visualize the distribution features of normal nets and Trojan nets. And we calculate the local outlier factor to select the outliers as the hardware Trojans.

The remainder of the paper is arranged as follows. In Section II, the related works are briefly introduced. Section III illustrates the necessary preliminaries and some basic definitions. We give the design goals and problem models of this paper in Section IV. Section V displays the performance analysis and experiment results for the anomaly detection method and some comparisons with other methods. Finally, the whole work is concluded and the future work is shown in Section VI.

## II. RELATED WORK

This section will discuss the research status of related methods at home and abroad, including the research status of hardware Trojan detection methods and that of gate-level netlists detection methods based on machine learning.

### A. HARDWARE TROJANS DETECTION

Up to now, more and more researchers have made several researches on the detection methods of HTs in ICs. At present, hardware Trojan detection methods can be divided into two categories: pre-silicon detection and post-silicon detection. The pre-silicon can be divided into dynamic detection and static detection. The post-silicon includes destructive detection, logic testing, and side-channel analysis.

#### 1) THE PRE-SILICON DETECTION

- **a: DYNAMIC DETECTION**

Reference [16] is to mark suspicious circuits inserted into the chip netlists through a trust verification or scoring mechanism, and then use formal verification and functional simulation to determine whether there is a hardware Trojan in the suspicious circuit. Chen and Liu [17] proposed a single-trigger hardware Trojans identification method based on the characteristics of the gate-level circuit structure. This method had the advantages of low false positive rate and low overhead, but it couldn’t identify hardware Trojans with a special structure.

- **b: STATIC DETECTION**

Reference [18] is a detection method to identify the existence of HTs in the circuit description file through a machine learning algorithm. Salmani [19] proposed a hardware Trojan detection technique based on fast scanning electron microscope(SEM) imaging. However, this method came at the cost of destroying the circuit, and could only detect the circuit in a specific area. Furthermore, it could not guarantee the safety of other circuits.

#### 2) THE POST-SILICON DETECTION

- **a: DESTRUCTIVE DETECTION**

is also called reverse analysis [20]. Courbon et al. [21] proposed a high-efficiency hardware Trojan detection technique based on fast scanning electron microscope(SEM) imaging. However, this method came at the cost of destroying the circuit, and could only detect the circuit in a specific area. Furthermore, it could not guarantee the safety of other circuits.

- **b: LOGIC TESTING**

is also called function testing [22]. Xue et al. [23] divided the circuit into regions by using a heuristic partitioning technique based on scanning unit distribution and detected the hardware Trojans by analyzing local instantaneous current.

- **c: SIDE-CHANNEL ANALYSIS**

is performed by checking the abnormal behavior of the circuit parameters to be tested [24]. Chi et al. [25] used the technology of combining circuit design and detection to output test vectors of different circuits. This method could detect the chip with 0.3% HTs, but it was not suitable for the circuit with low proportion of HTs. Since the post-silicon detection is susceptible to the influence of process noise, and the detection cost is high. In practice, the post-silicon detection is limited. The pre-silicon detection has become the mainstream trend of HTs detection technology because it operates flexibly and tests easily.

### B. GATE-LEVEL NETLISTS DETECTION BASED ON MACHINE LEARNING

Nowadays, machine learning is widely used in various research and application fields due to its excellent predictive performance. However, few researchers have applied
machine learning to the detection of HTs in gate-level netlists at present.

Hasegawa et al. [26] extracted 5 Trojan netlist features through the preliminary analysis of normal circuit and Trojan circuit and applied support vector machine (SVM) to the detection of HTs in gate-level netlists for the first time. The experimental results showed that the trained SVM classifier can achieve 100% TPR in some cases. In order to verify the effectiveness of detecting HTs method based on SVM against the unknown netlists, three HTs were designed and implanted into the normal netlists [27].

Hasegawa et al. [28] used the Neural Network (NN) to detect the HTs on the basis of five hardware Trojan features. This method could obtain 81% of the average TPR and 69% of the TNR. And Hasegawa et al. [30] proposed 51 Trojan netlists features and used the random forest (RF) classifier to extract 11 features that could effectively detect and maximize the $F$-measure. Based on the 11 Trojan netlists features, the multilayer neural network (MNN) algorithm was used to train the hardware Trojan classifier in gate-level netlists by Hasegawa et al. [29].

Dong et al. [10] came up with a framework called RG-Secure. They used the lightGBM algorithm to detect gate-level netlists from Trust-HUB [31]. However, this method discarded some circuits on the Trust-HUB, and its high detection effect could only be reflected in these circuits. Dong et al. [11] proposed a locating method for multi-purposes HTs based on the boundary network to achieve the HTs’ precise locations. Dong et al. [12] proposed an XGBoost-based HTs detection method at the gate-level which obtained the best detection results among the existing supervised machine-learning-based HTs detection methods.

The above supervised methods were based on data sets with a small number of samples, so their effect in chip detection is still questionable. In addition, the training process for supervised learning methods tends to be time-consuming and generally requires huge amount of balanced training data from non-anomaly and anomaly classes.

Netlist I: Schematic diagram of gate-level netlist.

| Line | Description |
|------|-------------|
| 1    | module sample (a, b, c, d, e, f, cout) |
| 2    | input a, b, c, d, e, f; |
| 3    | output cout; |
| 4    | wire x, y, z, k, w; |
| 5    | XOR g1(x, a, b); |
| 6    | NAND g2(y, c, d); |
| 7    | OR g3(z, e, f); |
| 8    | AND g4(k, x, y); |
| 9    | NOR g5(w, y, z); |
| 10   | NXOR g6(cout, k, w); |
| 11   | endmodule |

III. PRELIMINARIES AND DEFINITIONS

A. GATE-LEVEL NETLISTS

The gate-level netlist is an intermediate state file in the design of an integrated circuit. It is a file that uses Verilog-HDL hardware description language to express the relationship between various circuit components, as shown in Netlist I. The gate-level netlists start with module and end with endmodule. The variable sample in Line 1 represents the name of the module. All relevant input $a, b, c, d, e, f$ and output signals cout of the module are listed in brackets after the sample. In Line 2, the input is used to mark the input signal of the module, while the output of the circuit module is defined as output in Line 3. In Line 4, the wire defines the names of the wire nets connecting the logic gates in the module. Then, in the following Line 5-10, the relationship between logic gates in the circuit is described, including the specific type of logic gate used, the name of the logic gate, and the relevant input and output signals of the corresponding logic gate. For instance, Line 9 indicates that the logic-gate is a NOR gate. The input of the NOR gate is signal $y$ and signal $z$ and its output is signal $w$. The corresponding physical circuit structure of this table is shown in Fig. 2.

B. HARDWARE TROJAN

Hardware Trojan [32] refers to the special circuit module maliciously implanted by the attacker or the circuit defect left by the designer by accident during the design and production of the integrated circuit. Under special circumstances, attackers can use hardware Trojan to realize information leakage, device function tampering, data loss, denial of service, and system paralysis attacks.

Generally, hardware Trojan consists of two parts: the trigger part and the payload part as is shown in Fig. 3. The trigger part of hardware Trojan is mainly used to monitor some rare circuit signals in the normal circuit. After meeting the trigger conditions of hardware Trojan, the circuit of the trigger part converts the rare circuit signals into effective signals to activate the payload circuit. And the payload part is the actual working circuit to realize the malicious attack. The normal circuits are also operated by triggers and payloads, so the Trojans cannot be detected by these modules. It is such a simple hardware Trojan that has strong concealment. It usually doesn’t work and doesn’t affect the normal circuit unless it is triggered under special conditions. By ingeniously designing, the rare signals or events are unlikely to arise...
during design simulation or testing but can probably occur during a long period of field operation [4]. Therefore, how to detect the hardware Trojans is a great challenge.

C. UNSUPERVISED MACHINE LEARNING: LOCAL OUTLIER FACTOR ALGORITHM

Unsupervised learning attempts to find structure in data automatically by extracting useful features without clear and specific instruction to desired outcomes [33]. The anomaly score of each sample is called Local Outlier Factor, which was first proposed by Breunig et al. [34]. It measures the local deviation of density of a given sample with respect to its neighbors. It is local in that the anomaly score depends on how isolated the object is with respect to the surrounding neighborhood. More precisely, the locality is given by the object [36]. As is seen in Fig. 4, the local outlier factor of an object \( o \) is defined as

\[
O(p) = \frac{1}{k} \sum_{o' \in N_o(p)} \text{dis}(d(p,o'))
\]

where \( N_o(p) \) denotes that these objects \( o \) are called the \( k \)-nearest neighbors of \( p \).

Definition 2 \( k \)-Distance Neighborhood of an Object \( p \): Given the \( k \)-distance of \( p \), the \( k \)-distance neighborhood of \( p \) contains every object whose distance from \( p \) is not greater than the \( k \)-distance. As shown in formula (1):

\[
N_k(p) = \{ o \in D \mid d(p,o) \leq k \_\text{dis}(p) \}
\]

where \( N_k(p) \) denotes that these objects \( o \) are called the \( k \)-nearest neighbors of \( p \).

Definition 3 Reachability Distance of an Object \( p \) With Respect to Object \( o \):

The reachability distance of object \( p \) with respect to object \( o \) is defined as:

\[
\text{reach}_\_\text{dis}(p, o) = \max \{ \text{dis}(d(p,o)), d(p, o) \}
\]

where \( k \) is a natural number. The strength of this smoothing effect can be controlled by the parameter \( k \). The higher the value of \( k \), the more similar the reachability distances for objects within the same neighborhood.

Definition 4 Local Reachability Density of an Object \( p \):

The local reachability density of object \( p \) in data set \( D \) is defined as:

\[
lrd_{\text{MinPts}}(p) = \frac{1}{|N_{\text{MinPts}}(p)|} \sum_{o \in N_{\text{MinPts}}(p)} \frac{\text{reach}_\_\text{dis}_{\text{MinPts}}(p, o)}{|N_{\text{MinPts}}(p)|}
\]

where the parameter \( \text{MinPts} \) specifies a minimum number of objects. \( \text{MinPts} \) determines a density threshold for the clustering algorithms to operate. That is, objects or regions are connected if their neighborhood densities exceed the given density threshold. Intuitively, the local reachability density of an object \( p \) is the inverse of the average reachability distance based on the \( \text{MinPts} \)-nearest neighbors of \( p \).

If object \( p \) has a high level of outlier, that is, there are few neighboring objects in the surrounding area, so its \( k \)-distance neighborhood covers a wide range, such as object \( O \) in Fig. 4. For object \( o \) in the \( k \)-distance neighborhood of an object \( p \), if the probability of object \( p \) in the \( k \)-distance neighborhood of object \( o \) is very small, then \( \text{reach}_\_\text{dis}(p, o) \) is more likely to be \( d(p,o) \). On the contrary, if the object \( p \) is located in the class

FIGURE 3. The architecture of Hardware Trojan.

FIGURE 4. Schematic diagram for local outlier factor.
cluster, for the object \( o \) in the \( k \)-distance neighborhood of \( p \), the probability of the object \( p \) in the \( k \)-distance neighborhood of object \( o \) is relatively large, then reach\_dis\( (p, o) \) is more likely to be \( k\_dis(o) \).

**Definition 5 (Local) Outlier Factor of an Object \( p \):**
The (local) outlier factor of \( p \) is defined as
\[
\text{LOF}_{\text{MinPts}}(p) = \frac{\sum_{o \in \text{NearestMinPts}(p)} \text{lof}_{\text{MinPts}}(o)}{|\text{NearestMinPts}(p)|}
\]

(4)

The outlier factor of object \( p \) captures the degree to which we call \( p \) an outlier. It is easy to see that the lower \( p \)’s local reachability densities are, and the higher the local reachability densities of \( p \)’s MinPts-nearest neighbors are, the higher the \( \text{LOF} \) value of \( p \) is. The closer the local outlier factor is to 1, it indicates that the density of the neighborhood objects of \( p \) is similar, and \( p \) may belong to the same cluster as the neighborhood. Objects with high \( \text{LOF} \) values are considered outliers.

**D. FEATURE DIMENSIONALITY REDUCTION METHOD**

In the application of machine learning methods, it is often necessary to extract appropriate features to describe a practical problem. Due to addressing high-dimensional data requiring more computational resources, it is much difficult or even intractable to efficiently deal with these practical problems within acceptable computing overhead. Excessive features may increase the risk of overfitting and decrease the efficiency of the learning algorithms. In other words, discarding those unimportant and redundant features may improve the learning efficiency as well as increase the clustering accuracy [13].

Feature dimensionality reduction is to transform features from high-dimensional into low-dimensional by means of mapping on the basis of retaining as much information as possible from the original set, and create a new feature subset based on the original set. As is shown in formula (5).

\[
Y = \begin{pmatrix}
  y_{11} & \cdots & y_{1q} \\
  \vdots & \ddots & \vdots \\
  y_{p1} & \cdots & y_{pq}
\end{pmatrix}
\]

\[
X = \begin{pmatrix}
  x_{11} & x_{12} & \cdots & x_{1m} \\
  x_{21} & x_{22} & \cdots & x_{2m} \\
  \vdots & \vdots & \ddots & \vdots \\
  x_{n1} & x_{n2} & \cdots & x_{nm}
\end{pmatrix}
\]

(5)

where the left side is the original eigenmatrix \( X \), the right is the new eigenmatrix \( Y \) after dimension reduction, \( m, n, p \) and \( q \) are positive integers, and \( q \leq m, p \leq n \).

1) PRINCIPAL COMPONENT ANALYSIS (PCA)

The basic idea of principal component analysis [38] is to maximize the divergence of new features obtained after dimensionality reduction of original features. But information represented by the raw features data should be remained as much as possible. In other words, it makes the data easier to distinguish by projecting the feature data in the maximum direction.

For the sample points in the orthogonal attribute space, if all samples are properly expressed by a hyperplane, then the hyperplane should have the following properties:

- **Nearest refactoring:** the distance from the sample point to the hyperplane is close enough.
- **Maximal separability:** the projection of sample points onto the hyperplane can be separated as far as possible.

**Algorithm 1 Algorithm for Principal Component Analysis**

**Input:** The sample data set \( D = \{x_1, x_2, \ldots, x_m\} \).

**Low-dimensional space dimension \( d' \).**

**Output:** Projection matrix \( W = (w_1, w_2, \ldots, w_d') \);

1. Centralize all samples: \( x_i \leftarrow x_i - \frac{1}{m} \sum_{j=1}^{m} x_j \);
2. Calculate the covariance matrix of the samples \( XX^T \);
3. Eigen decomposition of covariance matrix \( XX^T \);
4. Take the eigenvectors \( (w_1, w_2, \ldots, w_d') \) corresponding to the largest \( d' \) eigenvalues as \( W \);
5. **return** \( W \);

Based on the nearest refactoring and the maximal separability, two equivalent derivations of principal component analysis can be obtained respectively. Let’s start with the nearest refactoring.

Given the sample data set \( D = \{x_1, x_2, \ldots, x_m\} \) with \( m \) feature dimensions. Assume that the data sample is centralized, that is, \( \sum x_i = 0 \). Suppose the new coordinate system obtained after the projection transformation is \( \{w_1, w_2, \ldots, w_d'\} \), where \( w_i \) is the orthonormal basis vector and \( \|w_i\|_2 = 1 \), \( w_i^T w_j = 0 (i \neq j) \). If some of the coordinates in the new coordinate system are discarded, and the dimensions are reduced to \( d' < d \), then the projection of \( x_i \) onto a lower-dimensional coordinate system is \( z_i = \{z_{i1}; z_{i2}; \ldots; z_{id'}\} \), where \( z_{ij} = w_i^T x_i \) means the \( j \)-th coordinate of \( x_i \) in low-dimensional coordinate system. If \( x_i \) is refactored based on \( z_i \), then \( \hat{x}_i = \sum_{j=1}^{d'} z_{ij} w_j \) will be obtained.

Considering the whole training set, the distance between the original sample point \( x_i \) and the sample point output \( \hat{x}_i \) based on projection refactoring is calculated as follows:

\[
\sum_{i=1}^{m} \left\| \sum_{j=1}^{d'} z_{ij} w_j - x_i \right\|^2 = \sum_{i=1}^{m} \sum_{i=1}^{d'} (z_{ij})^2 - 2 \sum_{i=1}^{m} z_{ij}^T W^T x_i + c \]

\[
\propto -tr \left( W^T \left( \sum_{i=1}^{m} x_i x_i^T \right) W \right)
\]

(6)

where \( \sum_i x_i x_i^T \) is a covariance matrix, \( c \) is a constant. Formula (6) should be minimized according to the nearest refactoring. And the optimization goal of PCA is shown as the following:

\[
\min_W tr \left( W^T XX^T W \right) \quad s.t. \quad W^T W = I.
\]

Another interpretation of principal component analysis can be obtained from maximal separability. We know that the projection of sample point \( x_i \) on the hyperplane in the new space
is $W^T x_i$. If the projection of all samples can be separated as far as possible, the variance of the samples after the projection should be maximized. The variance of the sample points after the projection is $\sum_i W^T x_i x_i^T W$, then the optimization objective can be written as:

$$\max_W \text{tr} \left( W^T XX^T W \right)$$

s.t. $W^T W = I$. \hspace{1cm} (8)

Obviously, formula (7) is equivalent to formula (8) which can be obtained by using the Lagrange multiplier method:

$$XX^T W = \lambda W$$ \hspace{1cm} (9)

As a result, it is only necessary to decompose the feature values of the covariance matrix $XX^T$ and sort the obtained values: $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_d$, then take the eigenvectors corresponding to the first $d'$ feature values to form $W = \{w_1, w_2, \ldots, w_{d'}\}$ which is the solution to PCA. The dimension $d'$ of a reduced dimensional space is usually specified by the user in advance.

Undoubtedly, lower-dimensional space must be different from the original higher-dimensional space. Because the eigenvectors that correspond to first $d - d'$ minimum feature values are discarded, that is the consequence of dimensionality reduction.

With the aforementioned analyses, the procedure for principal component analysis is summarized in Algorithm 1.

**IV. HARDWARE TROJAN DETECTING BASED ON UNSUPERVISED MACHINE LEARNING**

**A. THE OVERALL ARCHITECTURE OF PL-HTD**

In this paper, PCA is used to extract the principle components from hardware Trojan features to reduce the computation load of the LOF, while ensuring that the least information is lost, and providing a better centroid point for clustering. LOF is used to calculate the outlier factor of each net in the gate-level netlists to find the suspicious abnormal nets that are highly dissimilar from the others. Prior information (e.g., the names of Trojans) for labeling training data is unavailable in this study. Fig. 5 is the block diagram of PL-HTD, which shows the entire content of our theoretical model. We’ve roughly divided it into three parts: **input**, **system** and **output**. And the process of our PL-HTD is summarized as follows:

- **Part I: Input**
  - **STEP 1**: That’s the starting of the whole process. The gate-level netlist files to be detected with unknown security are used as input by our unsupervised-learning-based method.

- **Part II: System**
  - **STEP 2**: **Extract the feature values.** According to relevant circuit characteristics, we extract the feature data from the netlist and then label them to create the validation set. In this step, the gate-level netlist file is transformed into the data required during the execution of the unsupervised learning method. Summarize the corresponding net features as feature 1, feature 2, . . . , feature $n$.

  - **STEP 3**: **Normalization of feature data.** The range of values for different features is different. However, the large difference between the value range of features will have a certain impact on the establishment of the model. Consequently, the feature data set is standardized so that the corresponding data set is transformed into a standard normal distribution.

- **STEP 4**: **Dimension reduction of feature data set by PCA**
### TABLE 1. The extracted features from a netlist.

| Label | Trojan net feature candidate | Description ($x \in \{1, 5\}$) |
|-------|------------------------------|---------------------------------|
| $f_0$–$f_4$ | fan_in_x | The number of logic-gate fan-ins $x$-level away from the net $n$. |
| $f_5$–$f_9$ | in_fliplp_x | The number of flip-flops up to $x$-level away from the input side of the net $n$. |
| $f_{10}$–$f_{14}$ | out_fliplp_x | The number of flip-flops up to $x$-level away from the output side of the net $n$. |
| $f_{15}$–$f_{19}$ | in_multiplexer_x | The number of multiplexer up to $x$-level away from the input side of the net $n$. |
| $f_{20}$–$f_{24}$ | out_multiplexer_x | The number of multiplexer up to $x$-level away from the output side of the net $n$. |
| $f_{25}$–$f_{29}$ | in_loop_x | The number of up to $x$-level loops from the input side of the net $n$. |
| $f_{30}$–$f_{34}$ | out_loop_x | The number of up to $x$-level loops from the output side of the net $n$. |
| $f_{35}$–$f_{39}$ | in_const_x | The number of constants up to $x$-level away from the input side of the net $n$. |
| $f_{40}$–$f_{44}$ | out_const_x | The number of constants up to $x$-level away from the output side of the net $n$. |
| $f_{45}$ | in_nearest_pin | The minimum level to the primary input from the net $n$. |
| $f_{46}$ | out_nearest_pout | The minimum level to the primary output from the net $n$. |
| $f_{47}$–$f_{48}$ | (in, out).nearest_fliplp | The minimum level to any flip-flop from the input or output side of the net $n$. |
| $f_{49}$–$f_{50}$ | (in, out).nearest_multiplexer | The minimum level to any multiplexer from the input or output side of the net $n$. |
| $f_{51}$–$f_{55}$ | in_gate_x | The number of logic-gate $x$-level away from the net $n$. |

- **STEP 4**: Dimension reduction of feature set by PCA. In order to reduce the computational cost of the model, and to be able to visually display the distribution characteristics of the data of the gate-level netlist in a low-dimensional plane, the feature data set of the gate-level netlist can be operated with the corresponding dimensionality reduction. The dimension of the feature is reduced from $n$ to $m$.

- **STEP 5**: Calculate the local outlier factor using LOF. We use the LOF algorithm to calculate the corresponding outlier for each data sample object in the data set of the gate-level netlists, and then sort the outliers according to the degree of anomaly.

- **Part III: Output**

- **STEP 6**: Detect the hardware Trojans in gate-level netlists. By specifying relevant parameters in the local outlier algorithm, the outputs of the specified number of suspicious objects are sorted according to the outliers from large to small, and later the professional staff will make further analysis and detection.

### B. HARDWARE TROJAN FEATURE

1) **EXISTING HARDWARE TROJAN FEATURES**

Fifty-one features proposed by Hasegawa et al. [30] are feature candidates, which are shown in Table 1. Here we make a brief introduction to the first feature “fan_in_x”. It indicates that the number of logic-gate fan-ins $x$-level away from the arbitrary net $n$ from input to output. The other features are similar. By analyzing the fan_in_x feature among the existing features, it can be found that the malicious attacker can easily alter the structure of the Trojan circuit to an equivalent one, thus changing the value of the fan_in_x feature. So the number of logic gates far away from $x$-level target net $n$ is defined as in_gate_x except the 51 features [12].

The data provided by the Trust-HUB is the source file of the gate-level netlists, which cannot be directly used for the hardware Trojans classifier. Instead, the feature extraction of the gate-level netlist file is required to convert the text file into the data set required for our machine learning classifier. In this paper, feature extraction is carried out within the range of level 5 of the input direction and output direction of the target networks.

2) **SELECTING HARDWARE TROJAN FEATURES USING XGBoost AND ANALYZING THE CORRELOGRAM OF SELECTED FEATURES**

Sixteen features greater than 10000 points obtained by XGBoost were used as the original feature data set of this experiment to extract the corresponding feature data. These features are respectively: fan_in_1(T1), fan_in_2(T2), fan_in_3(T3), fan_in_4(T4), fan_in_5(T5), in_loop_4(T6), in_loop_5(T7), out_loop_5(T8), out_nearest_pout(T9), out_nearest_fliplp(T10), out_nearest_multiplexer(T11), in_gate_2(T12), in_gate_3(T13), in_gate_4(T14), in_gate_5(T15), where $T_n$ in parentheses is the alias of the corresponding feature. From the researchers’ point of view, they are significant features which can improve the discriminative capability of PL-HTD.

The correlogram [37] can visually indicate the correlation between the features, and the correlation strength is represented by the color and the corresponding value. The stronger the correlation, the darker the color, and the larger the corresponding value is. From the information redundancy which is shown in Fig. 6, it can be known that some features are highly correlated, that is, these features contain much overlapping information, and not all data are distributed independently of each other. Therefore, there are some offsets. For supervised learning, large offsets are likely to bring great noise to the classifier, which weakens the usability, reliability, and interpretability of classification results. While for unsupervised, the situation is much better. Hence, non-independent data is more suitable for unsupervised learning.

### C. PROBLEM FORMULATION

1) **DETECTION MODEL**

To efficiently detect the hardware Trojans in gate-level netlists, the overall architecture of PL-HTD is shown in Fig. 5 and the detection model can be defined as:
Given: (i) Gate-level netlists with unknown security status modelled as a set $G_n = \{G_f', G_{ht}^m\}$, where $n$ is the number of nets in the netlists, $m$ represents the dimension of the netlist feature. $G_f'$ and $G_{ht}^m$ respectively indicate the HT-free netlists and HT-inseted netlists. (ii) The computing system of eigenvectors after dimensionality reduction can be expressed as $\sum_{i=1}^{m} x_i^T \lambda_i = \sum_{j=1}^{d} \lambda_j w_j^T w_j^T$, $d \leq m$, where $x_i$ is the eigenvector of $G_{ht}^m$, $w_j$ is the orthonormal basis vector for the projection, $d$ is the feature dimension after projection transformation and $\lambda_j$ is the eigenvalue. Sort the obtained eigenvalues and define eigenvectors that correspond to the first $d'$ eigenvalues as $W = \{w_1, w_2, \ldots, w_{d'}\}$, where $d' < d$. (iii) The suspicious netlists are those which satisfy this expression: $G_{ht}^m \in \{p|p = \text{LOF}_k(p) \geq 1\}$. Here, $\text{LOF}_k(p)$ is from formula (4).

Find: A proper $d'$, the neighbors $k$ and contamination of nets make detected hardware Trojans as numerous and accurate as possible.

Constrains: $d' \in \{d|0.6 \leq \sum_{i=1}^{D} \lambda_i / \sum_{i=1}^{d} \lambda_i \leq 1\}$

Objective: Minimize the false positive rate of normal netlists and maximize hardware Trojans detection precision.

2) DESIGN GOAL
Our goal is to design a hardware Trojans detection model based on unsupervised machine learning with high detection accuracy. Specifically, we have the following goals:

- **Applicability.** We hope that this unique idea of our hardware Trojans detecting method in gate-level netlists can be applicable in a broad area.
- **Accuracy.** The Trojan nets should be detected as many as possible using our PL-HTD. Also, it can realize the correct judgment of normal netlists.

## D. EVALUATION METRIC
The detection of hardware Trojans in gate-level netlists belongs to the binary classification problem. There are usually four indexes to evaluate the classification results, as is shown in Table 2. In our experiment, $TP$ indicates the number of Trojan nets identified to be Trojan nets; $FN$ is the number of normal nets identified to be Trojan nets mistakenly; $TN$ is the number of normal nets identified to be normal nets; $FP$ is the number of Trojan nets identified to be normal nets mistakenly. Based on these indexes, several new indexes can be calculated as:

- \[
\text{Precision}(P) = \frac{TP}{TP + FP} \tag{10}
\]
- \[
\text{TNR} = \frac{TN}{TN + FN} \tag{11}
\]
- \[
\text{TPR} = \frac{TP}{TP + FN} \tag{12}
\]
- \[
\text{Recall} = \frac{TP}{TP + FN} \tag{13}
\]
- \[
\text{Accuracy} = \frac{TP + TN}{TP + TN + FN + FP} \tag{14}
\]

Without the guidance of supervised information, it is tough to search a correspondence between the predictive clustering labels and the real ground truths [40]. Therefore, the quality of clustering performance is assessed by five popular evaluation metrics, including the true positive rate (TPR), the true negative rate (TNR), the recall (R), the precision (P), and the accuracy (ACC). However, they are not satisfactory measures of assessment for unbalanced data sets due to the dominant effect of the majority class; therefore, the Receiver Operating Characteristic Curve (ROC) [41] is also used to select an appropriate algorithm for dimensionality reduction of feature set to optimize the detection effect of hardware Trojans. The area under the curve is known as $AUC$ [42]. In this study, we focus on TNR, Recall and Precision to evaluate the detection results.

## V. EXPERIMENTAL RESULTS AND ANALYSIS
### A. SET UP OF EXPERIMENTS
Our experiments detect Trojans in Trust-HUB [31] benchmark netlists. Figure 7 shows the statistics of the normal nets and Trojan nets for some netlists, which are used in our experiment. Though the same benchmark is used, the number of Trojan nets and normal nets may be different from other papers. The number of nets in the extracted netlist cannot be.

## TABLE 2. Confusion matrix representation.

| Truth         | Prediction       |
|---------------|------------------|
| Trojan net    | True Positive(TP) | False Negative(FN) |
| Normal net    | False Positive(FP) | True Negative(TN)  |

*Effectiveness.* Compared with other hardware Trojan detection methods, the time cost of our algorithm cannot be too high. And it’s essential to detect as many hardware Trojans as time permits.
guaranteed to be the same as the real circuits, but it is the nets extracted by our program.

The framework is built by the Python language. The feature extraction experiments are carried out on Windows 7 server with an Intel central processing unit (CPU) running at 3.20GHz and 4GB memory. The classification experiments are carried out on a Windows 10 server with an Intel i5-7200U CPU running at 2.71GHz and 4GB memory. The details are shown in Table 3.

### B. DATA PRE-PROCESSING: DATA LABELING

Because this research focuses on unsupervised learning, we need to cope with unlabeled data. That means that the data inside the dataset miss labels. Without the guidance of supervised information, it is tough to search a correspondence between the predictive clustering labels and the real ground truths. However, in order to validate the effectiveness and insurance that our clustering algorithms are accurately classifying most of the nets in the data, we first want to work with labeled data. This data labeling must be done manually by us. The labels are used to tune the unsupervised model, that is, they act on the validation set. In the process of marking Trojan nets, this paper only considers the internal nets of the Trojan circuit as Trojan nets. Because the boundary network and the normal one are connected, two types of circuits are respectively on both sides of this kind of network, so boundary network is not considered temporarily. The networks marked as the red dotted lines are considered as Trojan nets in this paper, as shown in Fig. 8 which represents the Trojans implanted in the RS232-T1000 gate-level netlist.

### C. MODEL SELECTION AND PARAMETER SETTING

In order to analyze the data distribution more intuitively, PCA and factor analysis (FA) [39] were used to reduce the dimension of 16 features which are based on s35932 benchmark, and the most suitable one was selected by comparison. At the same time, the first principal component and the second principal component obtained by PCA construction are respectively x-axis and y-axis, so as to map the data set to a two-dimensional plane, as shown in Fig. 9(a). Similarly, the first factor and the second factor constructed by FA are taken as the x-axis and y-axis respectively, as shown in Fig. 9(b). In Fig. 9, each point represents a net, the purple is the Trojan net, and the yellow is the normal net. According to Fig. 9, it can be seen that the data obtained by PCA have more characteristics of heap distribution. Moreover, the corresponding points of Trojan netlists are mostly distributed in the sparse area.

In the case of default parameters of LOF, the 2-d data obtained by PCA and those obtained by FA are used to draw the corresponding ROC curve and calculate the corresponding AUC. In Fig. 9(c), the AUC obtained based on PCA is 0.88 while the AUC obtained based on FA is 0.85 shown in Fig. 9(d). From what has been discussed above, PCA is better than FA not only in data density distribution, but also in the output effect of the preliminary model. So PCA is used for modeling in this paper.

As to parameter settings, the model parameters of our method are represented in Table 4. In order to get the best performance, the parameters $k$, $d'$ and contamination are adjusted during the experiments. We set $k$, $d'$ and contamination to 250, 2, 0.05 in RS232 series and s15850-T100 benchmark respectively, while setting 200, 10, 0.002 in s35932 series and s38417-T300. An appropriate $k$ is important for the proposed method to exactly detect the abnormal segment. It is not trivial to determine the right value of $k$ as lower value indicates more focus on local neighbours whereas larger $k$ may miss local outliers. Moreover, there is usually no information about hardware Trojans can be used to guide the selection of $k$ in detecting the gate-level netlists. The parameter contamination represents the proportion of outliers set during model training. It is regarded that the higher the contamination value is, the more normal nets will be.
TABLE 4. The classification results.

| Test data   | contamination | k  | $d'$ | TN   | FP   | FN   | TP   | TPR(%) | TNR(%) | Precision(%) | F-measure(%) | Accuracy(%) |
|-------------|---------------|----|------|------|------|------|------|--------|--------|--------------|--------------|-------------|
| R232-T1000  | 0.05          | 250| 2    | 270  | 10   | 5    | 5    | 50.00  | 96.43  | 33.33        | 40.00        | 94.83       |
| R232-T1100  | 0.05          | 250| 2    | 270  | 10   | 6    | 5    | 45.45  | 96.43  | 33.33        | 40.00        | 94.83       |
| R232-T1200  | 0.05          | 250| 2    | 272  | 9    | 7    | 6    | 46.15  | 97.14  | 40.00        | 42.86        | 94.56       |
| R232-T1300  | 0.05          | 250| 2    | 267  | 11   | 3    | 4    | 57.14  | 96.04  | 26.67        | 35.94        | 95.09       |
| R232-T1400  | 0.05          | 250| 2    | 268  | 10   | 7    | 5    | 41.67  | 96.40  | 33.33        | 37.04        | 94.14       |
| R232-T1500  | 0.05          | 250| 2    | 272  | 10   | 6    | 5    | 45.45  | 96.45  | 33.33        | 38.46        | 94.54       |
| R232-T1600  | 0.05          | 250| 2    | 272  | 11   | 5    | 4    | 44.44  | 96.11  | 26.67        | 33.33        | 94.52       |
| s15850-T100 | 0.05          | 250| 2    | 2079 | 91   | 7    | 19   | 73.08  | 95.81  | 17.28        | 27.95        | 95.54       |
| s35932-T100 | 0.002         | 200| 10   | 5799 | 52   | 13   | 1    | 7.14   | 99.11  | 1.89         | 3.00         | 99.13       |
| s35932-T200 | 0.002         | 200| 10   | 5842 | 6    | 8    | 4    | 33.33  | 99.90  | 40.00        | 36.36        | 99.76       |
| s38417-T300 | 0.002         | 200| 10   | 5463 | 2    | 34   | 10   | 22.73  | 99.96  | 83.33        | 35.72        | 99.37       |
| Average     | -             | -  | -    | -    | -    | -    | -    | -      | -      | -             | -             | -           |

FIGURE 9. Two-dimensional plane scatter diagram and ROC curve of model output.

- (a) The scatter diagram of PCA
- (b) The scatter diagram of FA
- (c) ROC of PCA
- (d) ROC of FA

Considered as abnormal nets by the model. Generally, a low value of contamination may extract very general patterns while a high value may be severely affected by noise. Thus, the adaptive selection of $k$, $d'$ and contamination will be considered in our future work to detect the abnormal nets effectively.

D. RESULTS OF THE TROJAN DETECTION

In this section, we analyze the performance of our proposed unsupervised learning detection method. The results of hardware Trojan detection by our PL-HTD are shown in Table 4. As shown in Table 4, all TNR of the benchmarks we test are above 95%, which means almost all the normal nets can be correctly judged. The average TPR reaches 42.42%. Certainly, PL-HTD may fail to improve the classifying performance in some specific situations. For instance, it performs undesirable in the netlist s35932-T100, which has a TPR below 10%. Although the detection result of s35932-T100 is relatively poor compared to other netlists, this result also proves that our proposed method is effective for the detection of hardware Trojans because it shows that at least one true Trojan netlist can be detected even if the number of model output Trojan netlists is small. At the same time, our detection algorithm can guarantee a low misjudgment rate. The average F-measure is 33.56% and the best Precision obtained by our method is 83.33%, to some extent, the workload of subsequent manual detection can be reduced, which is quite a good result. In terms of Accuracy, almost all netlists obtain more than 94.00% or more, especially for s35932-T100, s35932-T200 and s38417-T300, the average of accuracy can reach above 99%. It yields good results in detecting Trojans in unlabeled gate-level netlists that indicates the ability of PL-HTD to analyse latent circuit structure.

Generally speaking, the TNR is satisfactory, but the TPR and F-measure still need further improvement. A fraction of netlists remains misclassified by our unsupervised learning classifiers. On one hand, it is an unavoidable problem due to the design of the algorithm itself. While eliminating redundant information between the original features, some useful information about the original data will inevitably be lost. The new features constructed may not all be independent of each other. On the other hand, the inability of the classifiers to detect these Trojan indicates that they are very advanced and succeed in pretending to function and perform similarly to the original host circuit. Without the guidance of supervised information, it is tough to search a correspondence between the predictive labels and the real ground truths.

E. COMPARISON WITH SUPERVISED LEARNING METHODS

In order to prove the effectiveness and accuracy of our detecting method, some comparisons are made in this section. Then, we will systematically analyse the strengths and weaknesses of our method.

In the experiment, the performances of the proposed method have been compared with the other existing methods,
TABLE 5. Comparison between [28], [29] and our proposed method.

| Test data          | [28]    | [29]    | Ours     | [28]    | [29]    | Ours     | [28]    | [29]    | Ours     | [28]    | [29]    | Ours     | [28]    | [29]    | Ours     |
|--------------------|---------|---------|----------|---------|---------|----------|---------|---------|----------|---------|---------|----------|---------|---------|----------|
| RS232-T1000        | 53.33   | 100.00  | 50.00    | 30.82   | 24.00   | 96.43    | 11.54   | 14.34   | 33.33    | 18.79   | 25.08   | 40.00    | 34.08   | 32.58   | 94.83    |
| RS232-T1100        | 58.33   | 78.00   | 45.45    | 27.00   | 25.00   | 96.43    | 3.10    | 11.6   | 33.33    | 5.88    | 20.27   | 38.46    | 28.21   | 30.96   | 94.50    |
| RS232-T1200        | 80.00   | 91.00   | 46.15    | 25.57   | 55.00   | 97.14    | 3.40    | 19.22   | 40.00    | 6.53    | 31.74   | 42.86    | 27.30   | 58.76   | 94.56    |
| RS232-T1300        | 88.89   | 86.00   | 57.14    | 25.84   | 65.00   | 96.04    | 3.49    | 19.89   | 26.67    | 6.72    | 32.31   | 35.94    | 27.69   | 66.93   | 95.09    |
| RS232-T1400        | 83.33   | 100.00  | 41.67    | 22.07   | 15.00   | 96.40    | 4.12    | 16.24   | 33.33    | 7.84    | 27.95   | 37.04    | 24.44   | 27.03   | 94.54    |
| RS232-T1500        | 83.33   | 82.00   | 45.45    | 23.51   | 47.00   | 96.45    | 4.15    | 17.57   | 33.33    | 7.91    | 28.95   | 38.46    | 25.80   | 51.24   | 94.54    |
| RS232-T1600        | 88.89   | 100.00  | 44.44    | 25.84   | 28.00   | 96.11    | 3.49    | 11.80   | 26.67    | 6.72    | 21.04   | 33.33    | 27.69   | 34.23   | 94.52    |
| s15850-T100        | 92.59   | 97.00   | 73.08    | 65.75   | 96.00   | 95.81    | 2.92    | 21.14   | 17.28    | 5.66    | 34.71   | 27.95    | 65.64   | 96.01   | 95.54    |
| s35932-T100        | 93.33   | 81.00   | 7.14     | 59.76   | 99.00   | 99.11    | 0.54    | 15.77   | 1.89     | 1.07    | 26.35   | 3.00     | 59.85   | 98.96   | 99.13    |
| s35932-T200        | 100.00  | 80.00   | 33.33    | 59.18   | 88.00   | 99.90    | 0.61    | 1.34    | 1.33     | 0.01    | 40.00   | 1.31     | 59.29   | 87.96   | 99.76    |
| s38417-T300        | 100.00  | 100.00  | 22.73    | 71.60   | 94.00   | 99.96    | 2.60    | 11.22   | 83.33    | 5.07    | 20.18   | 35.72    | 71.82   | 94.05   | 99.36    |
| Average            | 83.82   | 90.45   | 42.42    | 39.72   | 57.82   | 97.52    | 3.63    | 14.53   | 33.56    | 6.67    | 24.60   | 33.56    | 41.07   | 61.70   | 96.00    |

FIGURE 10. The line charts of Trojan Detection algorithms.

The detection results of HT-free netlists are shown in Fig. 11. We compare the results with that of supervised learning XGBoost algorithm [12]. When PL-HTD was used to detect the three netlists of free-s15850, free-s35932 and free-s38417, TNR reached more than 99.80%, which can correctly judge the normal network, almost no misjudgment of the normal nets as Trojan nets. Compared with XGBoost, our PL-HTD's average TNR increases by 1.03% and the average Accuracy(ACC) is only about 3.01% smaller. To sum up, PL-HTD is effective for the detection of normal gate-level netlists without Trojan. The comparison results reveal that unsupervised approaches are also able to produce good prediction performance without any labeled data.

F. TIME COMPLEXITY ANALYSIS

Table 6 compares the time complexity of some existing works of literature. Reference [28] uses support vector machine(SVM) with a time complexity between $O(N^3 + l(N^2 + N_S d_L l))$ to $O(dL l^2)$, where $N_S$ is the number of support vectors, $l$ is the number of training points, and $d_L$ is the dimension of the input data [43]. And [29] uses a multi-layer neural network with a time complexity of $O(N_{m+1} - 1 N_m * N_{m+1})$, where $N_m$ is the number of nodes in the $m$-th layer of the neural network.

The complexity of the LOF algorithm depends on the neighborhood query operation. The time complexity...
of each neighborhood query is $O(n^2)$, so that of the LOF algorithm is $O(n^2)$ due to all neighborhood queries [34]. PCA feature extraction is mainly time-consuming in the singular value decomposition of the covariance matrix [44]. Combining the two algorithms, so PL-HTD’s time complexity is $O(n^2 + d \times n^2 + 3 \times d^2 \times (d - 1) + 2 \times d' \times d \times n)$, where $d$ is the dimension of the data set, and $d'$ is the dimension after feature extraction.

Some regrets, our TPR and F-measure do not reach the standard in detecting Trojans. Owing to the dimensionality reduction, we hold the view that PCA is very sensitive to outliers, and each principal component is obtained by all linear combinations of features. Therefore, its learning subspace cannot well reflect the importance of each feature. PCA only uses the global reconstruction information of data but does not make full use of the correlation between samples, such as local information and group information, which are often very discriminative and can greatly enhance the classification effect. Also, it is an unavoidable problem that the time complexity is high because LOF usually incurs significant efforts in collecting the complete set of normal nets before detection starts. PL-HTD also suffers from certain drawbacks such as unsupervised techniques require large amount of data with good coverage in order to perform effectively. But on the whole, our TNR achieves a similar performance level as the fully supervised classification XGBoost [12]. Our outliers detection is useful in improving the circuit quality and the follow-up analysis accuracy. In a sense, there are always such problems in real life, such as lack of sufficient prior knowledge, so it is difficult to manually label category or it can be expensive (or even impossible) to determine labels for all data. Unsupervised learning is unbiased since there is no need to utilize experts or data analysts to categorize the samples, and it still can perform well even when no prior knowledge is available [45].

VI. CONCLUSION AND FUTURE WORK

The objective of this paper is to detect the abnormal nets in gate-level netlists. To the best of our knowledge, this is the first attempt to apply the unsupervised machine learning to hardware Trojans detection in gate-level netlists. To discard some less important features, this paper uses the scoring mechanism of the XGBoost algorithm to further extract the effective set of 16 Trojan-net features out of 56 features. Next, the dimension of the dataset is reduced by PCA, the redundancy between the data is removed, and the outliers are selected as the suspicious abnormal nets by LOF. And then it is further analyzed and confirmed by professionals.

Our method shows a great development prospect during the experiments. Future work will include theoretical approaches on how to improve the performance (i.e., Precision, Recall, and F-measure) of the proposed method which identifies suspicious netlists. We will investigate the adaptive selection of the parameters for the wider application of our proposed method. The authors will also further study the structure of gate-level Trojans to select a more appropriate unsupervised learning algorithm to achieve a better Trojan detection result.

REFERENCES

[1] W. Guo, W. Zhu, Z. Yu, J. Wang, and B. Guo, “A survey of task allocation: Contrastive perspectives from wireless sensor networks and mobile crowdsensing,” IEEE Access, vol. 7, pp. 78406–78420, 2019.
[2] R. Khan, S. U. Khan, R. Zaheer, and S. Khan, “Future Internet: The Internet of Things architecture, possible applications and key challenges,” in Proc. 10th Int. Conf. Frontiers Inf. Technol., Dec. 2012, pp. 257–260.
[3] G. Sumathi, L. Srivani, D. T. Murthy, K. Madhusoodanan, and S. A. V. S. Murty, “A review on HT attacks in PLD and ASIC designs with potential defence solutions,” IETE Tech. Rev., vol. 35, no. 1, pp. 64–77, Jan. 2018.
[4] S. Dupuis, M.-L. Flottes, G. Di Natale, and B. Rouzeyre, “Protection against hardware trojans with logic testing: Proposed solutions and challenges ahead,” IEEE Des. Test. IEEE Des. Test. Comput., vol. 35, no. 2, pp. 73–90, Apr. 2018.
[5] G. Chen, W. Guo, and Y. Chen, “A PSO-based intelligent decision algorithm for VLSI floorplanning,” Soft Comput., vol. 14, no. 12, pp. 1329–1337, Oct. 2010.
C. Dong et al.: Unsupervised Detection Approach for HTs
JINGHUI CHEN received the B.E. degree in computer science from the College of Mathematics and Computer Science, Fujian Normal University, China, in 2017. He is currently pursuing the degree with the College of Mathematics and Computer Science, Fuzhou University. His main research interests include artificial intelligence and chip security.

XIMENG LIU (Member, IEEE) received the B.Sc. degree in electronic engineering and the Ph.D. degree in cryptography from Xidian University, Xi’an, China, in 2010 and 2015, respectively. He is currently a Full Professor with the College of Mathematics and Computer Science, Fuzhou University. He is also a Research Fellow with the School of Information System, Singapore Management University, Singapore. He has published more than 100 articles on the topics of cloud security and big data security, including articles in the IEEE TRANSACTIONS ON COMPUTERS, the IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, the IEEE TRANSACTIONS ON SERVICES COMPUTING, and the IEEE INTERNET OF THINGS JOURNAL. His research interests include cloud security, applied cryptography, and big data security. He is a member of ACM and CCF. He was awarded the Minjiang Scholars Distinguished Professor, the Qishan Scholars in Fuzhou University, and the ACM SIGSAC China Rising Star Award, in 2018. He has served on a program committee for several conferences, such as the 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, 2017 IEEE Global Communications Conference, and 2016 IEEE Global Communications Conference. He has served as the Lead Guest Editor for Wireless Communications and Mobile Computing.

WENZHONG GUO (Member, IEEE) received the B.S. and M.S. degrees in computer science and the Ph.D. degree in communication and information system from Fuzhou University, Fuzhou, China, in 2000, 2003, and 2010, respectively. He is currently a Full Professor with the College of Mathematics and Computer Science, Fuzhou University. He also leads the Network Computing and Intelligent Information Processing Laboratory, which is a key Laboratory of Fujian Province, China. His research interests include cloud computing, mobile computing, and evolutionary computation.

YUZHONG CHEN (Member, IEEE) received the B.S. and Ph.D. degrees in communication and information system from the University of Science and Technology of China, in 2000 and 2005, respectively. He is currently a Professor with the College of Mathematics and Computer Science, Fuzhou University. He is also the Vice Chief of the Fujian Provincial Key Laboratory of Network Computing and Intelligent Information Processing. His current research interests include computational intelligence, computer vision, and data mining.

* * *