Received November 11, 2018, accepted December 10, 2018, date of publication December 18, 2018,
date of current version January 11, 2019.

Digital Object Identifier 10.1109/ACCESS.2018.2887268

Defeating Untrustworthy Testing Parties: A Novel Hybrid Clustering Ensemble Based Golden Models-Free Hardware Trojan Detection Method

MINGFU XUE[1], **(Member, IEEE), RONGZHEN BIAN**[1], **WEIQIANG LIU**[2], **(Senior Member, IEEE), AND JIAN WANG**[1]
[1]College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China
[2]College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China

Corresponding author: Mingfu Xue (mingfu.xue@nuaa.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61602241, in part by the Natural Science Foundation of Jiangsu Province under Grant BK20150758, in part by the CCF-Venustech Hongyan Research Plan under Grant CCF-VenustechRP2016005, in part by the CCF-NSFocus Kunpeng Foundation under Grant CCF-NSFocus2017003, and in part by the Fundamental Research Funds for the Central Universities under Grant NS2016096.

ABSTRACT Due to the globalization of the design and fabrication process of integrated circuits (ICs), ICs are becoming vulnerable to hardware Trojans. Most of the existing hardware Trojan detection works assume that the testing stage is trustworthy. However, testing parties may collude with malicious attackers and modify the results of hardware Trojan detection. In this paper, two attack models for untrustworthy testing parties are formulated. We further propose an adversarial data generation method for untrustworthy testing parties to modify the collected test data. Then, we propose a novel hybrid clustering ensemble method to build a trusted hardware Trojan detection method (clustering ensemble-based hardware Trojan detection method) against untrustworthy testing parties. To alleviate the impact of process variations and noises on hardware Trojan detection in the actual measurement, the unsupervised correlation-based feature selection method is exploited to process the raw test data of ICs for feature selection. The proposed method can eliminate the need of the fabricated golden chips and the simulated golden models. It can also resist the malicious modifications on Trojan detection results introduced by untrustworthy testing parties. Besides, the following problems and questions are also theoretically analyzed and answered: 1) the number of necessary testing parties; 2) the time overhead and the computational overhead of the proposed method; 3) how to choose the basic clustering algorithms (by using a proposed diversity analysis algorithm); and 4) the reason why the proposed clustering ensemble method is superior to the majority voting method. Both the EDA evaluation on ISCAS89 benchmarks and field-programmable gate array evaluation on Trust-HUB benchmarks are performed to evaluate the performance of the proposed method. Experimental results demonstrate that the proposed method can resist malicious modifications robustly and can detect hardware Trojans with high accuracy (up to 93.75%). Meanwhile, the introduced time overhead is small.

INDEX TERMS Hardware security, hardware Trojan detection, untrustworthy testing parties, unsupervised learning, clustering ensemble.

I. INTRODUCTION

Currently, the design and fabrication process of integrated circuits (ICs) is distributed globally. The third-party intellectual property (IP) cores are also widely used in IC designs [1]. However, this presents serious threats on the security and reliability of ICs. Recently, the malicious modification of ICs, also known as hardware Trojan, has become an emerging security problem in many critical communities. Malicious attackers can alter IC designs or insert hardware Trojans during IC design and fabrication process [2]. When activated, these Trojans can make the system denial-of-service, or create a backdoor to leak sensitive information [3].

It is generally assumed that the testing service in the IC supply chain is trustworthy. However, Yasin *et al.* [4] have illustrated a HackTest technique which can extract secret information contained in the test data. This attack indicates that the testing stage is no longer secure and reliable. The testing house plays an important role in the IC supply chain. Nowadays, the fabricated ICs are only tested by one testing house in the testing stage. If the only testing house colludes with attackers, the detection results will be no longer trusted.

In this paper, we assume that the testing house is untrustworthy. Traditionally, the testing house collects test data and performs the hardware Trojan detection process. In this situation, the testing house can directly modify the results of hardware Trojan detection. In a special case, the testing house only collects the test data and transmits the test data to the designer, then the designer performs the hardware Trojan detection process. In this situation, the testing house can modify the collected test data and mislead the designers' detection of Trojan-inserted ICs.

In the past decade, there are many hardware Trojan detection works, including the side-channel analysis based detection approaches [5]–[9] and logic testing approaches [10], [11]. Some design-for-security approaches are also proposed to facilitate hardware Trojan detection or prevent hardware Trojan insertion, *e.g.*, built-in self-authentication technique to prevent inserting hardware Trojans [12], redundancy-based protection approach based on Trojan tolerance that modifies the application mapping process to provide high-level of protection against Trojans in FPGA [13], the approach promoting smart employment of circuit redundancy to counter Trojans in 3PIP cores [14], and the approach using a selective Triple Modular Redundancy (TMR) scheme to mask the effect of Trojans along with transient errors [15]. Most of them require golden chips or golden models for reference. A few machine learning based hardware Trojan detection methods have been proposed recently. In general, machine learning can be utilized to provide automatic layout identification in reverse engineering-based detection methods, runtime Trojan detection architecture which is trained by Trojan attack behaviors, feature analysis for gate-level netlists, and classification based golden chips-free hardware Trojan detection techniques [16]–[20]. There are only a few works exploiting clustering techniques to detect hardware Trojans which can eliminate the need of golden models [21]–[23]. Çakir and Malik [21] present an information-theoretic approach to evaluate the statistical correlation among the signals in an IC design [21]. Then, they use a clustering algorithm to analyze this correlation to reveal Trojan signals. Salmani [22] computes the controllability and observability features of the logic gates in an IC design. k-means clustering algorithm is used to analyze the inter-cluster distance of these logic gates. It is demonstrated that Trojan gates have significant inter-cluster distance from the genuine gates. Ba *et al.* [23] propose an anomaly detection based technique to identify suspicious signals in a netlist. These suspicious signals may be a part of a Trojan. The above works are focusing on the register-transfer level (RTL) or gate-level of IC designs, which may not be suitable for detecting hardware Trojans inserted in the fabrication stage.

Motivated by the above problems, we have proposed an unsupervised golden models-free hardware Trojan detection method using clustering analysis [24] which will be described in Section II-A as the preliminary. In [24], the major contribution is that we formulate the unsupervised hardware Trojan detection problem into two types of clustering based hardware Trojan detection models, the partition-based model and the density-based model. Then we exploit a clustering algorithm for unsupervised hardware Trojan detection to eliminate the need of simulated golden models and fabricated golden chips. A cluster labeling index (CMI) is proposed to identify which cluster is genuine [24]. The test performance of the method is evaluated with EDA experiments. However, in existing hardware Trojan detection works, including [24], the testing party is considered as trustworthy. Therefore, in the conference version of this work [25], we are the first to consider that the testing party may be untrustworthy, and we propose a clustering ensemble based hardware Trojan detection method to defeat untrustworthy testing parties [25], which can also improve the test performance of the basic clustering algorithms. Besides, in [25], the number of necessary testing parties is analyzed, and the test performance of the proposed method is evaluated with EDA experiments.

This paper is an extended version of the previous conference paper [25]. The main differences between the journal and conference versions are summarized as follows:

1) Two attack models for untrustworthy testing parties are formulated for the first time, and an adversarial test data modification algorithm for the untrustworthy testing party to modify the collected data is proposed. Besides, experiments are performed to evaluate the time overhead of these two attack models. In the first model which represents the common situation, the testing house performs the hardware Trojan detection process, and thus it can directly modify the results of hardware Trojan detection. In the second model, the testing house only collects the test data and the designer performs the hardware Trojan detection process. In this model, the untrustworthy testing house can modify the collected test data and mislead the designer's detection decision on the ICs.
2) To overcome the issues of the process variations and noises in the actual measurement, we introduce a preprocessing step to the method proposed in the conference version [25]. In particular, the unsupervised correlation-based feature selection (UCFS) method is modified and exploited to process the raw data of ICs for feature selection. Comparison experiments with and without preprocessing are performed, and the results show that the detection accuracy of the proposed method is effectively improved.
3) In order to build a clustering ensemble method with better performance on Trojan detection, we exploit the disagreement measure to analyze the diversity among basic clustering algorithms. Based on the disagreement measure, we further propose a diversity-based

clustering algorithm selection method in Section V-C. This algorithm can provide a theoretical basis for ensemble methods to solve the problem of how to choose basic learners to obtain better performance.

4) In [25], simulated ICs during the IC design flow are used to evaluate the proposed method. However, there may be a shift between the SPICE simulation and the actual silicon fabrication. Therefore, we further evaluate the proposed method with FPGA experiments in this paper to better characterize the effects of process variations and noises on hardware Trojan detection in actual measurements. Besides, in the conference version [25], we only evaluate the method with small benchmarks (ISCAS89 benchmark circuits), and the Trojans used in the evaluations are not real Trojans with meaningful attack functions. In this journal version, we use much bigger designs and real Trojans with meaningful attack functions (Trust-Hub benchmarks [26]) for evaluations.
5) In [25], we have only theoretically analyzed the number of necessary testing parties. In Section V-B, we theoretically analyze the time and computational overhead introduced by the increased number of testing parties. Moreover, we have presented the experimental analysis of the time overhead of the proposed method in both EDA evaluation and FPGA evaluation in Section VI-D.
6) In Section V-D, we illustrate the reason why the proposed clustering ensemble method is superior to the majority voting method. Meanwhile, we also experimentally compare the test performance of the proposed method with the majority voting method using both EDA and FPGA evaluations in Section VI.

The rest of this paper is organized as follows. Preliminary knowledge is described in Section II. The problem formulation and attack models for untrustworthy testing parties are presented in Section III. The proposed clustering ensemble based hardware Trojan detection method (CE-HTDM) is elaborated in Section IV. The number of necessary testing parties, the time and computational overhead of the proposed method, the clustering algorithm selection based on the proposed diversity-based algorithm, and the reason for using clustering ensemble rather than majority voting, are theoretically analyzed in Section V. Both the EDA and FPGA evaluations are provided in Section VI. We conclude the paper in Section VII.

II. PRELIMINARY

A. UNSUPERVISED HARDWARE TROJAN DETECTION METHOD USING CLUSTERING ANALYSIS

In unsupervised hardware Trojan detection scenario, a clustering technique is used to divide unlabeled ICs into two disjoint subsets. Each subset is referred as a cluster [27]. In [24], we have formulated the unsupervised hardware Trojan detection problem into two types of clustering based detection models, *i.e.*, the partition-based model and the density-based model.

The partition-based hardware Trojan detection model is illustrated in Fig. 1. It divides unlabeled ICs into two mutually independent clusters. However, it is unable to determine the label (Trojan-free or Trojan-inserted) of these two clusters. Therefore, a cluster labeling index (*CMI*) is proposed to identify which cluster is genuine [24]. Both theoretical analysis and experimental results demonstrate that the genuine cluster has a larger *CMI* than the Trojan-inserted cluster.

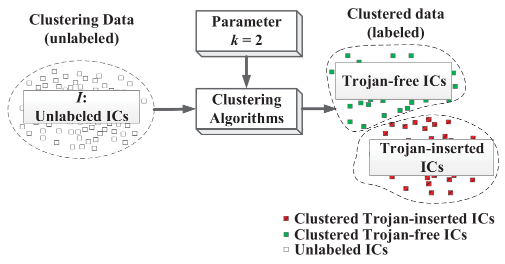

FIGURE 1. **The partition-based hardware Trojan detection model [24].**

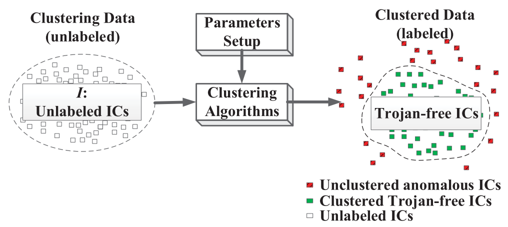

FIGURE 2. **The density-based hardware Trojan detection model [24].**

The density-based hardware Trojan detection model is illustrated in Fig. 2. Each IC can be represented as a point in the high-dimensional feature space. The theoretical analysis shows that genuine ICs are usually grouped together, while Trojan-inserted ICs are always scattered. The proposed technique can form a boundary around the population of genuine ICs [24]. The ICs within the boundary are considered as Trojan-free ICs, while the others outside the boundary are considered as Trojan-inserted ICs.

B. CLUSTERING ENSEMBLE

Clustering ensemble is a kind of ensemble method in which the basic learners are clustering algorithms [28]–[31]. It can be divided into the following four categories:

1) Similarity-based methods: A similarity-based method obtains a similarity matrix for each clustering result first. Then these similarity matrices are averaged to form the consensus similarity matrix that generates the ensemble result [28].
2) Graph-based methods: In graph-based clustering ensemble methods, an undirected graph is constructed to integrate the clustering information conveyed by all basic clustering results [29]. Then graph partitioning of the undirected graph is performed to identify the ensemble results.

3) Relabeling-based methods: The basic idea of relabeling-based methods is to align the cluster labels of all basic clustering results, so that the same label denotes similar clusters across the basic clustering results [30]. Then, for each instance, its ensemble result is obtained by performing majority voting method on its basic aligned cluster labels.
4) Transformation-based methods: A transformation-based method re-represents each instance as an r-tuple, where r is the number of basic clustering algorithms and the qth element is its cluster label given by the qth clustering algorithm [31]. Then it performs meta-clustering analysis over the transformed r-tuples to obtain the ensemble result.

Overall, the general process of clustering ensemble methods can be summarized as follows. Given a data set $D = \{d_1, d_2, \ldots, d_m\}$, where the p_{th} instance d_p is a n-dimensional feature vector, the clustering ensemble methods consist of two steps [28]–[31]:

1) Clustering generation: Given r basic clustering algorithms, each algorithm divides D into k clusters. For the q_{th} clustering algorithm $\partial^{(q)}(1 \leq q \leq r)$, its clustering result can be represented by a label vector $\lambda^{(q)} \in N^m$, where the pth element $\lambda_p^{(q)} \in \{1, 2, \ldots, k^{(q)}\}$ indicates the clustering label of the instance $d_p(1 \leq p \leq m)$.
2) Clustering combination: In this step, a consensus function is used to consolidate the basic clustering results into the final ensemble result.

As the calculation and transformation process of similarity-based methods and graph-based methods are more complex, we prefer to use the relabeling-based methods and transformation-based methods. However, the relabeling-based methods and transformation-based methods have their own drawbacks. For example, relabeling-based methods obtain the final ensemble result by performing majority voting. However, the ensemble result obtained by majority voting is not as good as the ensemble result obtained by the ensemble strategy of transformation-based methods. Meanwhile, transformation-based methods do not align the cluster labels of all basic clustering results before transformation. As a result, the final ensemble result of transformation-based methods obtained without the relabeling process is not as good as the ensemble result obtained with an additional relabeling process. In summary, the advantages and disadvantages of relabeling-based methods and transformation-based methods are complementary. Therefore, in this paper, we propose a novel hybrid clustering ensemble method, which combines relabeling-based methods and transformation-based methods. The proposed hybrid clustering ensemble method works as follows. Firstly, each testing party performs a clustering based hardware Trojan detection method on unlabeled ICs to obtain basic clustering results. After that, each testing party relabels the cluster labels of these basic clustering results by calculating the proposed cluster labeling index (CMI). Then the label information of basic clustering results is expressed as features for each IC. Lastly, the ensemble result is obtained via a categorical clustering technique.

III. PROBLEM FORMULATION AND ATTACK MODELS OF UNTRUSTWORTHY TESTING PARTIES

A. PROBLEM FORMULATION

During the test-time detection, a batch of unlabeled fabricated ICs are under test. Here, "unlabel" means an IC is under test and it is still unclear whether the IC contains Trojans or not, while "label" means the IC is marked as Trojan-free or Trojan-inserted after Trojan detection [32]. The problems of unsupervised hardware Trojan detection without and with untrustworthy testing parties can be formulated as follows, respectively:

Definition 1 (Unsupervised hardware Trojan detection): Given M fabricated ICs under test, denoted as $IC_p(1 \leq p \leq M)$, each IC has a few features extracted from its power traces. These ICs can be represented as points in the high-dimensional feature space. They can be divided into two clusters, Trojan-free and Trojan-inserted. The problem is to identify the correct cluster of IC_p using a clustering method.

Definition 2 (Unsupervised hardware Trojan detection against untrustworthy testing parties): Given R testing parties, denoted as $TP_q(1 \leq q \leq R)$, each one detects Trojans with a clustering algorithm. Assume that there are Q untrustworthy testing parties (UTPs), where $0 < Q \leq R$. These untrustworthy testing parties will collude with malicious attackers and modify their own detection results. The problem is to ensure reliable detection results under untrustworthy testing parties.

B. ATTACK MODELS

Traditionally, the testing house collects test data and then performs the hardware Trojan detection process. However, in theory, there can be a special case: the testing house only collects the test data and transmits the test data to the designer, and then the designer performs the hardware Trojan detection process.

If the testing house performs the hardware Trojan detection process, it can directly modify the results of hardware Trojan detection. If the testing house only collects test data and the designer performs the hardware Trojan detection process, the testing house needs to modify the collected test data and mislead the designer's testing results on the ICs. Therefore, we now formulate the attack models of these two cases.

1) ATTACK MODEL A (MODIFY LABEL)

In this case, the testing house performs the hardware Trojan detection process, and thus it can directly modify the results of hardware Trojan detection. The detailed attack process is shown in Fig. 3.

Given M fabricated ICs under test, denoted as $IC_p(1 \leq p \leq M)$, the testing house collects the raw power trace of each IC, and then extracts some features from raw power traces. For the i_{th} IC (IC_i), the testing house uses a kind

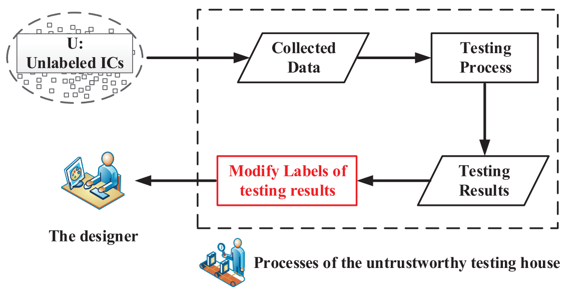
U:
Unlabeled ICs
Collected Data
Testing Process
Testing Results
Modify Labels of testing results
The designer
Processes of the untrustworthy testing house

FIGURE 3. Overall flow of the Attack Model A.

of hardware Trojan detection method to verify the IC and then labels the IC accordingly. The obtained label can be represented as $y_i \in \{0, 1\}$, where $y_i = 0$ represents that the i_{th} IC is Trojan-free, and $y_i = 1$ represents that the i_{th} IC is Trojan-inserted. In this case, the testing house controls the hardware Trojan detection method and detection results. It can directly modify the label of each IC. For the i_{th} IC, if its corresponding label is $y_i = 1$, the testing house will change the label from 1 to 0. Finally, the untrustworthy testing house transmits the modified detection results to the designer. In this way, the untrustworthy testing house can help Trojan-inserted ICs to escape detection.

2) ATTACK MODEL B (MODIFY COLLECTED TEST DATA)

In this case, the testing house only collects test data and the designer performs the hardware Trojan detection process. The untrustworthy testing house can only modify the collected test data to mislead the designer's detection on these ICs. The detailed attack process is illustrated in Fig. 4. In this paper, we further propose an algorithm for the untrustworthy testing house to modify the collected test data, as shown in Algorithm 1.

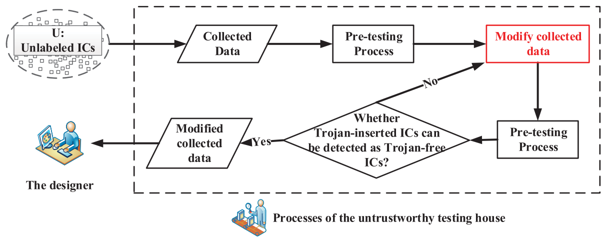

FIGURE 4. Overall flow of the Attack Model B.

Given M fabricated ICs under test, denoted as $IC_p (1 \leq p \leq M)$, the testing house collects the raw power trace of each IC, and then extracts some features from raw power traces to construct the test set $I = \{IC_1, IC_2, \ldots, IC_m\}$. In this case, the untrustworthy testing house can use an iterative adversarial data generation method to modify the collected test data, as shown in Algorithm 1. The input of the algorithm is an IC set I, a pre-detection method $\hat{k}$, a desired accuracy τ on perturbed ICs and a desired perturbation parameter ξ. The parameter ξ controls the magnitude of the perturbation

Algorithm 1 Adversarial Test Data Modification Algorithm

```
Input: IC set $I$, pre-detection method $\hat{k}$, desired accuracy $\tau$
    on perturbed ICs and desired perturbation parameter $\xi$.
Output: Modified IC set $I$
 1: Perform pre-detection method $\hat{k}$ on $I$.
 2: Select the ICs whose detection label is 1 to construct data
    set $\hat{I}$.
 3: for $IC_i \in \hat{I}$ do
 4:     Initialize perturbation vector $v \leftarrow 0$ and initialize
        minimal perturbation $\Delta v_i$ randomly.
 5:     $v \leftarrow P_{P,\xi}(v + \Delta v_i)$
 6:     Perform pre-detection method $\hat{k}$ on $IC_i + v$.
 7:     Obtain $Confidence(IC_i + v)$.
 8:     while $Confidence(IC_i + v) \leqslant \tau$ or $\hat{k}(IC_i + v) = \hat{k}(IC_i)$
        do
 9:         Compute the minimal perturbation that sends $IC_i + v$
            to the decision boundary:
            $\Delta v_i \leftarrow \arg\min_r ||r||_2 s.t. \hat{k}(IC_i + v + r) \neq \hat{k}(IC_i)$
10:         Update the perturbation:
            $v \leftarrow P_{P,\xi}(v + \Delta v_i)$
11:         Perform pre-detection method $\hat{k}$ on $IC_i + v$.
12:         Obtain $Confidence(IC_i + v)$.
13:         $IC_i = IC_i + v$
14:     end while
15: end for
```

vector. First, the untrustworthy testing house performs a pre-detection method on I. Then it selects the ICs whose label is 1 to construct a new data set $\hat{I}$. The ICs in $\hat{I}$ are considered as Trojan-inserted which need to be modified. The algorithm modifies the data of the IC by introducing a perturbation which acts like process variations but aims at covering the impact of a Trojan.

The algorithm is iteratively performed on $\hat{I}$. For each IC in $\hat{I}$, it eventually generates one corresponding perturbation. For the i_{th} IC (IC_i), the algorithm initializes a perturbation vector $v = 0$ and randomly initializes a minimal perturbation Δv_i. At each iteration, the algorithm updates the perturbation vector $v \leftarrow P_{P,\xi}(v + \Delta v_i)$ first. Then it performs the pre-detection method to obtain the detection label and confidence level of the perturbed point $IC_i + v$. If the current perturbation v does not make the perturbed point $IC_i + v$ fool the pre-detection method, the algorithm enters the next round. In the new round, it updates the minimal perturbation Δv_i by solving the following optimization problem [33]:

$$\Delta v_i \leftarrow \arg\min_r ||r||_2 \ s.t. \hat{k}(IC_i + v + r) \neq \hat{k}(IC_i) \qquad (1)$$

Then the algorithm updates the perturbation vector v and examines whether the new perturbed point $IC_i + v$ can fool the pre-detection method or not. For the i_{th} IC in $\hat{I}$, when the confidence level (obtained from the pre-detection method) satisfies $Confidence(IC_i + v) \geqslant \tau$ and the detection label of the perturbed IC has changed from 1 to 0 (indicating that the Trojan-inserted IC is detected as a Trojan-free IC), it stops modifying the test data of the current IC and begins to modify the test data of the next IC in $\hat{I}$.

Note that, motivated by the adversarial machine learning technique in the field of artificial intelligence [33], we propose an adversarial test data modification algorithm for the untrustworthy testing house in the hardware Trojan detection scenarios, as shown in Algorithm 1. There are several differences between the proposed method and the method in [33]. First, the final outputs of these two methods are different. The function of the technique proposed in [33] is to generate and output only a perturbation vector for an input image. Only a perturbation vector is output. This means it is not a complete process for modifying the input data. However, the output of our algorithm is the carefully modified test data of an IC set. In other words, our algorithm is a complete process for modifying the input data. Second, the technique proposed in [33] is an untargeted adversarial perturbation generation method, while our method is a targeted test data modification method. The technique proposed in [33] aims at causing a natural image to be misclassified, but it does not cause the image to be misclassified into a specific category of images. However, our method aims at causing the input data to be misclassified into a specific category of ICs, *i.e.*, a Trojan-inserted IC to be misclassified as a Trojan-free IC. Third, the execution process, loop structure, and decision conditions of these two methods are different. We first add an outermost loop to implement the modification of each IC. Besides, for the inner loop, the decision conditions of these two methods are different. We add a decision condition ($Confidence(IC_i + v)$) to determine whether the modified IC is identified as a Trojan-free IC. The only similarity between the two methods is the optimization equation (Equation (1)). To the best of our knowledge, this work is the first to propose an adversarial test data modification algorithm for an untrustworthy testing house to modify the collected test data of ICs to mislead hardware Trojan detection.

3) EXPERIMENTAL ANALYSIS

We perform an experiment to evaluate the time overhead of the two attack models. The process of modifying detection label (in Attack Model A) and collected data (in Attack Model B) is executed using PyCharm 2017.2.3. The pre-detection method used in Attack Model B runs in WEKA 3.6. These experiment tools run on the Intel(R) Core(TM) i5-4590 CPU. The frequency of the CPU is 3.30GHz. The log file of Pycharm and WEKA are obtained which record the time overhead of the processes in Attack Model A and Attack Model B. For Attack Model B, the time overhead of the pre-detection method in each round is much larger than the time overhead of modifying test data. Therefore, in each round, we use the time overhead of the pre-detection method to represent the total time overhead of that round.

As shown in Fig. 5, the time overhead of Attack Model A and Attack Model B are compared when 5% of the test data is modified. For Attack Model B, it takes several rounds of iterations to modify the test data in order to successfully

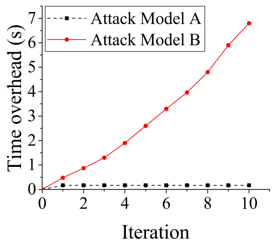

FIGURE 5. The comparison of the time overhead of the two attack models.

mislead the designer's detection results. It is shown that the time required for 10 iterations is about 7s. However, in Attack Model A, the untrustworthy testing party only needs to perform one modification process of the labels, and the modification time is 0.17s. Since the Attack Model B introduces large time overhead and computational overhead for iteratively modifying the test data, the Attack Model A is more favored by the attackers.

Overall, the final results of these two attack models are equivalent. The untrustworthy testing house will eventually mislead the designer's Trojan detection results. Therefore, in the subsequent sections, the theoretical analysis and experimental evaluations are presented under the Attack Model A.

IV. PROPOSED CLUSTERING ENSEMBLE BASED HARDWARE TROJAN DETECTION METHOD (CE-HTDM)

A. OVERALL FLOW OF THE PROPOSED METHOD

The overall flow of the proposed hybrid clustering ensemble based hardware Trojan detection method (CE-HTDM) is shown in Fig. 6. We assume that there are three testing parties and one of them is untrustworthy. Transient power traces of ICs are collected for testing. CE-HTDM consists of three phases, namely, preprocessing, clustering generation and clustering combination, as shown in Algorithm 2. Firstly, the raw data of power traces is preprocessed to extract predominant power features of each IC. Then the basic detection results from each testing party are obtained. Finally, the hybrid ensemble method is used to consolidate all the basic detection results, and obtain the final detection results.

B. PREPROCESSING OF CE-HTDM: UNSUPERVISED CORRELATION-BASED FEATURE SELECTION

The process variations and noises in the actual measurement can cover the Trojan's impact on IC's parameters. Therefore, we use an advanced signal processing method to preprocess raw power traces of ICs. In this paper, the unsupervised correlation-based feature selection (UCFS) method [34] was modified, and creatively applied in the field of hardware Trojan detection for the first time. It can filter out noisy and redundant features, and retain predominant features which can help to facilitate Trojan detection.

The process of the UCFS method is recapitulated in Algorithm 3. Given an IC set $I = \{IC_1, IC_2, \ldots, IC_m\}^T$, where $IC_i = \{f_{i1}, f_{i2}, \ldots, f_{in}\}$, $1 \le i \le m$. IC_i represents the

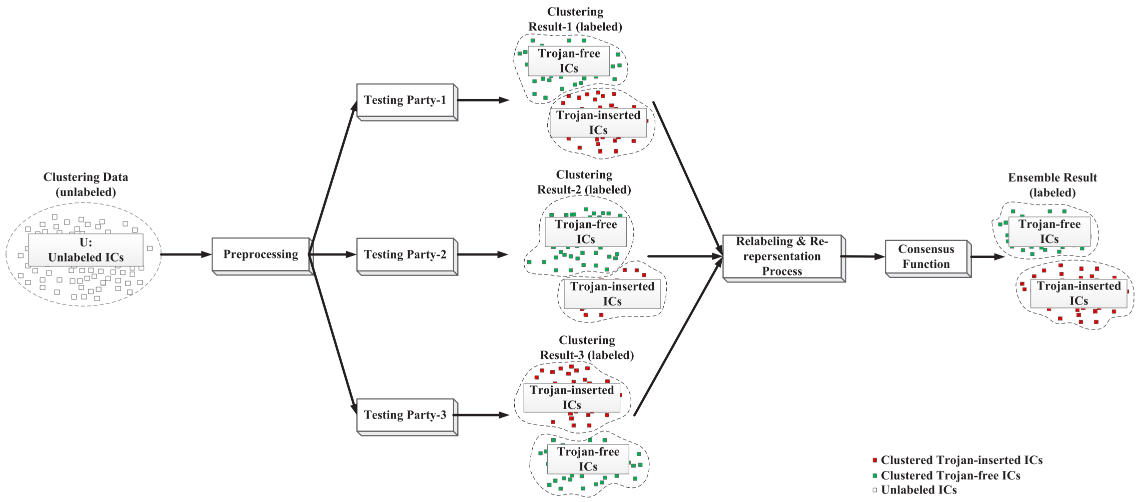
Clustering Data (unlabeled)
U: Unlabeled ICs
Preprocessing
Testing Party-1
Testing Party-2
Testing Party-3
Clustering Result-1 (labeled)
Trojan-free ICs
Trojan-inserted ICs
Clustering Result-2 (labeled)
Clustering Result-3 (labeled)
Relabeling & Re-repersentation Process
Consensus Function
Ensemble Result (labeled)
🟥 Clustered Trojan-inserted ICs
🟩 Clustered Trojan-free ICs
□ Unlabeled ICs

FIGURE 6. Overall flow of the proposed clustering ensemble based hardware Trojan detection method (CE-HTDM) for three testing parties [25].

i_{th} IC under test which has n features. Since the UCFS method handles features, the IC set I can be further represented by using its feature vectors, which can be denoted as $I = \{f_1, f_2, \ldots, f_n\}$. There are four main parameters in this algorithm, δ, SU, F_{list} and F_{best}. δ is a predefined threshold for selecting features. SU is the symmetrical uncertainty value of each feature. SU is a normalized information theoretic measure which uses entropy and conditional entropy values to calculate dependencies of features. The features whose SU value is greater than the threshold δ are stored in F_{list}. The final feature selection results are stored in F_{best}.

The UCFS process consists of two parts. In the first part, it calculates the SU value for each feature [34]. After that, it selects relevant features based on the predefined threshold δ and puts them into F_{list}. Then it sorts the selected features in descending order according to their SU values. In the second part, it removes redundant features from the ordered list F_{list}. The iteration starts from the first element f_p in F_{list} and continues as follows [34]. For all the remaining features, the algorithm selects the feature f_q which is right next to f_p as the first feature. If f_q is a redundant peer to f_p [34], f_q will be removed from F_{list} and the algorithm updates f_q with the feature right next to the current f_q. After one round of filtering based on f_p, the algorithm takes the first feature in the remaining features, which is right next to f_p, as the new f_p. Then it repeats the previous filtering process. The algorithm stops until there are no more features to be removed from F_{list}. Finally, all selected features are saved in F_{best}.

Fig. 7 shows an example of preprocessing results of the genuine Advanced Encryption Standard (AES) [35] benchmark circuit, where Fig. 7(a) is the original power trace and Fig. 7(b) is the preprocessed features of the power trace. For each IC, the features extracted from the preprocessing process are still in the time domain. It is shown that the UCFS method mainly extracts the features on the wave troughs of the raw power trace, while only few features on the wave crests of the power trace are retained. This indicates that the features on the wave crests of the power trace are redundant or highly correlated, which are useful for Trojan detection.

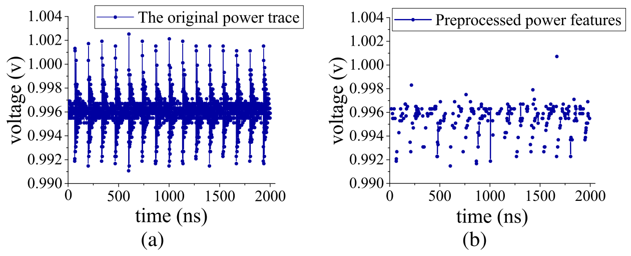

FIGURE 7. Preprocessing results of the genuine AES benchmark circuit. (a) The original power trace. (b) Preprocessed power features.

Note that, compared with the original UCFS method, this method has been modified as follows. First, we divide the execution phase of the algorithm into two phases. In the first phase, it calculates the correlation between an IC's features. In the second phase, it selects the features which meet the specific requirements. Second, we have changed the data object processed by the algorithm. The algorithm is modified to process raw data of power traces of ICs. Overall, the contribution of the UCFS based IC feature selection method is as follows. The process variations and measurement noises during the actual measurement have significant impacts on the existing side-channel signal analysis based hardware Trojan detection methods. We are the first to analyze the correlation between the features of an IC for feature selection, thus to alleviate the impact of process variations and noises on hardware Trojan detection in the actual measurement. Besides, since the UCFS based method is an unsupervised feature selection method, this work provides a beneficial attempt and a useful reference for hardware Trojan detection works without the need of golden fabricated chips or golden simulated models.

Algorithm 2 Clustering Ensemble Based Hardware Trojan Detection Method

```
Input: IC set $I = \{IC_1, IC_2, \ldots, IC_m\}^T$;
       Basic testing parties $\Lambda = \{\partial^{(1)}, \partial^{(2)}, \ldots, \partial^{(r)}\}$, each
       divides $I$ into $k$ clusters.
Process:
 1: // Preprocessing
 2: UCFS($I$)
 3: // Clustering Generation
 4: for $i = 1$ to $m$ do
 5:     for $j = 1$ to $r$ do
 6:         $\partial_i^{(j)'} = CMI(\partial_i^{(j)})$
 7:         $y_j^i = \partial_i^{(j)'} \in \{0, 1, \ldots, k\}$
 8:     end for
 9:     $\varphi(IC_i) = (y_1^i, y_2^i, \ldots, y_r^i)$
10: end for
11: // Clustering Combination
12: $I = (\varphi(IC_1), \varphi(IC_2), \ldots, \varphi(IC_m))^T$
13: Initialize ensemble clusters $\lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_m\}$
14: for $l = 1$ to $r$ do
15:     for $i = 1$ to $r$ do
16:         $E[l][i] = SIM(\varphi(IC)_l, \varphi(IC)_i)$
17:         $U[l] = 1$ (keeps track of active clusters)
18:     end for
19: end for
20: for $i = 1$ to $r - 1$ do
21:     $< i, n >= \arg\max_{\{<i,n>: i \neq n \wedge I[i]=1 \wedge I[n]=1\}} E[i][n]$
22:     $\lambda[< i, n >]$ (stores merge clusters)
23:     for $j = 1$ to $m$ do
24:         $E[i][j] = SIM(\varphi(IC)_i, \varphi(IC)_n, \varphi(IC)_j)$
25:         $E[j][i] = SIM(\varphi(IC)_i, \varphi(IC)_n, \varphi(IC)_j)$
26:     end for
27:     $U[n] = 0$ (deactivates cluster)
28: end for
29: return Ensemble hardware Trojan detection result $\lambda$.
```

C. CE-HTDM: CLUSTERING GENERATION

The proposed novel hybrid clustering ensemble method consists of two phases: clustering generation and clustering combination. In the clustering generation phase, there are r testing parties. For the same batch of ICs under test, each testing party uses a clustering algorithm to perform Trojan detection and outputs a set of basic detection results.

Firstly, the basic clustering result is obtained by each testing party. In a basic clustering result, there are two clusters. Then, each testing party exploits the cluster labeling index (CMI) to determine the cluster label of these two clusters. In this way, each testing party outputs two clusters, in which, one cluster is labeled as Trojan-inserted while the another cluster is labeled as Trojan-free. Lastly, the detection result of each testing party is transmitted to the clustering combination phase for further conversion.

Algorithm 3 Unsupervised Correlation-Based Feature Selection (UCFS) [34] Based IC Feature Selection Method

```
Input: $I = \{f_1, f_2, \ldots, f_n\}$; // an IC set represented by its
       feature vectors;
           $\delta$; // a predefined threshold
Output: $F_{best}$; // an optimal subset
 1: // First part
 2: for $i = 1$ to $N$ do
 3:     calculate $SU_i$ for $f_i$
 4:     if $SU_i > \delta$ then
 5:         append $f_i$ to $F_{list}$
 6:     end if
 7: end for
 8: order $F_{list}$ in descending $SU_i$ value
 9: // Second part
10: $f_p = getFirstElement(F_{list})$
11: for $f_P \neq NULL$ do
12:     $f_q = getNextElement(F_{list}, f_p)$
13:     for $f_q \neq NULL$ do
14:         $f_q{}' = f_q$
15:         if $SU_p \geq SU_q$ then
16:             remove $f_q$ from $F_{list}$
17:             $f_q = getNextElement(F_{list}, f_q{}')$
18:         else
19:             $f_q = getNextElement(F_{list}, f_q)$
20:         end if
21:     end for
22:     $f_p = getNextElement(F_{list}, f_p)$
23: end for
24: $F_{best} = F_{list}$
```

D. CE-HTDM: CLUSTERING COMBINATION

In the combination phase, firstly, each IC is represented as a r-tuple using the labels of its basic detection results, where r is the number of testing parties. The qth element in the r-tuple indicates the cluster label given by the qth testing party. Each transformed r-tuple $\varphi(IC) = (y_1, y_2, \ldots, y_r)$ can be regarded as a label vector, where $y_q \in K^{(q)} = \{1, 2, \ldots, k^{(q)}\}(q = 1, \ldots, r)$. Then a categorical clustering technique is exploited to analyze the set of transformed ICs to obtain the ensemble hardware Trojan detection result. In this paper, hierarchical agglomerative clustering (HAC) method [27] is exploited to partition the set of transformed ICs.

These transformed ICs should be clustered into two mutually independent clusters. Therefore, the parameter k is set to be 2. The set of transformed ICs is $I = (\varphi(IC_1), \varphi(IC_2), \ldots, \varphi(IC_m))^T$, which is the input of the hierarchical clustering algorithm. In the hierarchical agglomerative clustering algorithm [27], λ is a list used to store the clustering result. U is an array which records the state value (1 indicates activated cluster while 0 indicates deactivated cluster) of each cluster. SIM is a function which calculates the similarity among activated clusters. The algorithm calculates the similarity among each clusters to obtain the $r \times r$ similarity matrix E first, then $r-1$ iterations are executed to merge the most similar clusters. In each iteration, the most similar two clusters are merged into one cluster. Then the state value of these two clusters in U is set to be 0. It means that these two clusters are no longer activated clusters. In the similarity matrix E, the similarity values in the rows and columns of the merged cluster are updated. Then the algorithm proceeds to the next iteration. When the desired clusters are obtained, the algorithm terminates. The final clustering result is stored in λ, which is the ensemble hardware Trojan detection result.

V. THEORETICAL ANALYSIS AND DISCUSSION

In this section, four problems and questions are discussed: 1) the number of necessary testing parties of the CE-HTDM method; 2) the time and computational overhead of the CE-HTDM method; 3) the proposed diversity analysis algorithm to select basic clustering algorithms; 4) the reason why the proposed hybrid clustering ensemble method is superior to the majority voting method.

A. THE NUMBER OF NECESSARY TESTING PARTIES OF CE-HTDM

In this subsection, the number of required testing parties of the proposed method is analyzed. In general, the number of testing parties is influenced by the number of untrustworthy testing parties. In practical detection scenarios, there are two cases. In the first case, there are no trustworthy testing parties for cooperation. The total number of required testing parties is determined based on the majority rule. In other words, if there are n untrustworthy testing parties, a total of $2n+1 (n \geq 1)$ testing parties are needed in the proposed method.

In the second case, there are trustworthy testing parties for cooperation. Assuming that there are n testing parties with unknown trustworthiness. For security purpose, the method requires that the number of trustworthy testing parties is not less than the number of testing parties whose trustworthiness are unknown. Therefore, n trustworthy testing parties are introduced to build the proposed method. Hence, the total number of testing parties is $2n (n \geq 1)$.

Obviously, the second case is more demanding than the first case in actual hardware Trojan detection scenarios. Therefore, in this paper, we analyze the overhead and perform experimental evaluations based on the first case. In other words, we assume that the number of untrustworthy testing parties is n, and the number of required testing parties are $2n+1$.

B. TIME OVERHEAD AND COMPUTATIONAL OVERHEAD OF CE-HTDM

In this section, we theoretically analyze the time and computational overhead introduced by the increased number of testing parties. Besides, the experimental analysis of the time overhead of the proposed method in both EDA and FPGA evaluations will be presented in Section VI-D. Overall, both theoretical analysis and experimental results show that the CE-HTDM method does not introduce significant overhead.

Firstly, we discuss the time overhead of CE-HTDM. As shown in Fig. 8, the time overhead of the CE-HTDM method includes the time of the following five steps: collecting test data, preprocessing, clustering generation, process of relabeling and transforming, and clustering combination. We assume that there are M ICs under test. In the phase of testing data collection, the time of collecting one power trace of one IC is 5×10^{-6}s (obtained from oscilloscope measurement). Therefore the time overhead of collecting test data is $t_c = M \times 5 \times 10^{-6}$s. In the preprocessing phase, the time overhead is t_p which is usually less than 0.1s. As the testing parties in the CE-HTDM method are independent of each other, they can perform the hardware Trojan detection process in parallel. We define the time overhead of the i_{th} testing party performing clustering generation as tp_i. The time overhead of the clustering generation phase is $t_{cg} = \max\{tp_1, tp_2, \ldots, tp_{2n+1}\}$. The time overhead of relabeling and transforming the results can be represented as $t_{R\&T}$ which is also less than 0.1s normally. In the clustering combination phase, since we use the HAC method to obtain the final ensemble result, the time overhead of the clustering combination phase can be represented as $t_{cc} = t_{HAC}$. Therefore, the overall time overhead of CE-HTDM, denoted as $t_{CE-HTDM}$, is $t_{CE-HTDM} = t_c + t_p + t_{cg} + t_{R\&T} + t_{cc}$. In particular, experiments show that the time overhead of t_{cg} and t_{cc} is much larger than the other three parts. Therefore, we can ignore the time overhead of the other three parts and calculate the time overhead of CE-HTDM using $t_{CE-HTDM} \approx t_{cg} + t_{cc}$.

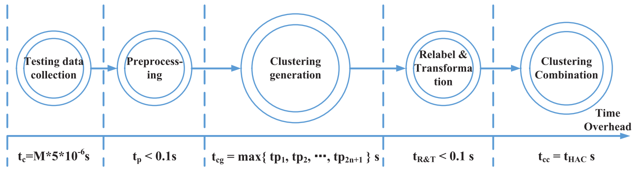

FIGURE 8. The analysis of the time overhead of CE-HTDM.

We now discuss the time overhead of the traditional hardware Trojan detection scenario. That is, there is only one testing house performs a kind of detection method. The time overhead can be represented by $t_{Tra} = t_c + t_p + t_{cg} \approx t_{cg}$.

In conclusion, compared with a traditional hardware Trojan detection method, the CE-HTDM method has mainly increased the time overhead of t_{cc}. Experimental evaluations show that t_{cc} is normally in a range of 0.02s to 0.05s. Therefore, the increased time overhead of the CE-HTDM method is small and acceptable.

Secondly, we discuss the computational overhead of the CE-HTDM method. The overall flow for one testing house to perform one testing process includes the following three steps: data collection, preprocessing, and clustering generation. The computational overhead of collecting test data can be considered as zero. The computational overhead of the preprocessing and clustering generation can be denoted as s_p and s_{cg}, respectively. As a result, the computational overhead for one test house to perform one testing process is $s = s_p + s_{cg}$. This is the computational overhead of a traditional method, *i.e.*, $s_{Tra} = s$.

For the CE-HTDM method, each testing party performs the process of data collection, preprocessing, and clustering generation, separately. Therefore, the computational overhead of the i_{th} testing party can be represented by $s_i = s_p + s_{cg} = s$. Before clustering combination, the time overhead of relabeling and transforming can be denoted as $s_{R\&T}$. At last, the computational overhead of clustering combination can be denoted as s_{cc}. Therefore, the computational overhead of CE-HTDM is $s_{CE-HTDM} = \sum_{i=1}^{2n+1} s_i + s_{R\&T} + s_{cg} \approx (2n+1)s + s_{R\&T} + s_{cc} \gg s_{Tra}$. Obviously, compared to a traditional method, CE-HTDM has increased large computational overhead. However, the computational overhead of CE-HTDM is distributed in each testing party. This means that CE-HTDM does not increase the storage and computing overhead for each testing party.

C. CLUSTERING ALGORITHM SELECTION BASED ON THE PROPOSED DIVERSITY ANALYSIS ALGORITHM

In this section, we discuss how to select the basic clustering algorithms to build the proposed CE-HTDM with good performance on Trojan detection. The disagreement measure is calculated, which shows the difference between the clustering results of each clustering algorithm. Based on the disagreement measure metric, we propose a diversity-based clustering algorithm selection method.

Given an IC set $I = \{IC_1, IC_2, \ldots, IC_m\}$, where $IC_i = \{f_{i1}, f_{i2}, \ldots, f_{in}\}$, $1 \leq i \leq m$. IC_i represents the i_{th} IC under test which has n features. During hardware Trojan detection, a label for each IC can be obtained after trust verification. The label is denoted by $y \in \{0, 1\}$, where Trojan-free is represented as 0 and Trojan-inserted is represented as 1. Considering two clustering algorithms, denoted by ∂_A and ∂_B, a contingency table of these two clustering algorithms is calculated to obtain the disagreement measure between them. The contingency table is shown in Table 1, where a indicates the number of ICs that both clustering algorithms ∂_A and ∂_B predict as Trojan-free; b indicates the number of ICs that the clustering algorithm ∂_A predicts as Trojan-free but ∂_B predicts as Trojan-inserted; c indicates the number of ICs that the clustering algorithm ∂_A predicts as Trojan-inserted but ∂_B predicts as Trojan-free; d indicates the number of ICs that both clustering algorithms ∂_A and ∂_B predict as Trojan-inserted. Obviously, $a + b + c + d = m$. Based on the contingency table, the disagreement measure can be

TABLE 1. The contingency table of clustering algorithms ∂_A and ∂_B.

	$\partial_A = 0$	$\partial_A = 1$
$\partial_B = 0$	a	c
$\partial_B = 1$	b	d

calculated as follows:

$$dis_{\partial_A \partial_B} = \frac{b + c}{m} \tag{2}$$

where $dis_{\partial_A \partial_B} \in [0, 1]$. The larger the value of $dis_{\partial_A \partial_B}$ is, the greater the diversity between the two clustering algorithms is. In this way, the basic clustering algorithms with larger diversity are selected to build the CE-HTDM.

We propose a diversity-based selection algorithm, as shown in Algorithm 4, where $I = [IC_1, IC_2, \ldots, IC_m]^T$ is a given IC set, $\partial = \{\partial_1, \partial_2, \ldots, \partial_R\}$ is the set of basic clustering algorithms, r is the predefined number of selected clustering algorithms, and θ is a predefined accuracy threshold. The final result will be stored in the selected clustering algorithm set $\hat{\partial}$.

The proposed diversity-based selection algorithm mainly consists of three parts. In the first part, each basic clustering algorithm is performed on I and the corresponding detection result is obtained. For the i_{th} clustering algorithm, if its detection accuracy is larger than a predefined accuracy threshold θ, its detection result is stored in $Y_i = [y_1, y_2, \ldots, y_m]^T$, where $y_j \in \{0, 1\}$, $1 \leqslant j \leqslant m$. Otherwise, its detection result is set to be $Y_i = NULL$, which means, the detection result is discarded.

In the second part, first, the disagreement measure matrix $DIS_{R\times R}$ and the average disagreement measure vector $aver_{1\times R}$ are initialized to be 0. Note that, $DIS_{R\times R}$ stores the disagreement measure between arbitrary two basic clustering algorithms in ∂. $aver_{1\times R}$ stores the average disagreement measure of each basic clustering algorithm in ∂. For the i_{th} basic clustering algorithm $\partial_i(\partial_i \in \partial, 1 \leqslant i \leqslant R)$, the disagreement measure between ∂_i and the j_{th} basic clustering algorithm $\partial_j(\partial_j \in \partial, 1 \leqslant j \leqslant R)$ is calculated and stored in $DIS_{R\times R}$. Then, for the i_{th} basic clustering algorithm $\partial_i(\partial_i \in \partial, 1 \leqslant i \leqslant R)$, its average disagreement measure is calculated by $\frac{1}{R}\sum_{j=1}^{R} DIS[i][j]$ and stored in $aver$.

In the third part, the selected clustering algorithm set $\hat{\partial}$ is initialized to be empty. Then, the clustering algorithm in ∂ which has the maximum value in $aver$ is selected. Then the selected algorithm is removed from ∂ and added to $\hat{\partial}$. After that, there are still $r - 1$ algorithms waiting to be selected from ∂, where r is a predefined number. Then a greedy strategy is exploited to select $r - 1$ clustering algorithms from ∂. The greedy strategy based method works as follows. In the $l_{th}(1 \leqslant l \leqslant r - 1)$ round, there will be l selected algorithms in $\hat{\partial}$ and $R - l$ basic algorithms in ∂. A disagreement measure matrix $DIS'_{l\times(R-l)}$ and an average disagreement measure vector $aver'_{1\times(R-l)}$ are initialized to be 0. $DIS'_{l\times(R-l)}$ stores the disagreement measure between arbitrary one basic algorithm in ∂ and arbitrary one selected algorithm in $\hat{\partial}$. $aver'_{1\times(R-l)}$ stores the average disagreement measure of each basic clustering algorithm in ∂. For the j_{th} basic algorithm $\partial_j(\partial_j \in \partial, 1 \leqslant j \leqslant R - l)$, the disagreement measure between ∂_j and the k_{th} selected algorithm $\hat{\partial}_k(\hat{\partial}_k \in \hat{\partial}, 1 \leqslant k \leqslant l)$ is calculated and stored in DIS' (denoted as

Algorithm 4 Diversity Analysis Based Clustering Algorithm Selection Method

```
Input: $I = [IC_1, IC_2, \dots, IC_m]^T$; //IC set
       $\partial = \{\partial_1, \partial_2, \dots, \partial_R\}$; //basic clustering algorithms
       $r$; //predefined number of selected clustering algorithms
       $\theta$; //predefined accuracy threshold
Output: Selected clustering algorithm set $\hat{\partial}$
1:  // First part
2:  for $i = 1$ to $R$ do
3:      perform the $i_{th}$ clustering algorithm on $I$ and obtain its detection accuracy $\theta_i$.
4:      if $\theta_i > \theta$ then
5:          $Y_i = [y_1, y_2, \dots, y_m]^T$ //recording the $i_{th}$ detection result
6:      else
7:          $Y_i = NULL$
8:      end if
9:  end for
10: $Y = [Y_1, Y_2, \dots, Y_R]$ //obtain basic detection results
11: // Second part
12: initialize the disagreement measure matrix $DIS_{R\times R}$ and average disagreement measure vector $aver_{1\times R}$ to be 0
13: for $i = 1$ to $R$ do
14:     for $j = 1$ to $R$ do
15:         $DIS[i][j] = dis_{\partial_i\partial_j}$
16:     end for
17:     $aver_i = Average(\partial_i) = \frac{1}{R}\sum_{j=1}^{R} DIS[i][j]$
18: end for
19: // Third part
20: initialize $\hat{\partial} = \emptyset$
21: select the clustering algorithm which has the maximum value in $aver$
22: remove the selected algorithm from $\partial$ and add it to $\hat{\partial}$
23: for $l = 1$ to $r - 1$ do
24:     initialize the disagreement measure matrix $DIS'_{l\times(R-l)}$ and average disagreement measure vector $aver'_{1\times(R-l)}$ to be 0
25:     for $j = 1$ to $R - l$ do
26:         for $k = 1$ to $l$ do
27:             $DIS'[j][k] = dis_{\partial_j\hat{\partial}_k}$
28:         end for
29:         $aver'_l = \frac{1}{l}\sum_{k=1}^{l} DIS'[j][k]$
30:     end for
31:     remove the basic algorithm which has maximum value in $aver'$ from $\partial$ and add it to $\hat{\partial}$
32: end for
```

$DIS'[j][k]$). Then the average disagreement measure of the j_{th} basic algorithm ∂_j is calculated by $\frac{1}{l}\sum_{k=1}^{l} DIS'[j][k]$ and stored in $aver'$ (denoted as $aver'_j$). Then the basic algorithm which has the maximum value in $aver'$ is removed from ∂ and added to $\hat{\partial}$. This process is repeated until the predefined number of clustering algorithms are selected.

D. CLUSTERING ENSEMBLE VS. MAJORITY VOTING

In this section, the reason why the proposed hybrid clustering ensemble method is used rather than the majority voting method, is analyzed. The experimental comparison of these two methods in terms of detection accuracy and robustness to malicious modifications, will be presented in Section VI.

Firstly, the proposed hybrid clustering ensemble method is useful to provide forensics to reveal untrustworthy testing parties. In the process of integrating basic detection results, the majority voting method only performs statistical counting operations. However, the proposed hybrid clustering ensemble method re-represents each IC as a r-tuple using the label information of all basic detection results, and calculates the correlation and inter-cluster distance between each IC to obtain the final detection result. In the detection process of the proposed hybrid clustering ensemble method, more formal evidence can be provided to reveal the untrustworthy testing party.

Secondly, the detection accuracy and robustness of the proposed hybrid clustering ensemble method are better than the majority voting method. The proposed hybrid clustering ensemble method has performed label normalization, IC re-representation and secondary clustering on basic clustering results, which can help to improve the robustness and detection accuracy of the hardware Trojan method. The majority voting method only performs a simple statistical counting of all basic detection results.

VI. EXPERIMENTAL RESULTS

In this section, the proposed method is evaluated using two kinds of experimental evaluations, EDA tools during IC design flow and the FPGA evaluation. The experimental setup of EDA and FPGA evaluations are described in Section VI-A. The EDA evaluation results of ISCAS89 benchmark circuits and FPGA evaluation results of Trust-HUB benchmark circuits are described in Section VI-B and Section VI-C, respectively. In Section VI-D, the time overhead of the CE-HTDM method is evaluated.

A. EXPERIMENT SETUP

The proposed technique is evaluated on ISCAS89 benchmark circuits first. Two basic Trojans, a 4-bit comparator and a 3-bit counter, are designed for the experiment [18]. Several logic gates are added on these two basic Trojans to generate other Trojans each time. These Trojans are inserted in the benchmark circuits separately. The hardware overhead of these Trojans is varying from 0.005% to 0.2% of the designs. Note that, the actual Trojans with malicious functions usually have much bigger sizes. Therefore if the proposed technique can detect these tiny modifications, it can also detect actual Trojans.

Synopsys Design Compiler is used to synthesize the designs with 65nm technology. Both the inter-die and intra-die process variation is set to be 10%. HSPICE is used to perform Monte Carlo simulation to obtain the transient power supply currents (I_{DDT}) of each IC. The testing set of each benchmark circuit includes 100 unlabeled ICs. For each benchmark circuit, 50 different Trojans are inserted to the design to generate 50 Trojan-infected samples. Then, 50 Trojan-free traces are collected from the genuine circuit, and 50 Trojan-inserted traces are collected from the 50 Trojan-infected samples.

Second, the CE-HTDM is also evaluated with FPGA experiments. The experimental platform is the SAKURA-X board, as shown in Fig. 9. Two Xilinx FPGAs are integrated on the board and serve as the main FPGA (Kintex-7 XC7K160T-1FBGC) and the control FPGA (Spartan-6 XC6LX45-2FGG484C), respectively. The signals are captured by a Keysight InfiniiVision DSOX3102A oscilloscope for further analysis. Two SAKURA-X boards are used to model the effect of process variations.

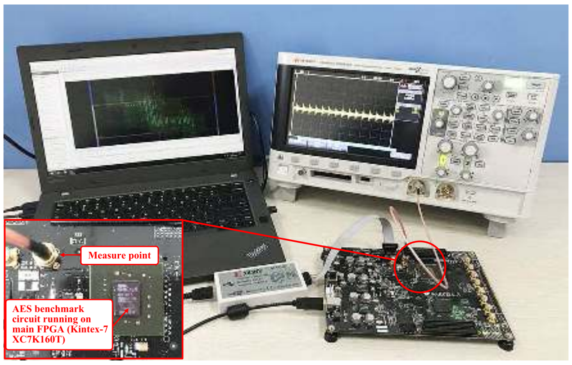
Measure point
AES benchmark circuit running on main FPGA (Kintex-7 XC7K160T)

FIGURE 9. FPGA Experimental platform.

In the FPGA evaluation, the benchmark circuit is the AES [35] circuit. The Trojans are obtained from Trust-HUB [26]. According to the study in [36], some of the Trojans will be optimized during the synthesis process. Therefore, we exclude these Trojan benchmarks from the experiment. The remaining benchmarks are AES-T100, AES-T200, AES-T400, AES-T700, AES-T800, AES-T900, AES-T1200, and AES-T1700. These 8 Trojans will not be optimized during the synthesis process. For each Trojan-inserted benchmark circuit, 10 power traces are collected from actual experiments without triggering the Trojans. For the genuine benchmark circuit, 80 power traces are collected for testing. Hence, the testing set I consists of 80 Trojan-free traces and 80 Trojan-inserted traces.

There are 3 testing parties and the testing party 1 (TP_1) is assumed to be untrustworthy. In general, untrustworthy testing parties will only modify a small percentage of the detection results to hide Trojans. This is because that if untrustworthy testing parties modify a large part of the detection results, it will be easy to expose themselves and their reputations will be destroyed. Therefore, the modified percentages of the detection results by the untrustworthy testing party are set to be 5%, 7% and 9% for evaluation.

B. EDA EVALUATION RESULTS ON ISCAS89 BENCHMARKS

First, the robustness of CE-HTDM against untrustworthy testing parties is analyzed, as shown in Table 2. The detection accuracy represents the rate of ICs correctly clustered among all the ICs under test. It is shown that the detection accuracy of CE-HTDM is better than each of the three testing parties. This means, the proposed method is robust against malicious modifications introduced by the untrustworthy testing party (testing party 1). As the modified percentage increases, the detection accuracy of CE-HTDM decreases slightly.

TABLE 2. Robustness: the detection results of CE-HTDM under untrustworthy testing parties.

Modified percentage	Benchmark	Detection accuracy			
		TP_1 (EM)	TP_2 (HAC)	TP_3 (sIB)	CE-HTDM
5%	s38417	86%	90%	86%	92%
	s35932	84%	92%	87%	93%
	s15850	82%	90%	87%	90%
	s5378	80%	85%	84%	89%
7%	s38417	85%	90%	87%	91%
	s35932	81%	91%	85%	93%
	s15850	79%	90%	87%	90%
	s5378	83%	87%	84%	91%
9%	s38417	82%	90%	86%	90%
	s35932	83%	91%	87%	91%
	s15850	80%	88%	86%	89%
	s5378	79%	85%	84%	86%

We now present the detection accuracy and clustering quality of CE-HTDM under trustworthy testing parties, which means there are no malicious modifications introduced by these three testing parties. The detection results are given in Table 3. For different benchmark circuits, the detection accuracy of CE-HTDM is higher than that of each testing party. The clustering quality of the detection results of s38417 is shown in Fig. 10. It is shown that the inter-cluster distance obtained by CE-HTDM is significantly larger than the inter-cluster distance obtained by each testing party. This is helpful for forming a clear cluster boundary. The clustering quality of other benchmark circuits is similar to that of s38417. In conclusion, CE-HTDM can improve both the detection accuracy and the clustering quality effectively.

TABLE 3. The detection results of CE-HTDM under trustworthy testing parties.

Benchmark	Detection accuracy			
	TP_1 (EM)	TP_2 (HAC)	TP_3 (sIB)	CE-HTDM
s38417	89%	88%	84%	93%
s35932	85%	91%	85%	93%
s15850	84%	88%	84%	91%
s5378	83%	83%	81%	92%

The comparison of CE-HTDM and the majority voting method is presented in Fig. 11, which includes two cases, under trustworthy testing parties (Fig. 11(a)) and under untrustworthy testing parties (Fig. 11(b), Fig. 11(c), Fig. 11(d)). As shown in Fig. 11(a), for different benchmarks, the detection accuracy of the CE-HTDM is significantly higher than that of the majority voting method. As shown

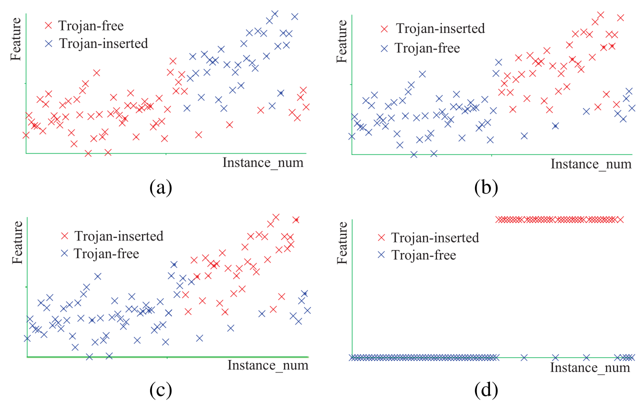

FIGURE 10. The clustering results of s38417 under trustworthy testing parties [25]. (a) TP_1 (EM). (b) TP_2 (HAC). (c) TP_3 (sIB). (d) The proposed CE-HTDM.

in Fig. 11(b), Fig. 11(c) and Fig. 11(d), with the increasement of the modified percentage, the detection accuracy of both methods slightly decreases, and the detection accuracy of the majority voting method has decreased more sharply than CE-HTDM. Obviously, the detection accuracy of the CE-HTDM is still higher than the majority voting method under these malicious modifications. Overall, the CE-HTDM method is more effective on detecting Trojans and more robust to malicious modifications than the majority voting method.

C. FPGA EVALUATION RESULTS ON TRUST-HUB BENCHMARKS

In the FPGA evaluation on Trust-Hub Trojan benchmarks [26], the number of LUTs and registers utilized by the Trojans are counted, and the area overhead of the Trojans are calculated, as shown in Table 4. The hardware overhead of these Trojans is varying from 1.93% to 9.45% of the design.

The FPGA evaluation results of the basic clustering algorithms, which includes two types proposed in [24], the partition-based hardware Trojan detection method and the density-based hardware Trojan detection method, are given in Table 5 and Table 6, respectively. It is shown that the detection accuracy of sIB algorithm in the FPGA evaluation is much lower than its detection accuracy in the EDA evaluation. The reason is that the sIB algorithm is more susceptible to noises and process variations. The detection accuracy of the remaining partition-based or density-based clustering algorithms is in a range of 87% $\sim$ 92.5%, which is lower than the detection accuracy of the supervised hardware Trojan detection technique proposed in [18]. However, the method proposed in [18] works in the supervised hardware Trojan detection situation in which the golden simulated models are needed for training, while the clustering based Trojan detection method works in the unsupervised situation does not require golden chips or golden models. Therefore the detection results of these clustering based hardware Trojan detection techniques are acceptable.

According to the discussion in Section V-C, the EM algorithm, k-means algorithm and DBSCAN algorithm are chosen as the basic clustering algorithms to build the proposed detection method. Other clustering algorithms are also feasible. The FPGA evaluation results of CE-HTDM are given in Table 7. It is shown that the detection accuracy has been increased to 93.75% under the trustworthy testing parties, which is higher than the detection accuracy of each testing party. As the modified percentage increases, the detection accuracy of CE-HTDM has slightly reduced. When the modified percentage is set to be 9%, the detection accuracy of CE-HTDM is still high up to 91.50%. This implies that the CE-HTDM method has a strong robustness to malicious modifications.

The comparison of CE-HTDM and the majority voting method on AES benchmark circuit (in the FPGA evaluation) is shown in Fig. 12. When all the testing parties are

TABLE 4. Hardware overhead of the genuine AES circuit and Trojan-inserted AES circuits.

Benchmarks	LUTs	Registers	Trojan_LUTs	Trojan_Registers
AES	1822	2692	0	0
AES-T100	1824	2776	5 (0.27%)	84 (3.12%)
AES-T200	1824	2776	5 (0.27%)	84 (3.12%)
AES-T400	1978	2848	158 (8.67%)	156 (5.79%)
AES-T700	1827	2776	6 (0.33%)	84 (3.12%)
AES-T800	1828	2777	9 (0.49%)	85 (3.16%)
AES-T900	2117	2776	299 (16.41%)	84 (3.12%)
AES-T1200	2111	2776	290 (15.92%)	84 (3.12%)
AES-T1700	2138	2846	317 (17.40%)	154 (5.72%)

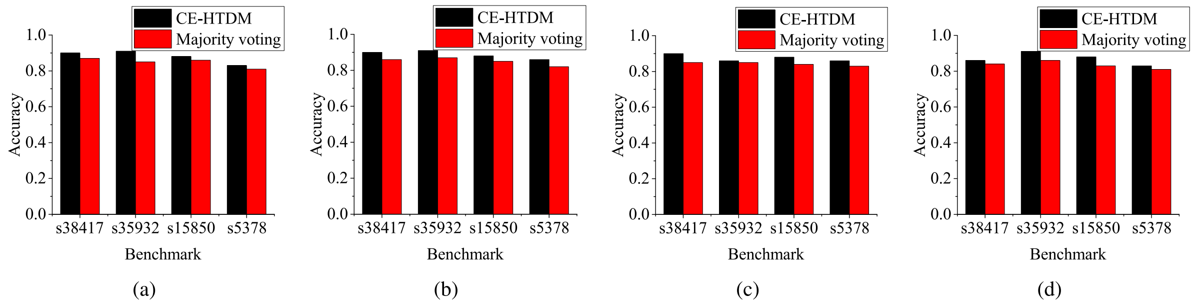

FIGURE 11. Comparison of the detection accuracy of CE-HTDM and the majority voting method under different modified percentages in the EDA evaluation [25]. (a) 0%. (b) 5%. (c) 7%. (d) 9%.

TABLE 5. FPGA evaluation results of the partition-based hardware Trojan detection method [24].

Scenarios	Partition-based					
Algorithms	EM	FarthestFirst	HAC	k-means	sIB	XMeans
Accuracy	92.50%	87.50%	90.63%	91.25%	68.13%	90.63%

TABLE 6. FPGA evaluation results of the density-based hardware Trojan detection method [24].

Scenarios	Algorithms	Parameter settings and detection accuracy					
Density-based	DBSCAN	$MinPts$	70	70	70	70	70
		ε	3.9	3.92	3.94	3.96	3.98
		Accuracy	90.63%	91.75%	90.50%	88.88%	87.63%

TABLE 7. FPGA evaluation results of the proposed clustering ensemble based hardware Trojan detection method (CE-HTDM).

Reliability	Modified percentage	TP_1 (EM)	TP_2 (k-means)	TP_3 (DBSCAN)	CE-HTDM
Trustworthy	0%	92.50%	91.25%	91.75%	93.75%
Untrustworthy	5%	87.50%	91.25%	91.75%	93.75%
Untrustworthy	7%	85.50%	91.25%	91.75%	92.13%
Untrustworthy	9%	83.50%	91.25%	91.75%	91.50%

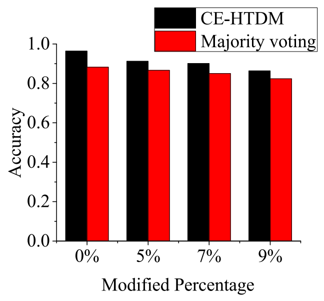

FIGURE 12. The comparison of the CE-HTDM method and the majority voting method in the FPGA evaluation.

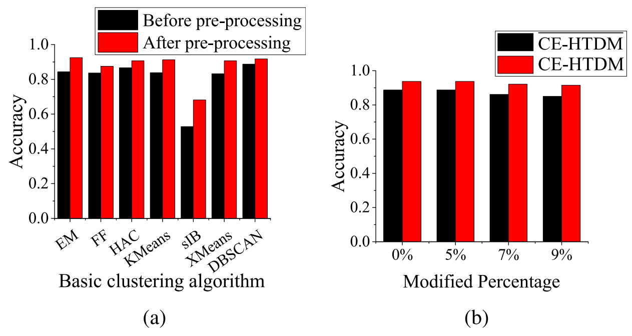

FIGURE 13. Detection accuracy with and without preprocessing. (a) Basic clustering algorithms. (b) CE-HTDM.

trustworthy, the detection accuracy of CE-HTDM is significantly higher than that of the majority voting method. When the modified percentage increases, the detection accuracy of CE-HTDM and majority voting method slightly decrease, and the detection accuracy of CE-HTDM decreases more slowly than the majority voting method. It is shown that the detection accuracy of CE-HTDM is still higher than that of the majority voting method under malicious modifications. This implies that the CE-HTDM method has better detection accuracy and robustness than the majority voting method under untrustworthy testing parties.

As shown in Fig. 13, the comparison of the detection accuracy with and without the preprocessing process is also presented. In Fig. 13(a), we present a comparison of the detection accuracy of the basic clustering algorithms, before and after the preprocessing process. Before preprocessing, the detection accuracy of basic clustering algorithms is in a range of 83.12% to 88.75% (except the sIB algorithm). After preprocessing, the detection accuracy has increased to a range of 87.5% to 92.50%. It is obvious that the detection accuracy of all basic clustering algorithms is significantly improved after preprocessing. As mentioned above, the performance of the sIB algorithm is poor in the FPGA evaluation due to it is susceptible to noises and process variations. In Fig. 13(b), the comparison of the detection accuracy of the CE-HTDM method with and without the preprocessing process is presented, where the CE-HTDM without preprocessing is represented as $\overline{\text{CE-HTDM}}$, while the CE-HTDM which has the preprocessing phase is represented as CE-HTDM. The detection accuracy of $\overline{\text{CE-HTDM}}$ is in a range of 85.50% to 88.75%, while the detection accuracy of CE-HTDM is in a range of 91.50% to 93.75%. Obviously, the detection accuracy of CE-HTDM has increased around 5% after preprocessing. In conclusion, the preprocessing method (UCFS) not only improves the detection accuracy of basic clustering methods, but also effectively improves the detection accuracy of the proposed CE-HTDM.

D. TIME OVERHEAD OF CE-HTDM

In this section, the time overhead of the CE-HTDM method is evaluated in EDA experiments and FPGA experiments.

TABLE 8. The time overhead of the CE-HTDM method in the EDA evaluation.

Modified percentage	Benchmark	t_c	t_p	t_{cg}			$t_{R\&T}$	t_{cc}	$t_{CE-HTDM}$
				tp_1 (EM)	tp_2 (HAC)	tp_3 (sIB)			
0% (Trustworthy)	s38417	0.0005	0.01	0.36	0.05	0.27	0.01	0.03	0.41
	s35932	0.0005	0.01	0.33	0.06	0.22	0.01	0.05	0.4
	s15850	0.0005	0.01	0.38	0.08	0.24	0.01	0.03	0.43
	s5378	0.0005	0.01	0.35	0.07	0.3	0.01	0.02	0.39
5% (Untrustworthy)	s38417	0.0005	0.01	0.36	0.05	0.27	0.01	0.03	0.41
	s35932	0.0005	0.01	0.33	0.06	0.22	0.01	0.03	0.38
	s15850	0.0005	0.01	0.38	0.08	0.24	0.01	0.02	0.42
	s5378	0.0005	0.01	0.35	0.07	0.3	0.01	0.02	0.39
7% (Untrustworthy)	s38417	0.0005	0.01	0.36	0.05	0.27	0.01	0.04	0.42
	s35932	0.0005	0.01	0.33	0.06	0.22	0.01	0.02	0.37
	s15850	0.0005	0.01	0.38	0.08	0.24	0.01	0.04	0.44
	s5378	0.0005	0.01	0.35	0.07	0.3	0.01	0.03	0.4
9% (Untrustworthy)	s38417	0.0005	0.01	0.36	0.05	0.27	0.01	0.02	0.4
	s35932	0.0005	0.01	0.33	0.06	0.22	0.01	0.04	0.39
	s15850	0.0005	0.01	0.38	0.08	0.24	0.01	0.02	0.42
	s5378	0.0005	0.01	0.35	0.07	0.3	0.01	0.03	0.4

TABLE 9. The time overhead of the CE-HTDM method in the FPGA evaluation.

Modified percentage	t_c	t_p	t_{cg}			$t_{R\&T}$	t_{cc}	$t_{CE-HTDM}$
			tp_1 (EM)	tp_2 (KMeans)	tp_3 (DBSCAN)			
0% (Trustworthy)	0.0008	0.01	0.28	0.03	0.11	0.01	0.05	0.35
5% (Untrustworthy)	0.0008	0.01	0.28	0.03	0.11	0.01	0.03	0.33
7% (Untrustworthy)	0.0008	0.01	0.28	0.03	0.11	0.01	0.02	0.32
9% (Untrustworthy)	0.0008	0.01	0.28	0.03	0.11	0.01	0.02	0.32

The experimental tools in the EDA evaluation are WEKA 3.6 and PyCharm 2017.2.3, which run on the Intel(R) Core(TM) i5-4590 CPU. The frequency of the CPU is 3.30GHz.

In Section V-B, we have analyzed that the time overhead of CE-HTDM includes the time of five steps: collecting test data, preprocessing, clustering generation, relabeling & transforming, and clustering combination, which can be denoted as $t_{CE-HTDM} = t_c + t_p + t_{cg} + t_{R\&T} + t_{cc}$. In EDA evaluation, 100 power traces are collected for testing. Therefore, the time overhead of collecting test data is $t_c = M \times 5 \times 10^{-6} = 5 \times 10^{-4}$s. As t_c is too small, it can be ignored.

According to the log files of WEKA and PyCharm, the time overhead of t_p, t_{cg}, $t_{R\&T}$ and t_{cc}, are calculated, as shown in Table 8. Note that, the time spending on modifying the detection results by untrustworthy testing house, is not a part of the time overhead of the CE-HTDM. Overall, the time overhead of CE-HTDM is in a range of 0.37s to 0.44s.

There are three testing parties in the CE-HTDM and each testing party uses a traditional hardware Trojan detection method. The time overhead for one test house to perform one testing process can be represented by $t_{Tra} = t_c + t_p + t_{cg} \approx t_p + t_{cg}$. Therefore, for the i_{th} testing party in CE-HTDM, its time overhead is $t_{Tra} = t_p + tp_i$, where tp_i is the time overhead of the i_{th} testing party performing a clustering generation step. The time overhead for each of these testing parties is in a range of 0.06s to 0.39s. Therefore, compared to the maximum time overhead of the traditional method (one testing party performs one detection process), the increased time overhead of CE-HTDM is small. Besides, it can be observed that the time overhead of CE-HTDM is mainly correlated with two parts: t_{cg} and t_{cc}. The time overhead of clustering generation $t_{cg} = \max\{tp_1, tp_2, tp_3\}$ (as discussed in Section V-B) is in a range of 0.33s to 0.38s, and the time overhead of clustering combination t_{cc} is in a range of 0.02s to 0.05s. The time overhead of the remaining three parts is relatively small, which has a low contribution to the total time of CE-HTDM.

The time overhead of the CE-HTDM method in the FPGA evaluation is presented in Table 9. 160 power traces are collected for testing. Therefore, the time overhead of collecting test data is $t_c = M \times 5 \times 10^{-6} = 8 \times 10^{-4}$s. The value of t_c in FPGA evaluation is also small, which can be ignored. It is shown that the time overhead of CE-HTDM is in a range of 0.32s to 0.35s, while the time overhead of the traditional method is in a range of 0.04s to 0.29s. Compared to the maximum time overhead of these traditional methods, the time overhead increased by CE-HTDM is very low.

When the modified percentage changes, the time overhead of clustering generation t_{cg} does not change, while the time overhead of clustering combination t_{cc} is in a range of 0.02s to 0.05s. The reason is that, when an untrusted testing party modifies the detection results, it will not affect the previous detection process (clustering generation). However, due to the changes in the data composition of the clustering generation results, the processing time of clustering combination will change. In general, since the untrusted testing party modifies the labels of Trojan-inserted ICs, the data composition of the clustering generation results will become simpler, which will lead to a shorter time for clustering combination phase.

VII. CONCLUSION

In this paper, we propose a golden models-free hardware Trojan detection method against untrustworthy testing parties by exploiting a novel hybrid clustering ensemble method. To the best of our knowledge, this work is the first to consider that the testing party may be untrustworthy. We formulate two attack models of untrustworthy testing parties, and propose an adversarial data generation algorithm for the untrustworthy testing party to modify the collected test data. Then, we propose a novel hybrid clustering ensemble method for Trojan detection, which can resist the malicious modifications on Trojan detection results introduced by untrustworthy testing parties. Besides, the UCFS method is modified and used to preprocess the raw power traces of ICs, which can filter out noisy features and extract predominant features to alleviate the impact of process variations and noises. Theoretical analysis on four keys problems are also presented: the number of necessary testing parties; the time overhead and computational overhead of the proposed method; choose the basic clustering algorithms by using a proposed diversity analysis based algorithm; the reason why the clustering ensemble method is superior to the majority voting method. Both EDA evaluation on ISCAS89 benchmarks and FPGA evaluation on Trust-HUB benchmarks are performed. Experimental results show that the proposed method can obtain better detection accuracy and clustering quality (larger inter-cluster distance) than each testing party. Meanwhile, it can defeat untrustworthy testing parties without increasing much overhead.

REFERENCES

[1] M. Rostami, F. Koushanfar, and R. Karri, "A primer on hardware security: Models, methods, and metrics," *Proc. IEEE*, vol. 102, no. 8, pp. 1283–1295, Aug. 2014.
[2] M. Tehranipoor and F. Koushanfar, "A survey of hardware trojan taxonomy and detection," *IEEE Des. Test Comput.*, vol. 27, no. 11, pp. 10–25, Feb. 2010.
[3] R. S. Chakraborty, S. Narasimhan, and S. Bhunia, "Hardware Trojan: Threats and emerging solutions," in *Proc. IEEE Int. High Level Des. Valid. Test Workshop*, Nov. 2009, pp. 166–171.
[4] M. Yasin, O. Sinanoglu, and J. Rajendran, "Testing the trustworthiness of IC testing: An oracle-less attack on IC camouflaging," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 11, pp. 2668–2682, Nov. 2017.
[5] D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi, and B. Sunar, "Trojan detection using IC fingerprinting," in *Proc. IEEE Symp. Secur. Privacy*, May 2007, pp. 296–310.
[6] Y. Jin and Y. Makris, "Hardware Trojan detection using path delay fingerprint," in *Proc. IEEE Int. Workshop Hardw. Oriented Secur. Trust*, Jun. 2008, pp. 51–57.
[7] A. N. Nowroz, K. Hu, F. Koushanfar, and S. Reda, "Novel techniques for high-sensitivity hardware trojan detection using thermal and power maps," *IEEE Trans. Comput.-Aided Design Integr.*, vol. 33, no. 12, pp. 1792–1805, Dec. 2014.
[8] M. Xue, W. Liu, A. Hu, and Y. Wang, "Detecting hardware Trojan through time domain constrained estimator based unified subspace technique," *IEICE Trans. Inf. Syst.*, vol. E97-D, no. 3, pp. 606–609, Mar. 2014.
[9] M. Xue, A. Hu, and G. Li, "Detecting hardware Trojan through heuristic partition and activity driven test pattern generation," in *Proc. Commun. Secur. Conf.*, May 2014, pp. 1–6.
[10] R. S. Chakraborty, F. Wolff, S. Paul, C. Papachristou, and S. Bhunia, "MERO: A statistical approach for hardware Trojan detection," in *Proc. Int. Workshop Cryptogr. Hardw. Embedded Syst.*, Sep. 2009, pp. 396–410.
[11] T. F. Wu, K. Ganesan, Y. A. Hu, H. S. P. Wong, S. Wong, and S. Mitra, "TPAD: Hardware Trojan prevention and detection for trusted integrated circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 35, no. 4, pp. 521–534, Aug. 2016.
[12] K. Xiao, D. Forte, and M. Tehranipoor, "A novel built-in self-authentication technique to prevent inserting hardware Trojans," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 33, no. 12, pp. 1778–1791, Dec. 2014.
[13] S. Mal-Sarkar, A. Krishna, A. Ghosh, and S. Bhunia, "Hardware trojan attacks in FPGA devices: Threat analysis and effective counter measures," in *Proc. 24th Great Lakes Symp. VLSI*, May 2014, pp. 287–292.
[14] M. M. Farag and M. A. Ewais, "Smart employment of circuit redundancy to effectively counter trojans (SECRET) in third-party IP cores," in *Proc. Int. Conf. ReConFig. Comput. FPGAs*, Dec. 2014, pp. 1–6.
[15] N. B. Gunti, A. Khatri, and K. Lingasubramanian, "Realizing a security aware triple modular redundancy scheme for robust integrated circuits," in *Proc. Int. Conf. Very Large Scale Integr.*, Oct. 2014, pp. 1–6.
[16] C. Bao, D. Forte, and A. Srivastava, "On reverse engineering-based hardware Trojan detection," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 35, no. 1, pp. 49–57, Jan. 2016.
[17] A. Kulkarni, Y. Pino, and T. Mohsenin, "Adaptive real-time Trojan detection framework through machine learning," in *Proc. IEEE Int. Symp. Hardw. Oriented Secur. Trust*, May 2016, pp. 120–123.
[18] M. Xue, J. Wang, and A. Hux, "An enhanced classification-based golden chips-free hardware Trojan detection technique," in *Proc. IEEE Asian Hardw.-Oriented Secur. Trust*, Dec. 2016, pp. 1–6.
[19] A. A. Nasr and Z. Mo Abdulmageed, "Automatic feature selection of hardware layout: A step toward robust hardware Trojan detection," *J. Electron. Test.*, vol. 32, no. 3, pp. 357–367, Jun. 2016.
[20] A. Kulkarni, Y. Pino, and T. Mohsenin, "SVM-based real-time hardware Trojan detection for many-core platform," in *Proc. 17th Int. Symp. Quality Electron. Design*, Mar. 2016, pp. 362–367.
[21] B. Çakır and S. Malik, "Hardware Trojan detection for gate-level ICs using signal correlation based clustering," in *Proc. Design, Autom. Test Eur. Conf. Exhib.*, Mar. 2015, pp. 471–476.
[22] H. Salmani, "COTD: Reference-free hardware trojan detection and recovery based on controllability and observability in gate-level netlist," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 2, pp. 338–350, Feb. 2017.
[23] P.-S. Ba, S. Dupuis, M.-L. Flottes, G. D. Natale, and B. Rouzeyre, "Using outliers to detect stealthy hardware Trojan triggering?" in *Proc. IEEE Int. Verification Secur. Workshop*, Jul. 2016, pp. 1–6.
[24] R. Bian, M. Xue, and J. Wang, "A novel golden models-free hardware Trojan detection technique using unsupervised clustering analysis," in *Proc. 4th Int. Conf. Cloud Comput. Secur.*, Jun. 2018, pp. 634–646.
[25] R. Bian, M. Xue, and J. Wang, "Building trusted golden models-free hardware Trojan detection framework against untrustworthy testing parties using a novel clustering ensemble technique," in *Proc. 17th IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun.*, Aug. 2018, pp. 1458–1463.
[26] *Trojan Benchmarks*. [Online]. Available: http://www.trust-hub.org/benchmarks/trojan
[27] M. L. Zepeda-Mendoza and O. Resendis-Antonio, "Hierarchical agglomerative clustering," in *Encyclopedia of Systems Biology*. New York, NY, USA: Springer, 2013, pp. 886–887.
[28] A. Strehl and J. Ghosh, "Cluster ensembles—A knowledge reuse framework for combining multiple partitions," *J. Mach. Learn. Res.*, vol. 3, no. 12, pp. 583–617, Mar. 2003.
[29] A. Fred and A. Jain, "Combining multiple clusterings using evidence accumulation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 6, pp. 835–850, Jun. 2005.
[30] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," *ACM Comput. Surv.*, vol. 31, no. 3, pp. 264–323, Sep. 1999.
[31] A. K. Jain and J. V. Moreau, "Bootstrap technique in cluster analysis," *Pattern Recognit.*, vol. 20, no. 5, pp. 547–568, 1987.
[32] M. Xue, R. Bian, J. Wang, and W. Liu, "A co-training based hardware Trojan detection technique by exploiting unlabeled ICs and inaccurate simulation models," in *Proc. 17th IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun.*, Aug. 2018, pp. 1452–1457.

[33] S. M. Moosavi-Dezfooli, A. Fawzi, O. O. Fawzi, and P. Frossard, ''Universal adversarial perturbations,'' in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 86–94.
[34] L. Yu and H. Liu, ''Feature selection for high-dimensional data: A fast correlation-based filter solution,'' in *Proc. Int. Conf. Mach. Learn.*, Aug. 2003, pp. 856–863.
[35] J. Daemen and V. Rijmen, *The Design of Rijndael: AES-the Advanced Encryption Standard*. Heidelberg, Germany: Springer, 2013.
[36] T. Reece and W. H. Robinson, ''Analysis of data-leak hardware Trojans in AES cryptographic circuits,'' in *Proc. IEEE Int. Conf. Technol. Home Secur.*, Nov. 2013, pp. 467–472.

MINGFU XUE (S'11–M'14) received the Ph.D. degree in information and communication engineering from Southeast University, China, in 2014. From 2011 to 2012, he was a Visiting Ph.D. Student with Nanyang Technological University, Singapore. He is currently an Assistant Professor with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, China. His research interests include hardware security, hardware Trojan detection, the Internet of Things security, and artificial intelligence security. He is a technical program committee member of several international conferences. He is an Executive Committee Member of the ACM Nanjing Chapter. He is the Program Chair of the third Chinese Symposium on Hardware Security. From 2014 to 2018, he has presided 10 research projects or fundings.

RONGZHEN BIAN received the B.S. degree in information security from the Nanjing University of Aeronautics and Astronautics, China, in 2016, where he is currently pursuing the master's degree in computer technology with the College of Computer Science and Technology. His research interests include hardware Trojan detection, hardware security, and information security.

WEIQIANG LIU (M'12–SM'15) received the B.Sc. degree in information engineering from the Nanjing University of Aeronautics and Astronautics (NUAA), Nanjing, China, in 2006, and the Ph.D. degree in electronic engineering from Queen's University Belfast, Belfast, U.K., in 2012. In 2013, he joined the College of Electronic and Information Engineering, NUAA, where he is currently an Associate Professor. He has published one research book by Artech House and over 70 leading journal and conference papers. His research interests include emerging technologies in computing systems and hardware security. He is a member of CASCOM and VSA Technical Committee, IEEE CAS Society. One of his papers received the Most Popular Article of the IEEE TRANSACTIONS ON COMPUTERS (IEEE TC), in 2017, and one was selected as the Feature Paper of the IEEE TC in the 2017 December issue. His paper also received the Best Paper Candidate of the ISCAS 2011 and the GLSVLSI 2015. He serves as an Associate Editor for the IEEE TC, as the Leader for the Multimedia Team at the TC Editorial Board, as a Steering Committee Member for the IEEE TRANSACTIONS ON MULTI-SCALE COMPUTING SYSTEMS, and as a Guest Editor for the IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTING and the *Elsevier Microelectronics Journal*. He is a program committee member for several conferences, including ARITH, ISCAS, ASAP, ISVLSI, NANOARCH, and ICONIP.

JIAN WANG received the Ph.D. degree in computer application technology from Nanjing University, China, in 1998. From 2001 to 2003, he was a Postdoctoral Researcher with The University of Tokyo, Japan. From 1998 to 2004, he was an Associate Professor with Nanjing University. From 2010 to 2015, he was the Vice Director of the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, China, where he is currently a Full Professor. His research interests include applied cryptography, system security, key management, security protocol, and information security. He is a Committee Member of the Chinese Cryptography Society, as well as the Director of the Jiangsu Provincial Cryptography Society. He was the Chair of the Nanjing Section of the China Computer Federation YOCSEF, from 2012 to 2013.