Genetic Algorithm for Hardware Trojan Detection with Ring Oscillator Network (RON)

Nima Karimian, Fatemeh Tehranipoor, Md. Tauhidur Rahman, Shane Kelly and Domenic Forte
Dept. of Electrical & Computer Engineering
University of Connecticut
{nima, f.tehrani, tauhid, spk09001, forte}@engr.uconn.edu

***Abstract*—Securing integrated circuits against malicious modifications (i.e., hardware Trojans) is of utmost importance, as hardware Trojans may leak information and reduce reliability of electronic systems in critical applications. In this paper, we use ring oscillators (ROs) to gather measurements of ICs that may contain hardware Trojans. To distinguish between Trojan-inserted ICs and Trojan-free ICs, we investigate several classification approaches. Furthermore, we propose a novel feature selection approach based on the Genetic Algorithm (GA) and evaluate its performance compared to several popular alternatives. The proposed method is an improvement over principal component analysis (PCA) in terms of accuracy and equal error rate by 30% and 97% respectively.**

***Index Terms*—Hardware Trojan Detection, Ring Oscillator Network, One class classification, Genetic Algorithm**

I. Introduction

In recent years, semiconductor companies have increasingly outsourced their fabrication of integrated circuits (ICs) to companies offshore. Since the IP owner cannot be present during fabrication, this makes IC designs increasingly vulnerable to malicious modifications referred as hardware Trojan horses (HTHs). A hardware Trojan is an alteration of the original integrated circuit (IC) design by an attacker in order to access and manipulate information stored or disable processing on the chip [1], [2].

Hardware Trojans are a realistic threat and should be considered as a serious concern to modern ICs. As a result, hardware Trojan detection has become an important research topic and gained significant attention in recent years. There are various hardware Trojan detection techniques that have been proposed so far. Most of them are nondestructive and based on either logic testing or side channel analysis. Side-channel analysis entails monitoring of operational parameters of the circuit, such as leakage current, dynamic power due to switching activity, and path delays [3]–[8]. The suspect ICs are considered as Trojan-free if they match the expected side channels compared to a golden model. In logic testing approaches, test vectors are applied in the hopes of activating a Trojan and detecting its effects on IC output [6]. Destructive approaches that require reverse engineering of ICs have also been proposed such as in [9]. The drawback of destructive approaches is that they destroy the ICs they are applied to and, therefore, cannot be applied to all suspect ICs.

Side channel-based approaches are the most widely investigated, but suffer from various shortcomings. Most notably, it is difficult to obtain fine-grained measurements of power, delay, etc. so that small Trojans can be detected, especially in the presence of large process variations. To address this issue, in [10], an on-chip structure called the ring oscillator network (RON) was proposed to improve Trojan detection. The frequency of each ring oscillator (RO) in RON is sensitive to power fluctuations caused by circuit switching activity as well as the switching activity of the Trojan. In a RON, the ROs can detect the increase in transient power consumption induced by Trojan circuits in different areas of the IC under authentication. Still, even the detection of all Trojan-inserted ICs by RON has not been obtained in previous work.

In this paper, we investigate different machine learning approaches for detecting hardware Trojans based on measurements gathered by RON. We shall assume that golden ICs are available to act as a baseline for learning. Our main contributions are as follows:

1) We discuss prior approaches based on Principal Component Analysis (PCA) and Support Vector Machines as well as their limitations.
2) We propose a new approach that utilizes the Genetic Algorithm (GA) to select the best ring oscillator (RO) measurements from RON. We use these optimal features and one-class SVM to distinguish between Trojan-free and Trojan-inserted ICs.
3) We perform experiments using real silicon data (33 ICs fabricated via MOSIS) and compare three machine learning approaches (PCA+convex hull, SVM, and GA+SVM). We use several metrics to measure the performance of each approach including receiver operating characteristic (ROC) curves which can show the trade-off between sensitivity and specificity. The proposed GA+SVM approach required the largest amount of training time, but could correctly classify Trojan inserted ICs with 99.6% accuracy. This was a significant improvement compared to the more commonly used PCA which at best achieved 76.8% accuracy.

The rest of the paper is organized as follows: Section II gives a brief review of the ring oscillator network (RON). Section III describes some background information about classification problems and algorithms. Section IV explains our proposed approach that combines the Genetic Algorithm with Support Vector Machines (SVM). Experiments and results are discussed in Section V. Finally, Section VI concludes the paper.

II. Ring Oscillator Network (RON)

Prior work has shown that a correlation exists between the switching activity caused by Trojans and ring oscillator (RO) activity. For example, if an RO is placed near a Trojan's trigger, it will have a lower frequency than expected. Several attempts have been made in prior work to exploit the sensitivity of ROs to Trojans for Trojan detection. In [11], the authors claimed that Trojans can be detected by observing the changes in the frequency of the ROs as well. They provided an algorithm and applied it to an FPGA unit and then measured the frequencies of the ROs to detect any malicious change to the design. Even though this method is applicable to both ASIC and FPGA, it requires reconfiguration

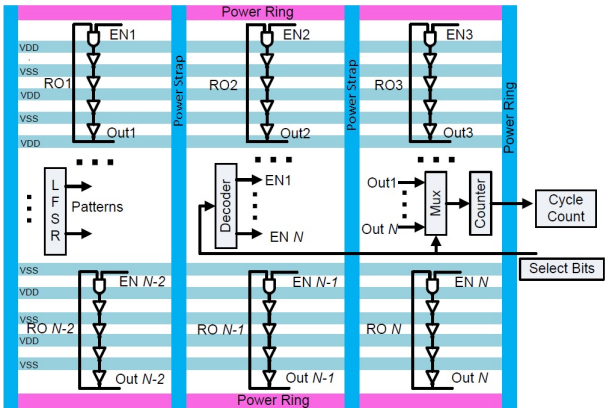

Figure 1. The ring oscillator network (RON) structure

of functional paths into ROs which has high overhead such as power consumption.

[10], [12] proposed an on-chip ring oscillator network (RON) structure for ASICs to detect hardware Trojans by exploiting ring oscillators (ROs) as sensors for power network noise. Figure 1 shows several ROs (RO_1 to RO_N) in RON. The number of the ROs depends on the area of the chip, the area needed to implement the RO network, and the power structure of the chip. Each stage of the RO is vertically placed within a different power/ground line of the IC. If a Trojan is activated, it creates a voltage drop in the power lines that will impact the RO frequencies. Thus, by measuring changes in the frequency, the malicious addition, deletion or alteration may be detected. The drawback thus far of this approach of paper [10] is that accuracy of detection for small Trojans is less than 50% and with false positives of approximately 30%.

In this paper, our goal is to improve upon RON's ability to distinguish between Trojan-free and Trojan-inserted ICs by developing better classification algorithms.

III. BACKGROUND

This section discusses classification problems and algorithms. We highlight the popular approaches for Trojan detection used in prior work and their limitations.

A. Classification

There are three different types of classification problems: multiclass, binary, and one class classification. In multiclass, the goal is to develop a classification rule that uses the "features" of an unknown object to determine its correct "class" among more than two possible classes. Binary classification has a similar goal except there are only two possible classes for each object. In one-class classification, there are two or more classes, but the goal is to correctly distinguish the objects in one "target" class from all other "outlier" classes. The data (features) available in each type of classification problem also differs. In multiclass and binary classification, data from all classes is assumed to be available in order to develop the classification rule. On the other hand, only the data related to the target class is available in one-class classification problems.

The problem of detecting ICs with hardware Trojans (i.e., Trojan-inserted) among ICs without hardware Trojans (i.e., Trojan-free) should be considered as an instance of one-class classification. Under many scenarios, one can assume that data from golden (i.e., known Trojan-free) ICs is available. However, the data from Trojan-inserted chips is impossible to know since the possibilities for Trojans are countless. In the subsections below, we describe two algorithms that have been used in prior work to create classification rules that separate Trojan-free from Trojan-inserted ICs: Principal Component Analysis (PCA) with convex hull and one-class Support Vector Machine (SVM). We also discuss their limitations.

B. Principal Component Analysis (PCA)

Dimensionality reduction is a process that is often used as a pre-processing step for classification. One very widely used technique is called principal component analysis (PCA). The purpose of PCA is to reduce the number of variables (number of features) of a data-set by orthogonal transformation of the original data into a new coordinate system. The new coordinate system is one where the greatest amount of variation in the data-set is captured by the first axis (first principal component), followed by the second (next principal component), and so forth. This is often considered as useful for distinguishing objects of different classes and can allow one to use only the first several principal components as classification features rather than all the data. More details for PCA are discussed in [13].

In prior work, PCA has been applied to the Trojan problem as follows (e.g., [14], [6]). The dataset to which PCA is applied contains only data from Trojan-free (golden) ICs. This data-set is transformed and only the first several principal components (typically 2 or 3) are used. A convex hull (boundary) is created around this data in the new space. Given data (features) of an IC under test, it is transformed using the above result into the new space. If the transformed data lies within the convex hull, it is assumed that the IC under test is Trojan-free.

Limitations: PCA is not really well suited for one-class problems. PCA is performed only with data from the target class (Trojan-free ICs) so the new space only gives maximum variance within the target class. While this can help distinguish between ICs in the target class, it may not help with the outlying class (Trojan-inserted ICs). This seems to be supported by analysis given in [15] which proves that the error on the outliers mainly depends on the eigenvalues of the estimated target covariance matrix. Thus the principal components with highest variance determined by PCA as features may give sub-optimal performance. This could explain why the classification accuracy in [10] is quite low.

C. Support Vector Machine (SVM)

Support vector machines (SVMs) have become one of the most important and widely-used classification techniques [9]. In SVM classification, an optimal separating hyperplane is generated that results in a maximum margin between two classes of data. For data that cannot be separated by a hyperplane, SVM can utilize a nonlinear mapping by applying a kernel function. Then, it constructs a hypersphere see Figure 2 between the two categories of data in the higher (kernel mapped) feature space. Data vectors which are touching the hypersphere are called support vectors (SVs) and contain all the information required for the classification rule. The hypersphere has a radius R and center denoted by b. Note that in one-class problems, only data from the target class is available so the hypersphere is constructed around the target class. The goal is to find the smallest possible hypersphere which contains the target data-set. The main details and formulation of the ν-SVM, an approach that adapts SVM to one-class problems, are discussed below. Given a training set $\bar{x_i} \in R^N, i = 1, \ldots, l$, we want to obtain a hypersphere from one class classification. To start ν-SVC, [16] gives some details on a model which gives a closed boundary around the data-set by hypersphere. The hypersphere is defined by center b

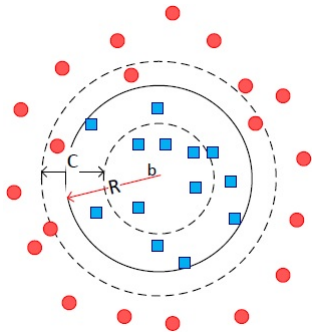

Figure 2. SVM example where circles and squares are unknown (outlier) and known(target) data respectively and the sphere with solid line represents dividing boundary. In ν-SVM, the C parameter can modify the boundary to include more or less outliers (both shown as dotted boundaries)

and a radius $R > 0$. Minimizing the size of the hypersphere is equivalent to minimizing R^2 as shown in the following quadratic programming problem:

$$\min_{R,b,\xi} \quad R^2 + C\sum_{i=1}^{l}\xi_i \tag{1}$$

$$Subject\ to, \quad ||\phi(\bar{x_i}) - b|| \leq R^2 + \xi_i \tag{2}$$
$$\xi_i \geq 0, i = 1,\ldots,l. \quad \forall_i$$

In the equation above, R and b are parameters determined by solving the above problem and represent the hypersphere. ξ_i are "slack" variables that represent the amount of misclassified objects (training errors). The parameter C is for controlling the trade-off between the mass of the sphere and the errors (outliers allowed in the sphere). C can be interpreted as the margin of the hypersphere used to separate the data (see Figure 2). As we have target (known) data in one class classification, the support vectors of the boundary only comes from the target. As can be seen in Figure 2, the blue points and the red points are target and outlier data respectively. By decreasing C, some of the blue points may lie outside the hypersphere and causes the error of classifying the target class to increase. On the other hand, by increasing C, some of the red points may enter the hypersphere and causes the error of classifying outliers to increase. It can be challenging to find the optimal C. The distance from $\bar{x_i}$ to the center b (see Figure 2) should not be strictly smaller than R^2 to allow the possibility of outliers in the training set. In fact, larger distances should be penalized through equation 1.

The sphere does not always give the optimal result. As discussed above, the data may not be separable by a sphere. Even if the data is separable, there also may be a large amount of empty space within the sphere where there is no data from the target class. To improve this, kernel functions have been used to project the data into higher dimensions where the target data is more easily contained by a hypersphere. For example, the Gaussian kernel is given by:

$$k(\bar{x_i}, \bar{y_i}) = exp(-||\bar{x_i} - \bar{y_i}||^2/\sigma^2) \tag{3}$$

Once the hypersphere parameters are determined, any unknown object with features denoted by D^* can be classified by the following decision rule. D^* is classified as part of the target class when:

$$\Sigma\alpha_i exp(-||D^* - \bar{x_i}||^2/\sigma^2) \geq -R^2/2 + C \tag{4}$$

where C depends only on the Support Vectors $\bar{x_i}$. This equation means that any unknown data which is contained within the hypersphere will be classified as part of the target class. Otherwise, it will be classified as part of the outlier class.

Advantages and Limitations: SVM is very robust to noise and can be used to separate nonlinear data via kernel functions. One limitation of using one class SVM is that if the number of features are greater than the number of samples, then one class SVM is likely to give poor performances. A large number of features and samples can also result in large training time (i.e., time needed to find the hyperplane or hypershere). While support vectors (SVs) of the hypersphere only comes from the target data-set, the ability to tune C and use kernel functions can result in better accuracy than PCA.

IV. Proposed Approach Based on Genetic Algorithm

In this section, we discuss our proposed approach which applies the Genetic Algorithm (GA) for feature selection and uses one class SVM for classification.

A. Genetic Algorithm (GA)

Genetic Algorithms (GAs) are exploration algorithms based on the theory of natural selection (i.e., "survival of the fittest"). GA moves from one population of chromosomes to a new population by using genetics inspired operators: crossover, mutation, and roulette wheel. Each chromosome is represented as a binary string. The population of chromosomes evolves over time. The population of chromosomes that survive (i.e., become part of the next population) are those that are the most fit [17].

This same concept can be applied towards feature selection for classification problems where the goal is to find a small subset of variables from a data-set that gives the best classification accuracy. A simple GA is shown in Algorithm 1. The algorithm begins from a population of randomly selected features and then iterates into the next generations (a new population). In each generation, the fitness of every set of features in the population is evaluated by a fitness function (e.g., classification accuracy). Then, multiple feature sets are selected from the current population (those with the largest fitness) and modified to form a new population using the genetic operators (crossover, mutation, etc.). The new population is then used in the next iteration of the algorithm and so forth. The algorithm terminates when some stopping criteria is reached (e.g., certain number of iterations or desired classification accuracy reached).

In Figure 3, 8 different genes are illustrated in a chromosome (selected features or feature set). Here, the presence of a '0' ('1') denotes that the feature shall be ignored (used) by the classification algorithm respectively. For generating a new population, GA uses mutation and crossover operators. For mutation, one or more genes (selected features) are inverted. As shown in Figure 3, two random genes from chromosome (feature set)shown with red color are selected and after mutation, genes which were '0' and '1' became '1' and '0', respectively. In other words, mutated features that were previously ignored become used for classification and vice versa.

The crossover mechanism on the other hand mixes up genes from existing solutions. Two members of the population are randomly chosen to act as "parents". A simple "crossing point" is also chosen randomly. A child (for the next generation) is created by mixing genes from both parents. Genes (features selected) from the first parent are copied from one side of the crossing point and the remaining features are taken from the other parent and other side of the crossing point. Note that

Randomly generate the first chromosome population P
while Number of iterations or desired classification accuracy is not satisfied **do**
  Evaluate all chromosomes in P with a fitness function
  Select N_c parents for crossover based on the fitness function and randomly select crossover points
  Select N_m parents for mutation based on the fitness function and features to mutate
  Perform crossover and mutation
  $Offspring = N_c + N_m$ {New generation of offspring}
  Evaluate offspring with fitness function
  Sort all the chromosomes based on the fitness function and select the population P to pass to the next iteration
end while
$Output = optimal features$ {Select the best chromosome from the last generation of population based on fitness function}

Algorithm 1: General genetic algorithm for feature selection

Figure 3. Eight different genes in chromosome and example of mutation

more complex schemes are possible with multiple crossing points as well. Figure 4 shows an example assuming a single point crossover. There are various parameters one can consider for GA such as population size, percentage of population to mutate, and percentage of population to crossover, etc. The choice of parameters can impact the solution (features selected) and the time required to converge to a solution.
Advantages and Limitations: In general, the main advantage of the GA for feature selection is its simplicity. For classification, it does not require too much mathematics and can find multiple near-optimal solutions (sets of features). On the other hand, the processing required by GA is often very demanding. It may take several hours or days to find the optimal features. However, once the optimal features are selected, classification can often be performed very quickly. In most cases, classification time is of greater concern than training time.

B. Proposed Approach

Our approach for classifying ICs as Trojan-inserted and Trojan-free is described in this section. Our assumptions are as follows. We are given a data-set of features from Trojan-free ICs and must use them to train a classifier that selects the correct class of an unknown IC (Trojan-free or Trojan-inserted). This is a basic instance of one-class classification with the target class as the Trojan-free class and the outlier class as the Trojan-inserted class. Features in the data set are side-channel measurements from ICs. For example, the features in this paper are frequencies of ring oscillators (ROs) obtained from RON. However, other features could be used as well. Our goal is to select an optimal set of features to distinguish between Trojan free and Trojan inserted ICs and find a good classification rule.

It is very important to select a minimum number of features needed in the classification process. This is due to the fact that performance of the classifier is sensitive to the choice of the features, since some of the features have high correlation, large amount of noise, etc. that are not useful and could be harmful for the classifier. Our approach relies on the Genetic Algorithm (GA) for feature selection rather than PCA. It utilizes SVM in order to determine a decision boundary. Since our proposed

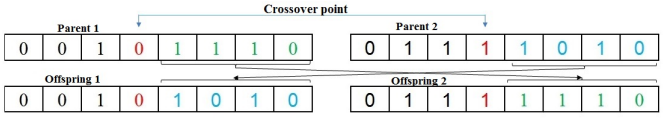

Figure 4. Example of crossover

Input: Golden model (Trojan free) IC data S_{TR} and algorithm parameters for GA and SVM
Generate the first chromosome population P randomly
while Number of iterations or desired classification accuracy is not satisfied **do**
  for $i = 1$ **to** P **do**
    $S_{pop} \Leftarrow S_{TR}/2$ {Randomly}
    $S_{fit} \Leftarrow S_{TR}/2$ {Randomly}
    Determine hypersphere via SVM for S_{pop} and all chromosomes (feature sets) in P
    Calculate fitness function using SVM output **and** S_{fit}
    $FF_i = EER(P_i)$ {Compute the fitness FF for each chromosome (feature set) in P}
  end for
  Select N_c parents for crossover based on fitness of P and randomly select crossover points
  Select N_m parents for mutation based on fitness of P and randomly select features to mutate
  $Offspring = N_c + N_m$ {Generate offspring}
  Determine hypersphere via SVM for S_{pop} and all chromosomes (feature sets) in $Offspring$
  Calculate fitness function using SVM output **and** S_{fit}
  $FF_i = EER(P_i)$ {Compute the fitness FF for each chromosome (feature set) in $Offspring$}
  Sort all the chromosomes based on the fitness function for P and $Offspring$ and select the new population P to pass to the next iteration
end while
$Output = optimal features$ {Select the best chromosomes from the last generation of population based on fitness function}

Algorithm 2: Calculating fitness function

approach combines the genetic algorithm with SVM, we shall refer to it as GA+SVM for the remainder of the paper.

Our approach is illustrated in Algorithm 2. We divide our dataset of Trojan-free ICs into two categories:

(i) Samples for training(S_{pop}): Half of the samples are selected for SVM training. In other words, they are used to compute an SVM boundary based on features selected in an iteration of the GA.

(ii) Samples for evaluating fitness(S_{fit}): The other half of the samples are selected to evaluate the fitness of the features selected by the GA.

As can be seen in Algorithm 2, we apply SVM to S_{pop} to generate hyperspheres for all chromosomes (feature sets). Then, we evaluate the fitness of each chromosome using the samples in S_{fit}. The fitness function in our case is the accuracy associated with classifying the S_{fit} samples using the hypersphere. Next, we apply mutation and crossover based on the best fitness function to generate new offspring (which may result in improved accuracy). We apply the same sequence of steps (SVM and fitness evaluation) for the chromosomes (feature sets) defined by the offspring. The new population is determined by the most fit among P and $Offspring$. The process is repeated with the new population until a stopping

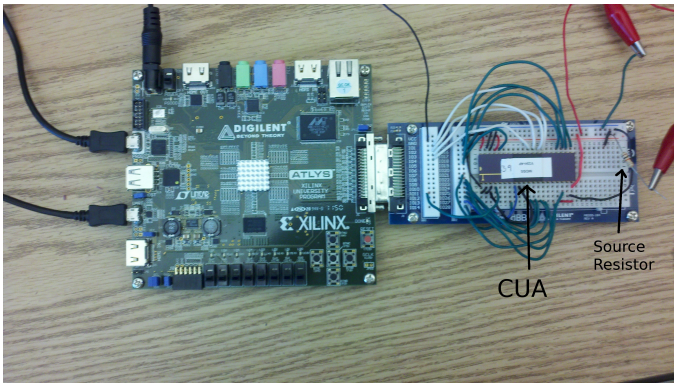

Figure 5. Test setup with Xilinx Spartan 6 FPGA (left) and the circuit under authentication (CUA) is connected to the FPGA through a serial connector (right).

criteria is met.

At the end of algorithm, the best features are selected as those from the chromosome (feature sets) with the highest fitness from the last population.

V. Experimental Setup

A. Data Collection

We analyzed the effectiveness of three different classification approaches algorithms on real silicon data from ring oscillator network (RON). 33 test chips were fabricated through MOSIS with IBM 90nm technology. The RON consists of 8 ROs in each test chip (RO_1 to RO_8). Each chip also contains 8 sequential and 15 combinational hardware Trojans, which can be turned on and off when gathering measurements. More details on the Trojans can be found in [10].

Data was collected from each chip with all Trojans turned off and each individual Trojan turned on. The measurements in the former case represent Trojan-free data and the latter measurements represent different Trojan-inserted cases. The chips were mounted on a prototyping board wired to a serial connector. The serial connector served as an interface between Xilinx Spartan-6 FPGA and Digilent Atlys board (see Figure 5). The programmed FPGA controlled the test sequence supplied to the IC and transmitted the measurements of the chip to a PC for analysis.

We applied three different classification approaches:

- **PCA+convex hull**: This is the approach described in Section III-B. We normalized the data before applying PCA which is a very common step. Three principal components were chosen from PCA as the features to create a convex hull.
- **SVM**: We applied the one class SVM approach described in Section III-C. In this paper we used a Gaussian kernel with $\sigma = 1$ and selected $C = 0.1$. For details on these parameters, please see Section III-C.
- **GA+SVM** This is the proposed approach described in the previous section which used the genetic algorithm (GA) for feature selection and SVM for fitness/classification. The detailed parameter setting for GA is as follows: population size (N_{POP}) is 20, mutation percentage (p_m) is 0.3, crossover percentage (p_c) is 0.8, mutation rate is 0.02, and number of iterations is 10. In addition the number of offspring (N_c) and number of mutants (N_m) are calculated by $N_c = 2 * ceil(p_c * N_{POP}))$ and $N_m = ceil(p_m * N_{POP})$ respectively. Note that $ceil(x)$ rounds the value of x up to the nearest integer. The cost fitness function of GA is the equal error rate (EER). EER refers to the trade-off between the two types of errors with one class (target and outlier). More details for EER are discussed below. The SVM parameters for GA+SVM are the same as above for SVM.

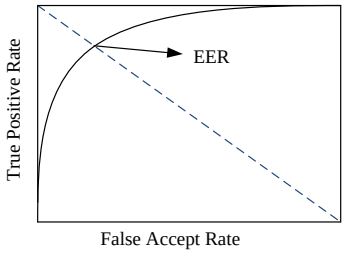

Figure 7. Example of ROC curve with EER.

B. Metrics for Comparison

To evaluate the performance of each classification algorithm, we calculated receiver operation characteristic (ROC) curves and accuracy. A *ROC* curve is illustrated in Figure 7. This curve shows the trade-off between false acceptance rate (FAR) and true positive rate (TPR). FAR refers to the rate at which a classifier incorrectly matches the unknown data (outlier) to the target class. TPR refers to the rate that a classifier correctly matches the known data (target) to the target class. Equal error rate (EER) is the location on the ROC curve where the FAR and 1-TPR are equal. The ideal ROC is one that looks like a step function. In that case, the FAR and TPR are always zero and one respectively, meaning there are no classification errors. This corresponds to an equal error rate (EER) of 0%. The worst possible ROC is one where the EER is 50%.

We define *accuracy* in terms of the number of samples correctly and incorrectly classified by a classification algorithm:

$$Accuracy = (TP+TN)/(TP+FN+TN+FP) \qquad (5)$$

where TP, TN, FN, and FP represent number of true positives, true negatives, false negatives, and false positives respectively.

C. Classification Results and Discussion

We split the chips into two groups: N_g chips representing the golden ICs (Trojan-free) and $33 - N_g$ chips representing the Trojan-inserted ICs. The experiments were conducted for 10 different trials. In addition, a different number of samples (N_g =8, 16, and 24 chips) were used for training to capture the trade-off between accuracy and number of training samples. The accuracy of classification was determined by comparing the classifier label that shows Trojan-free and Trojan-inserted in the chips sets. The percentage accuracy was computed by averaging the results of the 10 random trials.

The accuracy and EER for all approaches are shown in Table I. PCA exhibits the worst accuracy and EER for all cases. At best, PCA obtains an accuracy of 76.8%. As discussed earlier, PCA is not well suited for one-class problems because its choice of features does not capture the variance in the outlying class, only the target class. On the other hand, SVM and the proposed approach (GA+SVM) perform much better. The best accuracy and EER achieved are by GA+SVM. GA tries to find the best features among others. In our work, GA could find the optimal ROs among 8 ring oscillators. As some of the ROs may not be useful for classification problem (serving as added noise), GA has the ability to ignore them. This makes GA have

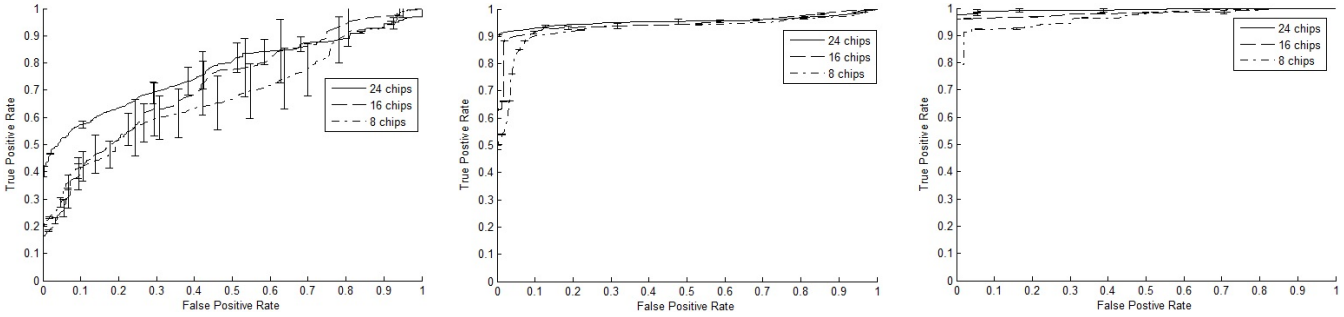

Figure 6. Performance of a classification based on Receiver operating characteristic (ROC) curve for (a) PCA, (b) SVM, and (c) GA+SVM.

Table II
RESULT FOR AMOUNT OF TIME REQUIRE FOR TRAINING IN EACH APPROACH.

N_g	PCA	SVM	GA+SVM
8	1.24 sec	1.31 sec	189.89 sec
16	1.32 sec	1.81 sec	221.54 sec
24	1.48 sec	2.20 sec	284.24 sec

the best accuracy and EER. In the best case (for 24 training samples), CA+SVM achieves 99.6% accuracy. Note that the accuracy and EER improve for all classification algorithms as the number of training samples increases. This makes intuitive sense.

Table I
PERCENTAGE ACCURACY AND EQUAL ERROR RATE FOR DIFFERENT APPROACHES IN CLASSIFICATION.

	PCA		SVM		GA+SVM	
N_g	Accuracy	EER	Accuracy	EER	Accuracy	EER
8	66.5%	37.1%	92.7%	10.1%	96.9%	7.4%
16	70.1%	33.4%	94.6%	9.2%	97.8%	6.9%
24	6.8%	28.8%	95.5%	7.7%	99.6%	0.8%

The ROC curves for each classification approach are shown in Figure 6. Error bars have also been added to the ROCs to show the variance across different trials. The ROC curves for PCA classification methods are far from the ideal ROC curve and have very large variance. The large variance indicates that the approach is very sensitive to the training samples used. The ROC curves for SVM and GA+SVM are much better. GA+SVM is very close to ideal (step function) for 24 samples. Both exhibit much smaller variance than PCA as well showing their robustness to the choice of training samples. Once again, for all algorithms the ROC curve improves as the number of training samples increases.

To summarize the classification performance, the proposed approach (GA+SVM) obtains the best performance overall. It represents a significant improvement over PCA which has been the most commonly used approach for Trojan detection thus far in the literature ([6], [14]).

We also recorded the time required to train a classifier using each approach. The times are summarized in Table II. The time to train is shortest with PCA followed by SVM then GA+SVM. GA+SVM requires the most time since it performs many iterations of SVM using different sets of features. While the GA+SVM has the worst trade-off with training time, all the approaches required about the same amount of time to classify an unknown IC with its classifier. Typically, the classification time is much more important than training since training only needs to be performed once. Classification must be performed for every suspect IC to determine if it has a Trojan or not. In other words, the time required to train GA+SVM is well spent because GA+SVM classifies the ICs with much better accuracy than the others (over 99% accuracy in the experiments above).

VI. CONCLUSIONS

In this paper, we presented a novel hardware Trojan detection approach that combined the genetic algorithm and support vector machines. The results demonstrated that the proposed approach is quite promising. The proposed approach showed significantly better performance than PCA which has been the most widely used approach for Trojan detection in prior work. In future work, we will investigate how sensitive the proposed approach is to changes in GA and SVM parameters. We will also investigate ways to remove the need for golden (known Trojan-free) samples.

REFERENCES

[1] M. Tehranipoor and F. Koushanfar, "A survey of hardware trojan taxonomy and detection," *Design Test of Computers, IEEE*, vol. 27, no. 1, pp. 10–25, Jan 2010.

[2] R. Karri, J. Rajendran, K. Rosenfeld, and M. Tehranipoor, "Trustworthy hardware: Identifying and classifying hardware trojans," *Computer*, vol. 43, no. 10, pp. 39–46, Oct 2010.

[3] M. Potkonjak, A. Nahapetian, M. Nelson, and T. Massey, "Hardware trojan horse detection using gate-level characterization," in *DAC*, July 2009, pp. 688–693.

[4] F. Koushanfar and A. Mirhoseini, "A unified framework for multimodal submodular integrated circuits trojan detection," *IEEE Transactions*, vol. 6, no. 1, pp. 162–174, March 2011.

[5] R. Rad, J. Plusquellic, and M. Tehranipoor, "Sensitivity analysis to hardware trojans using power supply transient signals," ser. HST. IEEE Computer Society, 2008, pp. 3–7.

[6] Y. Jin and Y. Makris, "Hardware trojan detection using path delay fingerprint," in *HOST*, June 2008, pp. 51–57.

[7] K. Xiao, X. Zhang, and M. Tehranipoor, "A clock sweeping technique for detecting hardware trojans impacting circuits delay," *Design Test, IEEE*, vol. 30, no. 2, pp. 26–34, April 2013.

[8] I. Exurville, J. Fournier, J.-M. Dutertre, B. Robisson, and A. Tria, "Practical measurements of data path delays for ip authentication amp; integrity verification," July 2013, pp. 1–6.

[9] C. Bao, D. Forte, and A. Srivastava, "On application of one-class svm to reverse engineering-based hardware trojan detection," in *(ISQED)*. IEEE, 2014, pp. 47–54.

[10] A. Ferraiuolo, X. Zhang, and M. Tehranipoor, "Experimental analysis of a ring oscillator network for hardware trojan detection in a 90nm asic," in *(ICCAD), IEEE/ACM*, Nov 2012, pp. 37–42.

[11] J. Rajendran, V. Jyothi, O. Sinanoglu, and R. Karri, "Design and analysis of ring oscillator based design-for-trust technique," May 2011, pp. 105–110.

[12] X. Zhang and M. Tehranipoor, "Ron: An on-chip ring oscillator network for hardware trojan detection," in *DATE*, March 2011, pp. 1–6.

[13] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*. John Wiley & Sons,, 1999.

[14] K. Hu, A. N. Nowroz, S. Reda, and F. Koushanfar, "High-sensitivity hardware trojan detection using multimodal characterization," in *DATE*, March 2013, pp. 1271–1276.

[15] D. M. Tax and K.-R. Müller, "Feature extraction for one-class classification," in *ICANN/ICONIP*. Springer, 2003, pp. 342–349.

[16] ——, "Feature extraction for one-class classification," in *ICANN/ICONIP*. Springer, 2003, pp. 342–349.

[17] C. To and M. Elati, "A parallel genetic programming for single class classification." ACM, 2013, pp. 1579–1586.