IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 7, 2019, accepted July 28, 2019, date of publication August 1, 2019, date of current version August 22, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2932478

A Locating Method for Multi-Purposes HTs Based on the Boundary Network

CHEN DONG[1,2,3], (Member, IEEE), FAN ZHANG[1,2], XIMENG LIU[1,2], (Member, IEEE), XING HUANG[1,4], (Member, IEEE), WENZHONG GUO[1,3], (Member, IEEE), AND YANG YANG[1,2], (Member, IEEE)

[1]Key Laboratory of Spatial Data Mining and Information Sharing, College of Mathematics and Computer Science, Ministry of Education, Fuzhou University, Fuzhou 350116, China
[2]Key Laboratory of Information Security of Network Systems, Fuzhou University, Fujian 350116, China
[3]Fujian Provincial Key Laboratory of Network Computing and Intelligent Information Processing, Fuzhou University, Fujian 350116, China
[4]Department of Computer Science, National Tsing Hua University, Hsinchu 30013, Taiwan

Corresponding author: Xing Huang (xing.huang1010@gmail.com)

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, and in part by the Foundation of the Education Department of Fujian Province, China, under Grant JAT170099.

ABSTRACT Recently, there are various methods for detecting the hardware trojans (HTs) in the integrated circuits (ICs). The circuit's logic representations of different types, structures, and functional characteristics should be different. Each type of circuit has its' own performance characteristics according to its' purpose. However, the traditional HTs detection methods adopt the same approach to deal with the multi-purpose hardware trojan. At the same time, as the scale of integrated circuits growing, the structures are more complex, and the functions are more refined. The current situation makes the traditional HTs detection methods weaker and even unfeasible. Therefore, we propose an HTs classified locating method based on machine learning, named ML-HTCL, which belongs to the static detection and locating method. In ML-HTCL, different purpose HTs were represented by different features. Due to the different features, the ML-HTCL employs the multi-layer BP neural network for the control signal type HTs and the one-class SVM for the information leakage HTs, respectively. To deal with the HTs completely, the boundary nets are considered for all data set, while those were ignored by the most existing methods due to the high detection error rate. After detection, the HTs' precise locations were achieved. To evaluate the ML-HTCL, 17 gate-level netlist benchmarks are used for training and testing by leave-one-out cross-validation. From the results, the ML-HTCL reaches 85.05% of TPR and 73.91% of TNR for all kinds of HTs, which performs better than the most existing methods.

INDEX TERMS Hardware trojan, Integrated circuit, boundary network, static detection, ML-HTCL.

I. INTRODUCTION

With the popularity of electronic devices, the chips are widely used in computers, mobile phones, home appliances, automobiles, high-speed rail, power grids, medical instruments, robots, industrial control, and other electronic products and systems. The integrated circuit (IC) is usually called chip, which is the core of high-end manufacturing. Currently, the global chip industry is booming, and the global chip sales are generally on the rise.

However, due to the untrusted third-party suppliers all over the world, it has been reported that there is a risk of malicious circuits that may be inserted into IC products [1], these malicious circuits are named Hardware Trojan (HT) [2]. Hackers are trying to insert the HT into the chip to achieve their purpose, such as stealing information, destroying equipment, denial of service, etc. The various designed tiny malicious circuits are inserted into the original ICs. Especially in the design stage, the malicious vendors can insert Hardware Trojans into IC products easily because they only need to modify the design file of IC, such as source codes written in hardware description language [3]. In the final analysis, the deliberately designed unknown circuits are the main source of the IC's

The associate editor coordinating the review of this manuscript and approving it for publication was Farid Boussaid.

security threat. The IBM Research Center in 2007 proposes the conception of the Hardware Trojan [4]: the Hardware Trojans refer to the malicious circuits or harmful alterations to the original circuit that exists from life cycle of the chip design phase to the packaging test phase.

With the improvement of chip technology, the scale of chips has now developed into a very large-scale integrated circuit(VLSI). Hundreds of millions of components make the traditional Hardware Trojan detecting technology has big trouble to deal with the HTs, even unfeasible, such as the method based on functional testing. Every year, numerous governments and companies face the threat of HTs, such as information disclosure, Denial of Service (DOS) [5], changes of original circuit function (When the counter reaches a certain number, the node ER (a node named ER) is modified into an incorrect value at a new node ER* [6]) and even the destruction of the chips. The most effective way to get a standard chip today is to use destructive testing, but this method would cost much money depending on the complexity of the circuit and take weeks or even months to process. Additionally, at the end of this invasive process, the IC cannot be used anymore, and we just get the information for a single IC sample [7].

From the processing stage, the current Hardware Trojan detection methods could be divided into pre-silicon detection and post-silicon detection. The post-silicon detection techniques [8]–[11] generally judge a chip according to the activation of HT parts. The most widely used dynamic detecting method is the side-channel signal analysis [12]–[19], but the cost of large-scale processing circuits is too high and even unimplemented. The pre-silicon detection method emerged in recent years, has attracted more attention. It does not require the circuit to be in operation and is studying the internal structure of the circuit. It generally uses a classification or matching algorithm. The novel method of static detection is combined with machine learning algorithms. In ICs manufacturing, especially in the design stage, the HTs are easily inserted by attackers. The gate-level netlist is used to describe the logical structure of the circuits in the design stage, which can be regarded as a structure description file. All circuits can be checked regardless of the detail functions. If the HTs can be found in the netlist, the most of HTs can be effectively defended.

As far as the existing literature is concerned, the researches of static detection only stay at the stage of detecting HT. In other words, the researches are without taking into account the impact of the boundary network of the HT circuit and the normal circuit on the test results, without further consideration of the positioning of HT. They are trying to find a universal detection method for HT. Although there has emerged some Hardware Trojan locating methods, these methods belong to roughly locating, which only realize the locating in the basic image module of a chip, and it is impossible to locate the specific components and the wire mesh of the circuit accurately.

In the paper, we proposed a method called ML-HTCL based on machine learning. Different measures are used for different HT circuits, and two new goals about the detection have been put forward: 1) Detect all the HTs in the testing netlist. Distinguish the normal nets with the HTs nets to protect the original structure of the netlist. 2) Locate the Hardware Trojans in these netlists based on the detecting results. Specifically, the main contributions of this paper are as follows:

- **Testing set split**. The benchmark circuits are split into the control signal type HTs and the information leakage type HTs, because of some differences of different purpose HT circuits. Based on these differences, some tests for different series of netlists have been carried out, and the results show that such an attempt is correct.
- **New features extract**. In allusion to the structure of the S series, which will be explained in the section III. We expanded the original range of feature values and created some new features. We have obtained a good result by filtering the new features through the random forest algorithm.
- **Boundary network redefine**. The concept of a boundary network is redefined. The detection and location of Hardware Trojan are implemented based on considering the boundary network. The consideration of the boundary network makes the boundary between the HT circuit and the normal circuit much more clear and improves the detection efficiency, as the meanwhile ensuring the completeness of locating the HTs.
- **HTs Precisely locate**. Basic locating of all the detected HTs is realized, including the most difficult boundary network. Comparing to some previous locating results, our results can accurately locate every wire in the circuit and related components instead of rough image positioning. It is the key technique for further dealing with Hardware Trojans.

A. ORGANIZATION OF THIS PAPER

The rest of this paper is organized as follows: Section II gives a brief introduction of the structure of HTs; here are some new ideas about detecting the RS series and the S series. Section III describes the Hardware Trojan locating flow with related contents. Section IV shows the experiment results on different types of HTs. It is the comparison of our methods with other papers in Section V. Finally, Section VI is the conclusion.

II. RELATED WORK

In the existing literature, Nowroz *et al.* [20] first proposed a method by the usage of machine learning to deal with Hardware Trojan detection problems in 2014. While there are quite a few papers about locating Hardware Trojan. Moreover, most of the locating methods are based on the traditional side-channel measurement.

Immersion of machine learning algorithms makes HTs detection problems more and more attention and research. For example, Salmani *et al.* [21] proposed an HT detection technology based on the controllability and observability characteristics of gate-level netlists. This method takes the controllability and observability of the IC netlist, which is extracted from software named SCOPE, as the main characteristics of the HTs to judge whether the netlist including HTs or not. However, this method only tells testers whether the chip netlist is inserted HTs, which result is true or false, which means we almost impossible to process further work with HTs. Hasegawa *et al.* [22] adopted the random forest to identify the HTs in 2017. They put forward 51 features to stand for an HT net, and they tested 17 IC netlists from the trust-hub.com. Most of the testing results are pretty well. However, in the repeat experiment, they just used the internal nets of the HTs, ignored numerous boundary network between the normal nets and HT nets. From experiments, we consider this ignorance may lead to good results. Besides, many experimental results in the result table were rounded up to 100%, and the more accurate results data should be preferred. Bao *et al.* [23] proposed a new way that combined machine learning with HTs detection. They transformed circuits into images and broke the images into smaller grids to parallelize or distribute the processing for each grid. Then they extracted the relevant image features, trained a classifier, and obtained a decision boundary. After training, they classified the grids in each layer of all the n chip areas as normal or malicious ICs based on the v-SVM decision boundaries of each layer. Determine a label for each chip based on these grid classifications. They also introduced the K-Means algorithm to solve the poor generalization ability of SVM. However, image features may not fully describe the characteristics of the circuits, especially in the VLSI. The more complex the netlist, the less accurate the features and the worse the results may be. Xue *et al.* [24] formulated the HTs detection problem into a two-class classification problem. Then they trained the classification algorithms using simulated ICs during the IC design flow. The trained algorithms will then form a classifier which can automatically identify fabricated ICs as HT-free or HT-inserted during test-time. Considering that there may be a shift that occurs between the Monte Carlo simulation and actual silicon, they also proposed several optional optimized methods. The problem is they use the transient power supply as the feature of learning of classifiers, which needs extra pay. Also, the HTs are made by them as the benchmarks. It may have a bad result on other HTs. Dong *et al.* [25] came up with a framework called RG-Secure. They used the lightGBM algorithm to detect gate-level netlists from Trust-HUB, reaching nearly 100% TPR and 94% TNR. However, this method discards some circuits on the Trust-HUB, and its high detection effect can only be reflected in these circuits.

Here is a few relatively authoritative literature on HT locating, the most of methods are based on side-channel measurement technology, which is also the future research direction of channel measurement technology. Bazzazi *et al.* [26] proposed a method to detect and locate the HTs based on the real-time logical values of nodes. The algorithm can extract the nodes with special attributes. When the numbers of these nodes come to be sufficient, the logical relationship between these nodes yields a function, the logical values of which differ in the Trojan-free and Trojan-inserted nodes, thus they detected the potential HTs. Zhong *et al.* [19] proposed a Hardware Trojan detection method based on differential temperature matrix analysis. According to a matrix with each pixel's differential temperature value from the thermal image, the difference between the testing chips and the golden chip was obtained, and then the location of HTs was obtained. Nasr *et al.* [27] were the first to propose the approach of HT detection that is based on automatic feature extraction and machine learning algorithms. They used a tool named Haar feature extractor to determine the basic features automatically. With combining the AdaBoost learning algorithm to improve the original cascaded classifiers, they built up an automatic Hardware Trojan location and detection tool. While this solution did not realize the ultimate purpose of HT locating, that is the removing of HTs. So the ultimate goal of Hardware Trojan locating is to accurately locate the Hardware Trojan and all gate-level components associated with it.

III. PRELIMINARIES AND DEFINITIONS

The systems and threat models, a common HT circuit, the HTs of control signal type and information leakage type, the multi-layer back propagation neural networks and the one-class SVM are described in section III.

A. SYSTEM AND THREAT MODELS

The system model for the basic HT detection system model comprises four types of entities, namely: Attackers(**AT**s), Untrusted Gate-level netlist(**UGN**s), Feature set(**FS**) and The operators(**TO**s)-see Fig.1. It is worth noticing that the hardware Trojan comes primarily from attacks with **AT**s at the design stage, possibly in the specification, possibly at the RTL, possibly at the gate-level. Before the chip is put into production, any deliberate modification to the circuit may be a Hardware Trojan. The circuit that is intentionally modified after the manufacturing and packaging test results in an untrusted circuit. Therefore, before the chip is manufactured, the easy-to-handle gate-level netlist is processed, which has a great probability of reducing the harm caused by the Hardware Trojan. The steps in the model diagram will be described in detail below:

- **First step: TO**s analyzes the circuit that will be put into the manufacturing site, and the target is locked in the gate-level netlist stage. Since this phase does not involve the function of the circuit, only the logical structure information of the circuit needs to be processed.
- **Second step: TO**s extracts the features of these possible **UGN**s and obtains a **FS**. Based on the characteristics that

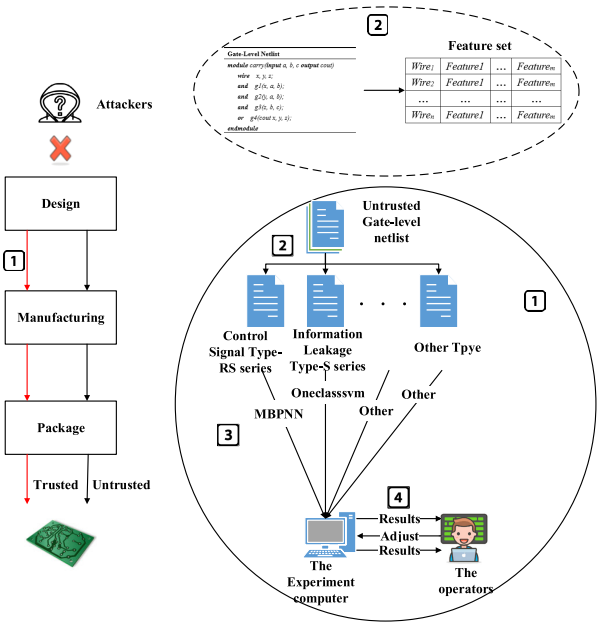


FIGURE 1. Basic HT detection system model.

HTs are difficult to activate, the features are specifically selected.

- **Third step: TO**s uses different types of algorithms to detect different functions of HTs. The problem here is how to judge the function of the unknown netlist and adopt the corresponding algorithm. The current research in this paper is to detect the HT circuit under the premise that the function, structure, and scale of the HT are known. The future work should be based on the structure of the circuit to judge the Hardware Trojans that may exist in the circuit, and then use different algorithms to detect the unknown circuits in turn. This part of the work remains to be studied.
- **Fourth step: TO**s obtains the results of the first experiment through the computer. The result of the first time may be poor, at least not the best. Constantly adjust the algorithm parameters to obtain the current optimal results, and then perform other operations such as the positioning proposed in this paper.

After all the step is processed by **TO**s, in an ideal state, a trusted chip can be obtained.

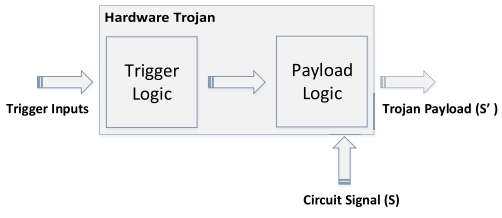


FIGURE 2. A basic HT structure.

B. A COMMON HT CIRCUIT

Fig.2 shows a simplified block diagram of an HT, which causes a malfunction (by modifying signal S to S') [9]. The composition of HTs is elementary, consisting of a trigger module and a payload module. These modules are no different from ordinary circuits, and there are many similar modules in the general circuit. The normal circuits are also operated by triggers and payloads, so the Trojans cannot be detected by these modules. The trigger module receives a special circuit parameter signal. After disposing, the payload module output the result. It is such a simple HT, but with a high degree of concealment and harm. The HT can keep silent most of the time due to their rare activation conditions, so it is tough to discover the malicious circuits. When the specified signal or event appears, the payload circuit is activated and implements malicious functions. By cleverly designing, the rare signals or events are unlikely to arise during design simulation or testing but can occur during a long period of field operation [28].

C. HTS OF CONTROL SIGNAL TYPE AND INFORMATION LEAKAGE TYPE

Here we name the HTs of control signal type as the RS series, and we name the HTs of information leakage type as the S series. The names of these two series of Hardware Trojans are also the beginning of the names of the test circuits used in the experiments. Also, it can be found in [29] that the HTs in RS series belong to the functional type, whose HT trigger is a sequential comparator whenever the HTs get triggered, its payload gains some internal signals. Nevertheless, the HTs in the S series have diverse functions including DOS, information disclosure, controlling internal signals, and so on.

D. MULTI-LAYER BACK PROPAGATION NEURAL NETWORKS

Considering that BP neural network has nonlinear mapping ability, it can transform the nonlinear problem into a linear problem. At the same time, it can adaptively learn in the training process, with high generalization ability and fault tolerance. Therefore, this paper gives priority to BP neural network as a classification algorithm. Drawing on the literature [30], the ''Multi-layer Back Propagation (BP) neural networks'' is redefined as the BP neural networks with at least one hidden layer.

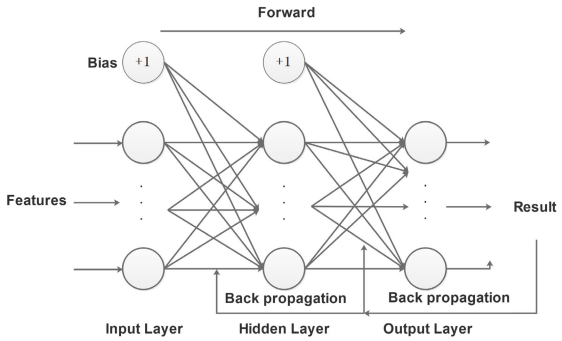


FIGURE 3. BP neural network.

As shown in Fig.3, it is known that there are three basic layers in the BP neural network, which are the input layer,

the hidden layer, and the output layer. In addition to the output layer, the input layer and each hidden layer have a bias value. The function of the bias is to make the activation function move, which can better achieve the purpose of classification. Therefore the weight of the input layer to the hidden layer is set to υ_{ih}. The threshold of the h-th neuron in the hidden layer is set to γ_h. The weight between the hidden layer and the output layer is set to ω_{hj}. The threshold of the j-th neuron in the output layer is expressed by θ_{hj}.

Suppose that this BP neural network model has d input neurons, q hidden neurons, the hidden layer has q hidden neuron thresholds, and l output neurons, so there are l output neuron thresholds.

The input of the j-th output neuron is

$$\beta_j = \sum_{h=1}^{q} w_{hj} * b_h, b_h = f(\alpha_h - \theta_h) \tag{1}$$

the f is the activation function (here is the sigmoid function); The input of the q-th hidden neuron is

$$\alpha_h = \sum_{i=1}^{d} \upsilon_{ih} * x_i \tag{2}$$

The activation functions of the hidden layer and the output layer are both using the Sigmod function.

$$Sigmoid(x) = \frac{1}{1+e^{-x}} \tag{3}$$

Assume that in a training example(x_k, y_k), the training output of the neural network is $y_k = (y_1^{k'}, y_2^{k'}, \ldots, y_l^{k'})$

$$y_i^{k'} = f(\beta_i - \theta_i) \tag{4}$$

Then the error of this prediction result can be expressed by the least squares method:

$$E_k = \frac{1}{2}\sum_{j=1}^{l}(y_j^{k'} - y_j^k)^2 \tag{5}$$

BP neural network uses gradient descent method to minimize this error. The first value is the weight adjustment value from the hidden layer to the output layer:

$$\Delta\omega_{hj} = -\eta\frac{\partial E_k}{\partial \omega_{hj}} \tag{6}$$

1. Input layer to hidden layer:

$$\alpha_h = \sum_{i=1}^{d} \upsilon_{ih} * x_i \tag{7}$$

which expressed by matrix is

$$|x_1\ x_2\ x_3\ \ldots\ x_d| \cdot \begin{vmatrix} \upsilon_{11} & \upsilon_{12} & \upsilon_{12} & \ldots & \upsilon_{1q} \\ \upsilon_{21} & \upsilon_{22} & \upsilon_{23} & \ldots & \upsilon_{2q} \\ & \ldots & \ldots & & \\ \upsilon_{d1} & \upsilon_{d2} & \upsilon_{d3} & \ldots & \upsilon_{dq} \end{vmatrix}$$

2. The activation function after the hidden layer:

$$b_h = f(\alpha_h - \theta_h) \tag{8}$$

3. Hide layer to output layer:

$$\beta_j = \sum_{h=1,j=1}^{q,l} w_{hj} * b_h \tag{9}$$

which expressed by matrix is

$$|b_1\ b_2\ b_3\ \ldots\ b_d| \cdot \begin{vmatrix} \omega_{11} & \omega_{12} & \omega_{12} & \ldots & \omega_{1l} \\ \omega_{21} & \omega_{22} & \omega_{23} & \ldots & \omega_{2l} \\ & \ldots & \ldots & & \\ \omega_{q1} & \omega_{q2} & \omega_{q3} & \ldots & \omega_{dl} \end{vmatrix}$$

4. After the activation function of the output layer:

$$y_i^{k'} = f(\beta_j - \theta_j) \tag{10}$$

5. Error

$$E_k = \frac{1}{2}\sum_{j=1}^{l}(y_j^{k'} - y_j^k)^2 \tag{11}$$

In summary, we can know that ω_{hj} affects β_j first, then $y_j^{k'}$, and finally affects E_k. (One ω only affects one β) So we can get:

$$\Delta\omega_{hj} = -\eta\frac{\partial E_k}{\partial \omega_{hj}} = -\eta\frac{\partial E_k}{\partial y_j^{k'}} \cdot \frac{\partial y_j^{k'}}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial \omega_{hj}} \tag{12}$$

η is the learning rate. Learning rate is an important parameter in supervised learning and deep learning. It determines whether the objective function converges to the local minimum and when it converges to the minimum. If the learning rate is set too small, the convergence process would become very slow. While the learning rate is set too large, the gradient may oscillate back and forth around the minimum value, and may not even converge. Where $\frac{\partial \beta_j}{\partial \omega_{hj}} = b_h$, as mentioned earlier, b_h is the output of the h-th hidden neuron.

$$g_j = \frac{\partial E_k}{\partial y_j^{k'}} \cdot \frac{\partial y_j^{k'}}{\partial \beta_j} = (y_j^{k'} - y_j^k) \cdot f'(\beta_j - \theta_j) \tag{13}$$

And the activation function we chose is the Sigmod function, which has a very good property.

$$f(x) = \frac{1}{1+e^{-x}} \ldots f'(x) = f(x)(1 - f(x)) \tag{14}$$

So

$$\begin{aligned} f'(\beta_j - \theta_j) &= f(\beta_j - \theta_j) \cdot (1 - f(\beta_j - \theta_j)) \\ &= y_j^{k'} \cdot (1 - y_j^{k'}) \end{aligned} \tag{15}$$

Combine equations (9), (10), (12)

$$\begin{aligned} \Delta\omega_{hj} &= -\eta\frac{\partial E_k}{\partial \omega_{hj}} = -\eta g_i b_h \\ &= -\eta(y_j^{k'} - y_j^k) \cdot y_j^{k'} \cdot (1 - y_j^{k'}) \cdot b_n \end{aligned} \tag{16}$$

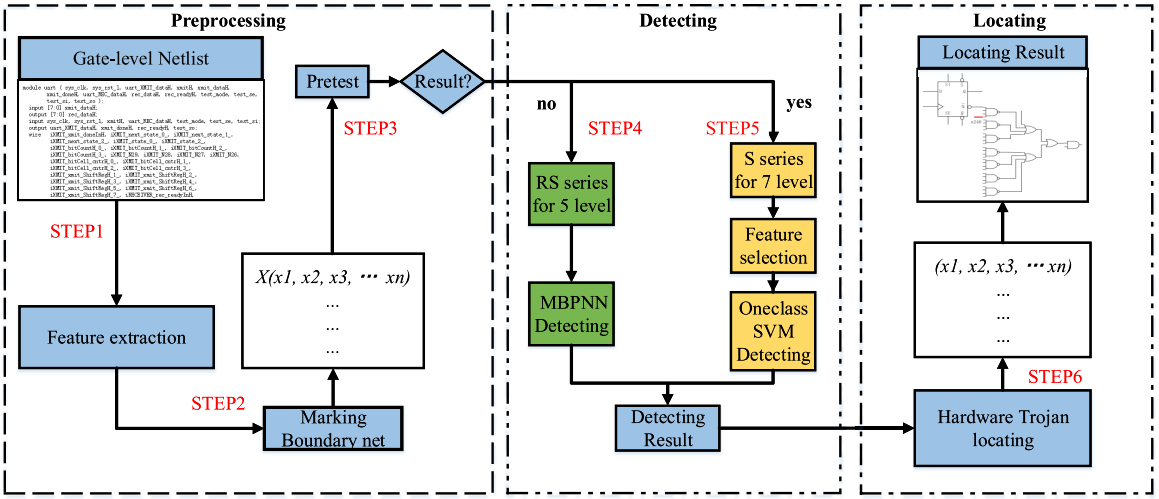


FIGURE 4. Hardware Trojan locating flow chart.

The same reason

$$\Delta\theta_j = -\eta\frac{\partial E_k}{\partial \theta_j} = -\eta\frac{\partial E_k}{\partial y_j^{k'}}\cdot\frac{\partial y_j^{k'}}{\partial \theta_j} = \eta\cdot g_j \tag{17}$$

Now focus on the value of $\Delta\upsilon_{ih}$, or derive it from equations (4), (5), (6), (7), (8). One υ weight will affect all β.

$$\Delta\upsilon_{ih} = -\eta e_h x_i \tag{18}$$

$$\Delta\gamma_h = \eta e_h \tag{19}$$

$$\begin{aligned} e_h &= (\sum_{j=1}^{l}\frac{\partial E_k}{\partial \beta_j}\cdot\frac{\partial \beta_j}{\partial b_j})\cdot f'(\alpha_h-\gamma_h) \\ &= (\sum_{j=1}^{l}(y_j^{k'}-y_j^{k})\cdot f'(\beta_j-\theta_j)\cdot\omega_{hj}) \end{aligned} \tag{20}$$

Finally, an iterative termination condition is set, which is generally the end of the algorithm when the test error reaches a certain threshold.

E. ONE-CLASS SVM

The one-class SVM was put forward by Bernhard Scholkopf in [31]. The algorithm aims to construct a hypersphere (depending on the dimension of the training data). It wraps one class with more amount in this high-dimensional hypersphere. Another class is excluded outside the hypersphere, achieving the purpose of single classification.

The detailed model is as follows:

$$\min_{\omega\epsilon F,\xi\epsilon R^l,\rho\epsilon R}\ \frac{1}{2}\parallel\omega\parallel^2+\frac{1}{\nu\iota}\sum_i\xi_i-\rho. \tag{21}$$

$$subject\ to\ (\omega\cdot\Phi(x_i))\geq\rho-\xi_i,\quad \xi_i\geq 0. \tag{22}$$

There are some parameters in the one-class SVM affect the result of the algorithm. Φ is the mapping from x to f, l is the number of observations, ξ is the non-zero slack variable, and ω and ρ are the final required values.

$\boldsymbol{\gamma}$: the kernel coefficient, the default value is equal to 1/features. $\boldsymbol{\nu}$: the patience to the error class in the training set. It ranges from 0 to 1, the default is 0.5. Above these, two parameters are the main tuning objects in the experiment.

IV. HARDWARE TROJAN LOCATING BASED ON MACHINE LEARNING

A. OVERALL ARCHITECTURE OF ML-HTCL

After completing the experiment of HTs detection using different algorithms, locating the HTs in the netlist is the next work.

At present, there are very few articles about Hardware Trojan locating, and most of them only locate the HTs from the circuits' image. It can help users to abandon these dangerous ICs but has no help to prevent the spreading of HT-inserted ICs. Only starting from the fabrication of the ICs, focusing on the netlists can help to clean the HTs from the root.

Fig.4 is the block diagram of ML-HTCL, which shows the entire content of our theoretical model. The orange parts are unique to the S series circuit, the green parts are unique to the RS series circuit, and the blue parts are both. The procedure of ML-HTCL is summarized as follows:

- **Stage I: Preprocessing**
  - **STEP 1**. Input the gate-level netlist. Draw the logical structure diagram of all HT circuits, and mark the lines at the edge of the HT circuit as the boundary network.
  - **STEP 2**. Extract the netlist features, and then label them to create the training and testing data set, meanwhile, label the boundary network as HT.
  - **STEP 3**. Pre-train and pretest the all data set by using BPNN. If TPR and TNR are both above 85%, go to the **STEP 5**, if not, go to the **STEP 4**.
- **Stage II: Detecting**
  - **STEP 4**. Reduce the feature dimension of data set by employing the random forest algorithm, and then

TABLE 1. The extracted features from a netlist.

HT-net feature candidate	Description($x \in [1,5]$ in RS series, $x \in [1,7]$ in S series)
fan_in_x	The number of logic-gate fanins x-level away from the net n.
in_flipflop_x	The number of flip-flops up to x-level away from the input side of the net n.
out_flipflop_x	The number of flip-flop up to x-level away from the output side of the net n.
in_multiplexer_x	The number of multiplexers up to x-level away from the input side of the net n.
out_multiplexer_x	The number of multiplexers up to x-level away from the output side of the net n.
in_loop_x	The number of up to x-level loops from the input side of the net n.
out_loop_x	The number of up to x-level loops from the output side of the net n.
in_const_x	The number of constants up to x-level away from the input side of the net n.
out_const_x	The number of constants up to x-level away from the output side of the net n.
in_nearest_pin	The minimum level to the primary input from the net n.
out_nearest_pout	The minimum level to the primary output from the net n.
{in,out}_nearest_flipflop	The minimum level to any flip-flop from the input or output side of the net n.
{in,out}_nearest_multiplexer	The minimum level to any multiplexer from the input or output side of the net n.

train and test by using one-class SVM. Calculate the TPR and the TNR.
  - **STEP 5**. Train and test the data set by using the MBPNN algorithm and calculate the TPR and the TNR.
- **Stage III: Locating**
  - **STEP 6**. Put the netlist feature set which obtained in **STEP 2** as the judgment standard data set. Match the features of the test data set with the judgment standard data set, and obtain the net's names in the testing set and the detecting results. Draw the HT circuit, and mark the HT wire which has been detected with a black line, and the undetected HT wire with a red line. The experiment is finished.

B. HARDWARE TROJAN FEATURE

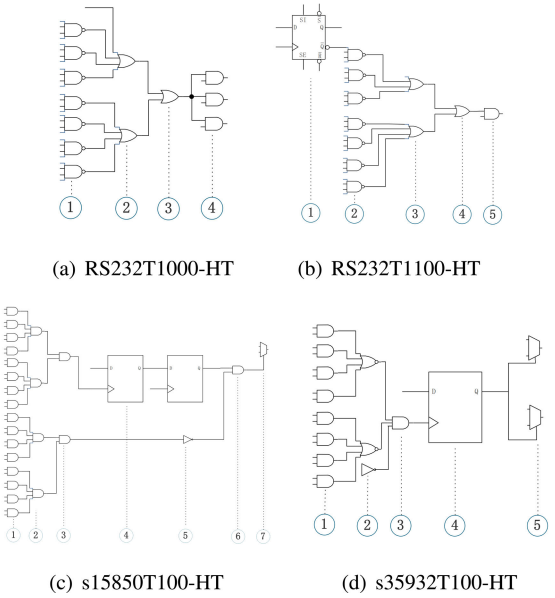
Taking into account the experimental results of the article, these 51 features from [8] are feature candidates, which are shown in Table 1. In the [8], the concept of HT circuit features was first proposed. Since Hardware Trojan triggering is a small probability event, they summarized the possible HT characteristics and proposed the concept of LSLG (low-switching logic gate). LSLGs have lower switching probabilities, so they think that a trigger circuit is mainly composed of LSLGs. The characteristics of HT circuits can be divided into 9 cases. According to these 9 cases, these 51 features can be defined. Take the first feature "fan_in_x" for example, it means in the direction of input to output, for an arbitrary net, the number of the input of all the logic gates in the level near this arbitrary net, the x is the level. The rest features are similar.

We study all the HTs structure in the benchmarks and find the logic range of the HTs in the testing RS series is from 1 to 5, but the logic range of HTs in the testing S series is from 1 to 7, which means in the netlists of the RS series and S series, from the input to the output or to the opposite, the first logic gate or flip-flop or multiplexer of the HT belongs to the first level, the second belongs to the twice level and so on. The last logic gate or flip-flop or multiplexer of the HT belongs to the last level. The highest level of the RS series is 5, but which of S series is 7.

When testing the S series netlist, the test results are very unsatisfactory. Almost no HT nets can be identified. In the beginning, we suspected that it was a problem with the classifier. However, after continually adjusting the parameters, The results of the test are still not ideal. So we compared the training set of the S series with the training set of the RS series. It can be found that the ratio of the number of all normal nets and that of HT nets in the RS series is 1:5, but the ratio of the S series is 1:40. At the same time, it can be found that these two types of netlists have different functions and HT structures. So far, almost all the relevant papers have not realized this.

Considering that the maximum levels of the two netlists are different and the two netlists carry different HTs, we will respectively deal with the benchmarks with different features because of this difference. This is a considerable attempt for HT detection different from past papers. The [32] only tested the netlists of the RS series, but they did not realize the essential difference between different series netlists, they focused on researching the logic structure of the misidentified net. The final experimental results were not satisfactory enough. The paper [30] did not consider that different purposes of HTs should be detected by different methods, and it ignored some difficult-to-detect boundary networks when performing HTs detection. On the contrary, the ML-HTCL method considers HTs with different functions may have different structures, so their circuit characteristics should be modified (evidently, this information is proved when we draw the circuit diagram). For the HT of RS series and S series, ML-HTCL proposes different detection approach and circuit features, and also adds all the boundary network to a data set that is difficult to handle.

The netlist of RS series and S series are separated. For the RS, the x ranges from 1 to 5, but for the S, the x ranges from 1 to 7. Note that in different series of netlists, the model has



(a) RS232T1000-HT (b) RS232T1100-HT

(c) s15850T100-HT (d) s35932T100-HT

FIGURE 5. The HTs structure in part netlists.

different feature vector depending on x. Some parts of the HT structure with the most representative are shown in Fig.5. The black line represents the wire mesh of the HT circuit. In the vertical direction, the label below each sub-picture represents the level of all components on the column of the label. The basic components in the circuit are arranged in order, and the first column is the first level, the second column is the second level, and so on. This concept is proposed to give readers a better understanding of the meaning of Trojan features. Fig.5a and Fig.5b belong to the RS series, Fig.5c and Fig.5d belong to S series. Fig.5a is the netlist with the most level in the RS series, and it has five levels. Fig.5c is the netlist with the most level in the S series, and it has seven levels.

The MBPNN is used to detect the RS series netlist. The detection has a good experimental result, almost all HTs can be detected, and there is almost no misidentification in the normal circuit, so we decided to use the 51 features listed in Table 1 as the characteristics of the RS series HT. The one-class SVM is used to detect the remaining S series netlists in the benchmark set because of the unbalance problem of two class training data. The number of the features is altered considering the different structures of the HT nets. Extract these features programmatically with a range of x from 1 to 7. When analyzing these features, it is found that some features are always 0 in all test netlists. Considering that these features may be unnecessary features, 20 valid features are retained. These valid features are shown in Table 2.

C. HARDWARE TROJAN BOUNDARY NETWORK

In this part, we redefine the concept of the boundary network [32]. The reason for redefining the Hardware Trojan boundary network is that this intersection of the circuit belongs to the fuzzy judgment part, that is, when it is judged

TABLE 2. The extracted features.

The serial number	Feature
1	fan_in_1 to fan_in_5
2	in_const_2 to in_const_5
3	in_flipflop_2 to in_flipflop_5
4	in_nearest_flipflop
5	in_nearest_pin
6	out_const_1 to out_const_5

manually, only half of it can be regarded as an HT circuit, and half is regarded as a normal circuit. This makes it impossible to explicitly define a label for this part when using a supervised learning algorithm for experimentation and will cause confusion.

As shown in Fig.6, this is a common Hardware Trojan structure in the experimental. All black lines (including logic gates) in the black dotted frame are a complete Hardware Trojan, the function of this HT is to control part of the signal inside the circuit. All the red lines outside the black dotted frame (including the internal part of the red line) are connected to the components of the normal circuit where the Trojan is located in (including as the input and output). In this figure, if draw these red lines into a circle, all the lines that the circle passes are the boundary network of the circuit.

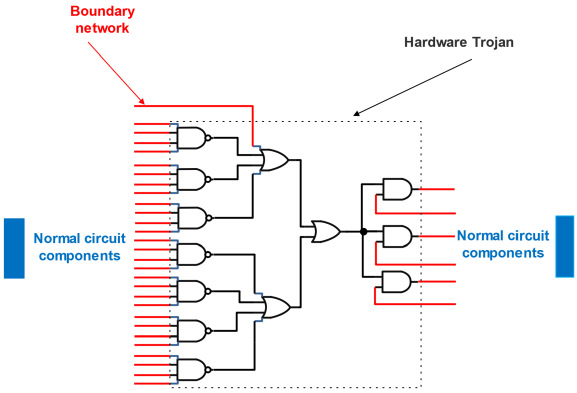


FIGURE 6. Boundary network diagram.

As the name implies, the boundary network is the boundary between the normal circuit and the HT circuit. Of course, here the HT is only considered to be connected to the normal circuit only through the net without common components. (logic gates, registers, triggers, etc.).

The shared components mentioned above do not appear in this experiment, so they are not considered here, but when referring to the whole Trojan study, The part of the Trojan and the normal circuit connected can be unified called the boundary field, which includes the boundary network and common components, this part of the content is still under our study.

As shown in Fig.7, this is a locating result of using the method in [30] for RS232-T1000 netlist detection. The red

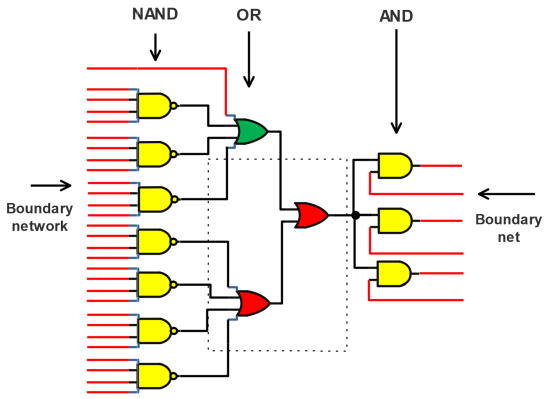


FIGURE 7. The locating result of [30].

wires belong to the boundary network, and the black wires belong to the HT. In the [30], the boundary network was neglected. When locating, only the components in the black dotted frame (i.e., the red component and the connected wire mesh) were detected. For these yellow components (NAND), among all the connected wire nets, the number of nets considered to be normal nets is larger than the number of nets considered to be HT nets, so these yellow components have a high probability of being recognized as normal components; For this green component (OR), the number of nets in all connected nets is considered to be the number of nets of the HT net than the number of nets considered to be normal nets. So this green component has a high probability of being recognized as HT components. Therefore, it is difficult to detect Hardware Trojan using the past method completely [30]. Even if some rules [32] are used to make up for the drawbacks caused by the neglect of the boundary network, the HTs cannot be completely detected.

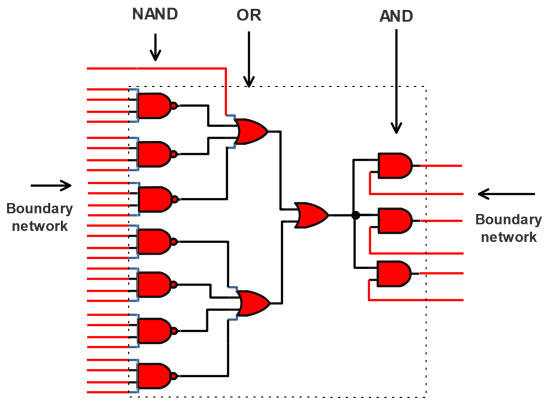


FIGURE 8. The locating result of ML-HTCL.

When the boundary network is under consideration, the locating of the HT becomes easy. As shown in Fig.8, the red net is the boundary network. At this time, our method regards the boundary network as HT nets. Then the component inside the black dotted frame covering all the boundary network is the HT we have located, and it is the real complete Hardware Trojan in the RS232-T1000 netlist. All red components and the black net can be identified. There are no abnormal components or ignoring of HT components. Therefore, HT locating work is completed.

D. MACHINE LEARNING METHODS

1) MULTI-LAYER BACK PROPAGATION NEURAL NETWORKS (MBPNN)

It is well known that changes in the structure of the neural network will have an impact on the classification results. We experiment by constantly changing the number of the Hidden layer and the number of units in the Hidden Layer. The number of the Hidden layer is from 1 to 3, and the number of the units is 10, 12, 15, 20, 50, 100, 200, 500.

The input layer has 51 units in the experiment, which means per wire is represented by a 51-dimension vector, and these 51 features are listed in Table 1. The output layer is only 1 unit. The goal value of a normal net is 1, but the goal value of an HT net is 0. If the real output ranges from 0.5 to 1, it is an HT net; If the real output ranges from 0 to 0.5, it is a normal net. Considering the boundary network (Here we define the boundary network as all the nets that connect the HT circuit to the normal circuit. These nets are very vague in the ownership of the HT circuit and the normal circuit.), although there is no value of 0.5 in the experiment, it will be regarded as an HT net here. When the number of Hidden layers is different, the number of per layer will change, the detail will be introduced in section V.

2) ONE-CLASS SVM

As is shown in Table 3, the proportion of all the normal nets and HT nets of S series is slightly larger, which causes the classifier to ignore the HT nets in the training data set. So when inputting the testing data set containing HT nets and normal nets, the HT nets cannot almost be identified. Considering the situation, One-class Support Vector Machine (one-class SVM) is used to solve the problem of uneven quantity of positive and negative samples. Using random forest, one-class SVM, isolated forest, and other outlier detection algorithms to test some data randomly selected, one-class SVM is selected as a further experimental test algorithm because it has a better detection effect. After comparing some algorithm like the random forest, one-class SVM, isolated forest and so on, the algorithm with the best results is chosen.

E. EVALUATION INDEXES OF MACHINE LEARNING

In the dichotomy problem, there are usually four indexes to evaluate the classification results: the true negative value (TN), the false positive value (FP), the true positive value (TP) and the false negative value (FN); In our experiment, TN is the number of HT nets identified to be HT nets; FP is the number of normal nets identified to be HT nets mistakenly; TP is the number of normal nets identified to be normal nets; FN is the number of HT nets identified to be normal nets mistakenly. These indexes can be used to judge the quality of the results.

TABLE 3. Benchmarks.

Netlist	num of the HT nets	num of the normal nets
RS232-T1000	44	211
RS232-T1100	44	212
RS232-T1200	45	211
RS232-T1300	31	222
RS232-T1400	50	205
RS232-T1500	48	209
RS232-T1600	39	216
s15850-T100	61	2371
s35932-T100	34	6368
s35932-T200	40	6359
s35932-T300	59	6365
s38417-T100	29	5772
s38417-T200	35	5769
s38417-T300	31	5802
s38584-T100	21	7271
s38584-T200	198	7274
s38584-T300	976	7275

According to the four indexes, many other indicators can be calculated for better evaluating classification results, such as the true positive rate (TPR), the true negative rate (TNR), the recall (R), the precision (P) and the accuracy (Acc), which show the effect of the result in different direction. The TPR and TNR are the main indexes in the paper.

V. EXPERIMENTS AND RESULTS

In this section, the detail experiment results are demonstrated, and ML-HTCL is compared with some similar methods. In order to compare with the existing methods, 17 benchmark circuits are used and shown in Table 3. Moreover, the statistics number of normal nets and HT nets are counted.

Though the same benchmark is used, the number of HT nets and normal nets is different from these papers. The number of nets in the extracted netlist cannot be guaranteed to be the same as the real circuits, but this is indeed the nets extracted using our program. The testing HT nets include the boundary network which is more difficult in other papers to detect and perform well in our experiment.

The experiment environment is Intel(R) Core(TM) i5-6500 computer environment with an 8.00GB memory. The netlist features are extracted by g++ in Ubuntu 18.04.1 LTS. The multi-layer BP neural network is achieved in MATLAB R2014a, and the one-class SVM is implemented with Python 3 in PyCharm.

The leave-one-out cross-validation [33] is used in the testing. 16 benchmarks are employed as the training data set for every run, the rest one as the testing data set. Our purpose is to get an excellent classifier to look for the HT nets in an unknown netlist (not as the training set).

In most of the papers about machine learning, some evaluation indexes are used such as the TN, TP, FN, FP. Based on these indexes, some new indexes can be calculated:

$$TPR = TP/(TP + FN) \tag{23}$$
$$TNR = TN/(TN + FP) \tag{24}$$
$$Precision = TP/(FP + TP) \tag{25}$$
$$Recall = TP/(TP + FN) \tag{26}$$
$$F - measure = 2PR/(P + R) \tag{27}$$
$$Accuracy = (TP + TN)/(TP + TN + FN + FP) \tag{28}$$

For testing the benchmarks respectively, the mean value of the two series netlist is regarded as the final result. While, if the final TPR and TNR are close enough to make a difficult decision, the other indexes will be considered for further comparison.

A. EXPERIMENT 1: DETECTION FOR CONTROL SIGNAL HTS

Here, the ML-HTCL will be tested by the RS series netlists separately, which is from [22]. Firstly, let the number of the hidden layer be 1, 2 and 3. The units of the input layer are equal to the number of the HTs' features (here is 20). The unit of the output layer is one. And when the number of the hidden layer is one, let the number of the units to be 10, 12, 15, 20, 50, 100, 200, and 500. Finally, the average TPR and TNR of all the netlists are calculated by using leave-one-out cross-validation. Table 4 shows the results of the experiment.

TABLE 4. Result of one hidden layer.

Number of the units	TPR	TNR
10	99.80%	88.70%
12	99.81%	90.33%
15	99.87%	90.76%
20	99.74%	87.54%
50	99.87%	91.18%
100	99.87%	91.80%
200	99.87%	91.91%
500	99.93%	91.49%
Average value	99.85%	90.46%

As the number of units increases, the classification effect is not improved when only one hidden layer. So the hidden layer is added to be 2, and let the units of the first hidden layer to be 20, 50, 100 and 200, the units of the second hidden layer to be 10, 12, 15, 20, 50, 100, 200, and 500. The results are shown in Table 5.

When the hidden layer is added to two, the experimental results are slightly improved but it still is not satisfied. So another one layer is added again to the current network. The units of the first hidden layer are 50, 100 and 200. The units of the second hidden layer are 20, 50, 100 and 200. The units of the third hidden layer are 50, 100 and 200. Table 6 shows the final results.

During the experiments, it is found that the results are not improved continuously with adding units or hidden layers,

TABLE 5. Result of two hidden layer.

Number of units		Average values	
1st layer	2nd layer	TPR	TNR
20	10	99.74%	87.92%
	12	99.80%	89.41%
	15	99.67%	89.66%
	20	99.46%	88.49%
	50	99.80%	88.36%
	100	99.87%	88.20%
	200	99.80%	88.17%
	500	99.22%	89.69%
	Average	99.67%	88.76%
50	10	99.80%	92.61%
	15	99.60%	89.09%
	12	99.80%	90.63%
	20	99.66%	90.43%
	50	99.74%	91.85%
	100	99.75%	90.35%
	200	99.87%	90.04%
	500	99.81%	88.90%
	Average	99.75%	90.49%
100	10	99.80%	90.81%
	12	99.87%	92.18%
	15	99.67%	91.59%
	20	99.87%	91.18%
	50	99.87%	90.89%
	100	**99.93%**	**92.22%**
	200	**99.80%**	**92.54%**
	500	99.81%	91.07%
	Average	99.83%	91.56%
200	10	99.87%	91.25%
	12	99.87%	91.03%
	15	99.87%	91.49%
	20	99.53%	92.21%
	50	99.87%	90.79%
	100	99.93%	91.55%
	200	**99.87%**	**92.25%**
	500	99.67%	90.14%
	Average	99.81%	91.34%

TABLE 6. Result of three hidden layer.

Number of units			Average values	
1st layer	2nd layer	3rd layer	TPR	TNR
20	20	50	99.73%	87.82%
		100	99.74%	88.27%
		200	99.80%	87.52%
	50	50	99.87%	90.83%
		100	99.81%	89.48%
		200	99.87%	88.87%
	100	50	99.73%	88.76%
		100	99.93%	87.56%
		200	99.87%	90.31%
	200	50	99.93%	90.72%
		100	99.87%	90.64%
		200	99.81%	91.18%
	Average		99.83%	89.33%
50	20	50	99.93%	91.50%
		100	99.93%	88.23%
		200	99.80%	89.37%
	50	50	99.93%	90.57%
		100	94.82%	90.90%
		200	99.15%	90.55%
	100	50	99.87%	90.49%
		100	99.93%	90.92%
		200	99.93%	89.84%
	200	50	99.79%	91.14%
		100	99.85%	89.80%
		200	99.93%	91.20%
	Average		99.41%	90.38%
100	20	50	99.73%	90.60%
		100	99.87%	91.08%
		200	99.73%	90.52%
	50	50	99.87%	91.84%
		100	99.83%	89.49%
		200	99.93%	90.95%
	100	50	99.87%	89.25%
		100	99.87%	91.14%
		200	93.71%	97.74%
	200	50	99.93%	91.85%
		100	99.87%	91.02%
		200	**99.93%**	**92.10%**
	Average		99.35%	91.47%

so the test is stopped here to analyze the results. Four relatively good results are selected for comparison. Because they have very close TPR, TNR and error rate, the precision, the Recall, the F-measure, and the Accuracy of all the four data are calculated. The comparison results are shown in Fig.9. It is clear that the data with units of 100, 100, and the data with units of 200, 200, 200 is better. Finally, the former is chosen as the best result due to the TPR and TNR, which are 99.93% and 92.22%.

Finally, the TPR is satisfactory, but the TNR still needs further improvement. High time consumption is the biggest problem in this experiment. It is an unavoidable problem due to the design of the algorithm itself, such as so many units. Besides this, there is another problem. Analyzing the results, some nets belonging to original HT nets are always identified as normal nets, and some nets belonging to original normal nets are always identified as HT nets. We consider that if further work can be done in the feature selection of the misidentified nets, the experimental results should be improved extremely.

B. EXPERIMENT 2: DETECTION FOR INFORMATION LEAKAGE HTS

Different from the RS series above, the feature number of the S series is extended to 69 (the range of x becomes 1-7). The random forest algorithm is used to assess the importance of 69 features, and 20 more characteristic features

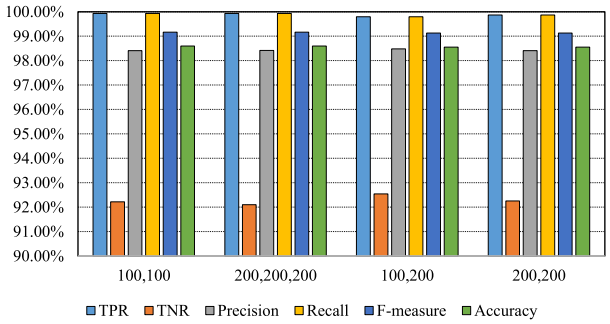


FIGURE 9. The comparison of the result.

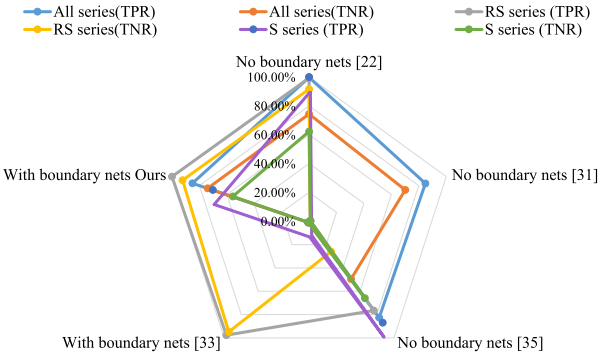


FIGURE 10. ML-HTCL exceeds all the existing methods in the RS series except the TNR and Precision of [32]. However, compared with [22], [30], [34], if we pay attention to the S series, ML-HTCL is not good enough. This is because [22], [30], [34] did not consider the problem of the boundary, but we considered it.

are obtained. The processing data set is divided into a training set and a testing set. ML-HTCL is trained by the training data set and then tested by the testing data set. When a net is an HT circuit, the output is -1, and when a net is a normal circuit network, the output is 1.

TABLE 7. The experiment result of One-class SVM.

Gamma(NU=0.06)	TPR	TNR	(ave)TPR	(ave)TNR
0.1	68.98%	55.13%	84.46%	73.68%
0.11	67.12%	63.69%	83.53%	77.96%
0.12	64.12%	63.32%	82.03%	77.77%
0.09	69.22%	56.00%	84.58%	74.11%
0.094	70.16%	55.59%	85.05%	73.91%
0.095	67.71%	53.95%	83.82%	73.09%
0.096	71.17%	53.5%	85.55%	72.86%

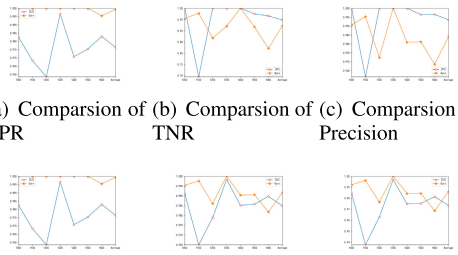
In order to get the best performance, two vital parameters are changed in the experiment: γ and ν. When the ν is equal to 0.06, the effect is relatively better. It is to say. This classifier allows 6 percent of anomalous training data, and these anomalous training data is the HT nets here for its small number. However, in fact, it is not equal to the real abnormal data percentage. It is a future problem that need to be studied. Different γ is tested constantly. The detail experiment result is shown in Table 7. The final average TPR and TNR are calculated by combining the result of the RS series with the result of the S series. We hope to get a higher TNR, so the data which the γ is 0.094 is the final result with the final TPR of 85.05%, the final TNR of 73.91%.

C. HARDWARE TROJAN LOCATING AND THE COMPARISON WITH OTHER METHODS

Our detecting experiment result performs better than most papers nowadays comparing with these papers [22], [30], [32], [34] similar to our method. We learned from these papers, discovered some problems and improved them. Fig.10 shows the comparison of our method with other existing methods. In this paper, the boundary network is seen as part of the Trojan circuit. If the boundary network is ignored in detection, the entire HT circuit could not be detected completely. Therefore, detecting the boundary network is the key to detect HTs entirely. Different ratios of HT nets and normal nets in the netlist perform differently in HT detection. The ratio of the RS series is 1:5, the ratio of S series is 1:40 here. As mentioned before, the boundary nets belong to the fuzzy set in the netlist and are tough to detect. In RS series, a larger HT net ratio causes the larger ratio of boundary network, which can better train classifiers; while in the S series, the smaller HT net ratio causes the smaller ratio of boundary network, which makes it easier for classifiers to ignore the boundary network. Hence, it is challenging to detect the HT with a small ratio of boundary network.

From Fig.10, the reference [22], [30], [34] consider the boundary network in the RS series. Our method exceeds all the existing methods in the RS series except the TNR and Precision of [32]. So ML-HTCL compares with [32] separately and the result is shown in Fig.11. However, compared with [22], [30], [34], if we look at the S series, our effect is not good enough. It is because [22], [30], [34] do not consider the boundary network in the S series, but we do. This phenomenon reveals that the abandonment of the boundary network would make a better performance on HT detection. Since the number of normal circuit nets is increased and the number of HT circuit nets is reduced, such results are expected. It is well known that the use of machine learning algorithms to deal with the two-category problem is unexplained, it may cause reduce the detection rate of the normal circuits while improving the detection rate of the HT circuits. Hence the complete detection of the normal circuit should be prioritized. Only gradually improve the detection efficiency of the hardware Trojan under the premise of ensuring the safety and integrity of the normal circuit is helpful for practical application; The partial detection of the Hardware Trojan helps the manual to continue the detection, but if the normal circuit is misidentified, the subsequent processing will destroy the original function of the circuit, which is contrary to the original intention of the Hardware Trojan detection.

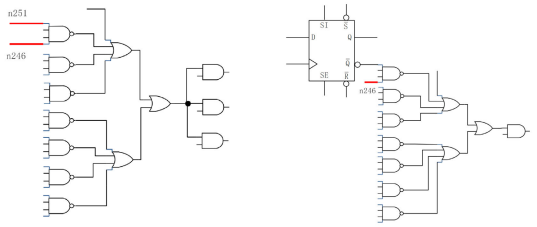
In summary, ML-HTCL is relatively excellent beyond others. By comparing the test sequence and the order of the



(a) Comparsion of TPR (b) Comparsion of TNR (c) Comparsion of Precision

(d) Comparsion of Recall (e) Comparsion of F-measure (f) Comparsion of Accuracy

FIGURE 11. In the case of joining the boundary net, TPR, Recall, F-measure, and Accuracy of ML-HTCL surpass the [32]. The reason that TNR and Precision do not exceed [32] may be that ML-HTCL is more inclined to a higher TPR value. Of course, increasing the value of TNR and Precision is one of our future jobs.



(a) F-RS232T1000-HT (b) F-RS232T1100-HT

FIGURE 12. Part location results.

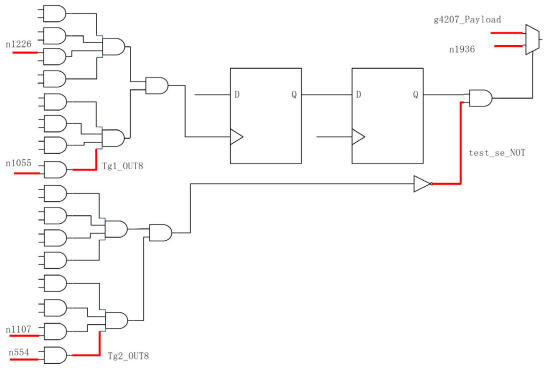


FIGURE 13. Final-s15850T100-HT.

output results, the basic locating of the HT can be achieved. It can accurately locate per detected net. The detected HT nets are marked in the netlist, which is shown in Fig.12, Fig.13, and Fig.14. All the nets in these figures are testing HT nets. The black nets indicate that it is the detected HT nets; the red nets indicate that it is not the detected HT nets. It is easy to find most of the HT nets in the netlist and their details, which can be helpful to process these HT nets further.

Attributed to the addition of the boundary network, the HTs and the normal circuits have clear boundaries. There are no obscure and difficult-to-handle circuit components in the HT detection process, which ensures the entire HT detection and locating easier.

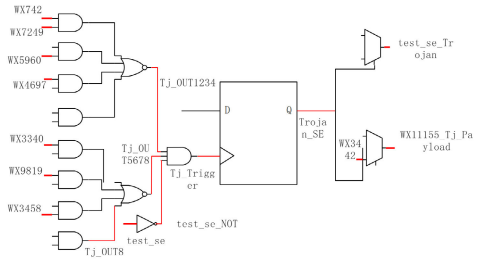


FIGURE 14. Final-s35932T100-HT.

Undoubtedly, the HTs locating results based on the proposed method flow is closely related to the quality of the detection results. The advantage of ML-HTCL is that each wire can be located accurately, which means all nets belonging to an HT can be found in the netlist. According to the gate-level netlist of a chip, after the detection of the machine learning algorithm, each specific net and the basic components of the circuit like logic gates where the HT circuit is located can be found in the netlist document which is suffixed with *.v*. The purpose of an accurate location is to find the name of the net of an HT (each net has a corresponding name in the netlist). If all the input and output nets are HT nets, then this component must be a Hardware Trojan; if only part of the input and output net is HT net, there are generally two possibilities: 1) This component belongs to the boundary area, 2) This part of the HT is not completely detected. This second problem will be greatly reduced when the detection effect continues to increase in the future.

TABLE 8. Similar method algorithm complexity comparison.

Algorithm	Time complexity
[22]-Random Forest	$O(N * M * D)$
[31]-Neural Network	$O(\sum_{m=1}^{k-1} N_m * N_{m+1})$
ML-HTCL-BP neural Network	$O(\sum_{m=1}^{k-1} N_m * N_{m+1})$
ML-HTCL-OneClass-SVM	$O((Nsv)^3 + L * (Nsv)^2 + d * L * (Nsv))$ to $O(d * L^2)$
[33]-Unknown	N/A

Table 8 compares the time complexity of the existing literature. Reference [22] uses a random forest algorithm with a time complexity of $O(N * M * D)$, N is the number of training samples, the feature dimension is M, and the depth of the tree is D; [30] uses a multi-layer neural network with a time complexity of $O(\sum_{m=1}^{k-1} N_m * N_{m+1})$. We only consider the time consumption of internal data transmission, N_m is the number of nodes in the m-th layer of the neural network; the multi-layer BP neural network in this paper has the same time complexity as [30]; the one-class SVM used in this paper does not find time complexity in the related literature, so we consider to the time complexity of a basic SVM [35], whose time complexity is between $O((Nsv)^3 + L * (Nsv)^2 + d * L * (Nsv))$ to $O(d * L^2)$, where Nsv is the number of support vectors, L is the number of training set samples, and d is the dimension of

each sample; [32] does not mention the algorithm used, so its time complexity cannot be judged.

Overall, the performance of ML-HTCL is quite satisfied. Because the locating result is directly proportional to the detection result, almost all the HTs in the RS series are located. Some HTn nets in the s series netlist can never be located, and these HT nets are almost all boundary networks. So the future work should be carried out on the processing of the boundary network. Locating the HTs beyond on detection only needs to backtrack the detected HT nets, so its time complexity is just $O(n)$.

VI. CONCLUSION

In this paper, the ML-HTCL method is proposed, which employs the different treatment methods for different series of netlists with a new theoretical basis. The experiment results show our method is better than most similar methods in the RS series in the same benchmarks. Besides, we first proposed the concept of positioning hardware Trojan in the gate-level netlist. In the future, our goal is to achieve 100% detection of normal circuits and further improve the detection rate of HT circuits. At the same time, we will turn to the process the circuit that has never been detected in past articles, and promote the friendly development of machine learning and HT detection in academia. Finally, based on the current locating results, our work is to further deal with HTs to better protect integrated circuits.

REFERENCES

[1] B. Liu and G. Qu, "VLSI supply chain security risks and mitigation techniques: A survey," *Integration*, vol. 55, pp. 438–448, Sep. 2016.

[2] 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, 2018.

[3] J. Francq and F. Frick, "Introduction to hardware Trojan detection methods," in *Proc. Design, Automat. Test Europe Conf. Exhib.* San Jose, CA, USA: EDA Consortium, 2015, pp. 770–775.

[4] D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi, and B. Sunar, "Trojan detection using IC fingerprinting," in *Proc. IEEE Symp. Secur. Privacy*, May 2008, pp. 296–310.

[5] O. A. Osanaiye, A. S. Alfa, and G. P. Hancke, "Denial of service defence for resource availability in wireless sensor networks," *IEEE Access*, vol. 6, pp. 6975–7004, 2018.

[6] R. S. Chakraborty, S. Narasimhan, and S. Bhunia, "Hardware Trojan: Threats and emerging solutions," in *Proc. IEEE Int. High Level Design Validation Test Workshop*, Nov. 2009, pp. 166–171.

[7] K. Xiao, D. Forte, Y. Jin, R. Karri, S. Bhunia, and M. Tehranipoor, "Hardware Trojans: Lessons learned after one decade of research," *ACM Trans. Des. Automat. Electron. Syst.*, vol. 22, no. 1, p. 6, 2016.

[8] M. Oya, Y. Shi, M. Yanagisawa, and N. Togawa, "A score-based classification method for identifying hardware-Trojans at gate-level netlists," in *Proc. Design, Automat. Test Eur. Conf. Exhib.*, 2015, pp. 465–470.

[9] M. Hicks, M. Finnicum, S. T. King, M. M. K. Martin, and J. M. Smith, "Overcoming an untrusted computing base: Detecting and removing malicious hardware automatically," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2010, pp. 159–172.

[10] J. Zhang, F. Yuan, L. Wei, Y. Liu, and Q. Xu, "VeriTrust: Verification for hardware trust," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 34, no. 7, pp. 1148–1161, Jul. 2015.

[11] A. Waksman, M. Suozzo, and S. Sethumadhavan, "FANCI: Identification of stealthy malicious logic using Boolean functional analysis," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2013, pp. 697–708.

[12] H. Xue and S. Ren, "Self-reference-based hardware Trojan detection," *IEEE Trans. Semicond. Manuf.*, vol. 31, no. 1, pp. 2–11, Feb. 2018.

[13] X. Chen, L. Wang, T. Wang, Y. Liu, and H. Yang, "A general framework for hardware Trojan detection in digital circuits by statistical learning algorithms," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 36, no. 10, pp. 1633–1646, Oct. 2017.

[14] A. Nejat, D. Hely, and V. Beroulle, "ESCALATION: Leveraging logic masking to facilitate path-delay-based hardware Trojan detection methods," *J. Hardw. Syst. Secur.*, vol. 2, no. 1, pp. 83–96, 2018.

[15] G. Chi, Z. Yu, Y. Zhou, and S. Lei, "A hardware Trojan detection method based on region segmentation technology," *Semicond. Technol.*, 2017.

[16] A. Amelian and S. E. Borujeni, "A side-channel analysis for hardware Trojan detection based on path delay measurement," *J. Circuits, Syst. Comput.*, vol. 27, no. 9, 2017, Art. no. 1850138.

[17] N. Mohankumar, M. Jayakumar, and M. N. Devi, *CRC-Based Hardware Trojan Detection for Improved Hardware Security*.

[18] M. Cozzi, J.-M. Galliere, and P. Maurine, "Thermal scans for detecting hardware Trojans," in *Proc. Int. Workshop Constructive Side-Channel Anal. Secure Design*. Cham, Switzerland: Springer, 2018.

[19] J. Zhong and J. Wang, "Thermal images based Hardware Trojan detection through differential temperature matrix," *Optik*, vol. 158, pp. 855–860, Apr. 2018.

[20] 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.

[21] 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.

[22] K. Hasegawa, M. Yanagisawa, and N. Togawa, "Trojan-feature extraction at gate-level netlists and its application to hardware-Trojan detection using random forest classifier," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2017, pp. 1–4.

[23] C. Bao, D. Forte, and A. Srivastava, "On reverse engineering-based hardware Trojan detection," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 35, no. 1, pp. 49–57, Jan. 2016.

[24] 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. 2017, pp. 1–6.

[25] C. Dong, G. He, X. Liu, Y. Yang, and W. Guo, "A multi-layer hardware Trojan protection framework for IoT chips," *IEEE Access*, vol. 7, pp. 23628–23639, 2019.

[26] A. Bazzazi, M. T. M. Shalmani, and A. M. A. Hemmatyar, "Hardware Trojan detection based on logical testing," *J. Electron. Test.*, vol. 33, no. 4, pp. 381–395, 2017.

[27] A. A. Nasr and M. Z. Abdulmageed, "Automatic feature selection of hardware layout: A step toward robust hardware Trojan detection," *J. Electron. Test.*, vol. 32, no. 3, pp. 357–367, 2016.

[28] 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*, vol. 35, no. 2, pp. 73–90, Apr. 2018.

[29] *Trust-HUB*. [Online]. Available: http://www.trust-hub.org

[30] K. Hasegawa, M. Yanagisawa, and N. Togawa, "Hardware Trojans classification for gate-level netlists using multi-layer neural networks," in *Proc. IEEE 23rd Int. Symp. On-Line Test. Robust Syst. Design (IOLTS)*, Jul. 2017, pp. 227–232.

[31] B. Schölkopf, R. Williamson, A. Smola, J. Shawe-Taylort, and J. Platt, "Support vector method for novelty detection," in *Proc. Adv. Neural Inf. Process. Syst.*, 2000, pp. 582–588.

[32] K. Hasegawa, M. Yanagisawa, and N. Togawa, "A hardware-Trojan classification method utilizing boundary net structures," in *Proc. IEEE Int. Conf. Consum. Electron.*, Jan. 2018, pp. 1–4.

[33] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Proc. IJCAI*, Montreal, QC, Canada, 1995, pp. 1137–1143.

[34] K. Hasegawa, M. Oya, M. Yanagisawa, and N. Togawa, "Hardware Trojans classification for gate-level netlists based on machine learning," in *Proc. IEEE 22nd Int. Symp. On-Line Test. Robust Syst. Design*, Jul. 2016, pp. 203–206.

[35] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining Knowl. Discovery*, vol. 2, no. 2, pp. 121–167, 1998.

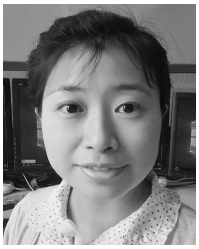
CHEN DONG received the B.S. and M.S. degrees from the College of Mathematics and Computer Science, Fuzhou University, China, in 2002 and 2005, respectively, and the Ph.D. degree in computer science from the Computer School, Wuhan University, China, in 2011. She was a Visiting Researcher with the University of California at Los Angeles, from 2015 to 2016. She is currently an Assistant Professor with the College of Mathematics and Computer Science, Fuzhou University. Her research interests include intelligent computing, and integrated circuit physical and security design.

FAN ZHANG received the B.E. degree in computer science from the University of Science and Technology of Hunan, China, in 2017. He is currently pursuing the degree with the College of Mathematics and Computer Science, Fuzhou University. His research interests include intelligent computing, and integrated circuit physical and security design.

XIMENG LIU received the B.Sc. degree in electronic engineering from Xidian University, Xi'an, China, in 2010, and the Ph.D. degree in cryptography from Xidian University, China, in 2015. 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 papers on the topics of cloud security and big data security, including papers in the IEEE TRANSACTIONS ON COMPUTERS, the IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, the IEEE TRANSACTIONS ON SERVICE 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, Qishan Scholars in Fuzhou University, and ACM SIGSAC China Rising Star Award, in 2018. He served as 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 served as the Lead Guest Editor for Wireless Communications and Mobile Computing.

XING HUANG received the B.S. degree in computer science and technology and the Ph.D. degree in electronic science and technology from Fuzhou University, Fuzhou, China, in 2013 and 2018, respectively. He was a joint Ph.D. Student with the Department of Electrical and Computer Engineering, Duke University, USA, supported by the Chinese Scholarship Council. He is currently a Postdoctoral Research Fellow with the Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan. His current research interests include design automation for microfluidic biochips, and integrated circuits and chip security.

WENZHONG GUO 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, where he is currently a Full Professor with the College of Mathematics and Computer Science. He also leads the Network Computing and Intelligent Information Processing Lab, which is a key Laboratory of Fujian Province, China. His research interests include cloud computing, mobile computing, and evolutionary computation.



YANG YANG received the B.Sc. and Ph.D. degrees from Xidian University, Xi'an, China, in 2006 and 2011, respectively. She is an Associate Professor with the College of Mathematics and Computer Science, Fuzhou University. She is also a Research Fellow with Singapore Management University, under the supervision of Robert H. Deng. She has published more than 60 papers in the IEEE TIFS, IEEE TDSC, IEEE TSC, IEEE TCC, and IEEE TII. Her research interests include information security and privacy protection.