Machine Learning Techniques for Hardware Trojan Detection with Ring Oscillator Network

by

Kyle Allen Worley

An Honors Capstone

submitted in partial fulfillment of the requirements

for the Honors Diploma

to

The Honors College

of

The University of Alabama in Huntsville

2019-04-24

Honors Capstone Director: Dr. Tauhidur Rahman

Assistant Professor, Electrical and Computer Engineering Department

Student (signature)	Date 2019-04-24
Director (signature)	Date 2019-04-25
Department Chair (signature)	Date 4/25/19
Honors College Dean (signature)	Date 4/25/19

HONORS COLLEGE
THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Honors College
Frank Franz Hall
+1 (256) 824-6450 (voice)
+1 (256) 824-7339 (fax)
honors@uah.edu

Honors Thesis Copyright Permission

This form must be signed by the student and submitted as a bound part of the thesis.

In presenting this thesis in partial fulfillment of the requirements for Honors Diploma or Certificate from The University of Alabama in Huntsville, I agree that the Library of this University shall make it freely available for inspection. I further agree that permission for extensive copying for scholarly purposes may be granted by my advisor or, in his/her absence, by the Chair of the Department, Director of the Program, or the Dean of the Honors College. It is also understood that due recognition shall be given to me and to The University of Alabama in Huntsville in any scholarly use which may be made of any material in this thesis.

Kyle Worley

Student Name (printed)

Student Signature

2019-04-24

Date

Machine Learning Techniques for Hardware Trojan Detection with Ring Oscillator Network

by

Kyle Allen Worley

An Honors Capstone

submitted in partial fulfillment of the requirements

for the Honors Diploma

to

The Honors College

of

The University of Alabama in Huntsville

2019-04-24

Honors Capstone Director: Dr. Tauhidur Rahman

Assistant Professor, Electrical and Computer Engineering Department

_____

Student (signature) Date

_____

Director (signature) Date

_____

Department Chair (signature) Date

_____

Honors College Dean (signature) Date

Honors College
Frank Franz Hall
+1 (256) 824-6450 (voice)
+1 (256) 824-7339 (fax)
honors@uah.edu

Honors Thesis Copyright Permission

This form must be signed by the student and submitted as a bound part of the thesis.

In presenting this thesis in partial fulfillment of the requirements for Honors Diploma or Certificate from The University of Alabama in Huntsville, I agree that the Library of this University shall make it freely available for inspection. I further agree that permission for extensive copying for scholarly purposes may be granted by my advisor or, in his/her absence, by the Chair of the Department, Director of the Program, or the Dean of the Honors College. It is also understood that due recognition shall be given to me and to The University of Alabama in Huntsville in any scholarly use which may be made of any material in this thesis.

_____

Student Name (printed)

_____

Student Signature

_____

Date

Contents

0.1 Dedication

I would like to dedicate this work to my parents out of gratitude for the loving support they have provided throughout my academic career and beyond. Without the love of knowledge they instilled in me I simply would not be writing this. I would also like to extend my deepest appreciation for the support my advisor, Dr. Rahman, has provided me in preparing this work.

0.2 Abstract

With the globalization of the semiconductor manufacturing process, electronic devices are powerless against malicious modification of hardware in the supply chain. In order to maintain supply chain security and prevent the insertion of Trojans two possible approaches can be taken. One possible approach is to establish a trusted supply chain from the design house to the finished product. However, this often a difficult and expensive endeavor. The other possible solution is verify the integrity of the final product once it has been delivered.

This second option has spurred a need for accurate and efficient detection methods. The Ring oscillator network (RON) architecture is used to detect the Trojan by capturing the difference in power consumption; the power consumption of a Trojan-free circuit is different from the Trojan-inserted circuit. However, the process variation and measurement noise are the major obstacles to detect hardware Trojan with high accuracy. In this work, a quantitative comparison is used to evaluate four supervised machine learning algorithms and classifier optimization strategies for maximizing accuracy and minimizing the false positive rate (FPR). These supervised learning techniques show an improved false positive rate compared to principal component analysis (PCA) and convex hull classification by nearly 40% while maintaining $> 90\%$ binary classification accuracy.

Chapter 1

Introduction

While the transition from vertically integrated supply chains to horizontally integrated has decreased costs for integrated circuit (IC) designers; the "fabless" approach comes with the steep price of trust [2, 3, 5, 6, 7, 9, 10, 11, 12]. Semiconductor designers now must trust their intellectual property (IP) to multiple parties in order to have their ICs manufactured at foundries [12, 9]. Not only do they run the risk of having their IP stolen, but it is not uncommon for untrusted system integrators and foundries to insert hardware Trojans before shipping the final product [2, 3, 5, 6]. These Trojans are capable of leaking sensitive information, disabling key portions of the IC, self-destructing the chip, or hindering performance [2, 3]. This has driven the need for fast, accurate, and simple methods of detecting infected ICs before they are able to taint the supply chain.

These attacks are only on the rise and growing in magnitude. Recently, allegations that even one of the world's most well-known computer hardware manufacturers, SuperMicro, was subject to such additions to the hardware they prepared for technology giants such as Amazon and Apple [1]. Furthermore, the United States government has been quite active in finding ways of securing government and contractor supply chains [31]. Modern supply chains are complex and require systems of trust to prevent malicious actors taking advantage of any steps. By and large these supply chains have become more exploitable due to the transition from vertically to horizontally integrated supply chains. In an vertically integrated configuration every stage of the manufacturing process is owned by one company. Everything

from the initial design to the manufacturing to the final shipments are handled by groups owned by a single company. While these configurations are much more secure they are often expensive, and the rising costs of shrinking process nodes has driven semiconductor designers to outsource fabrication to one of the few global foundries. Fig. 1.1 provides an example of the many places a destination can reach before its final delivery.

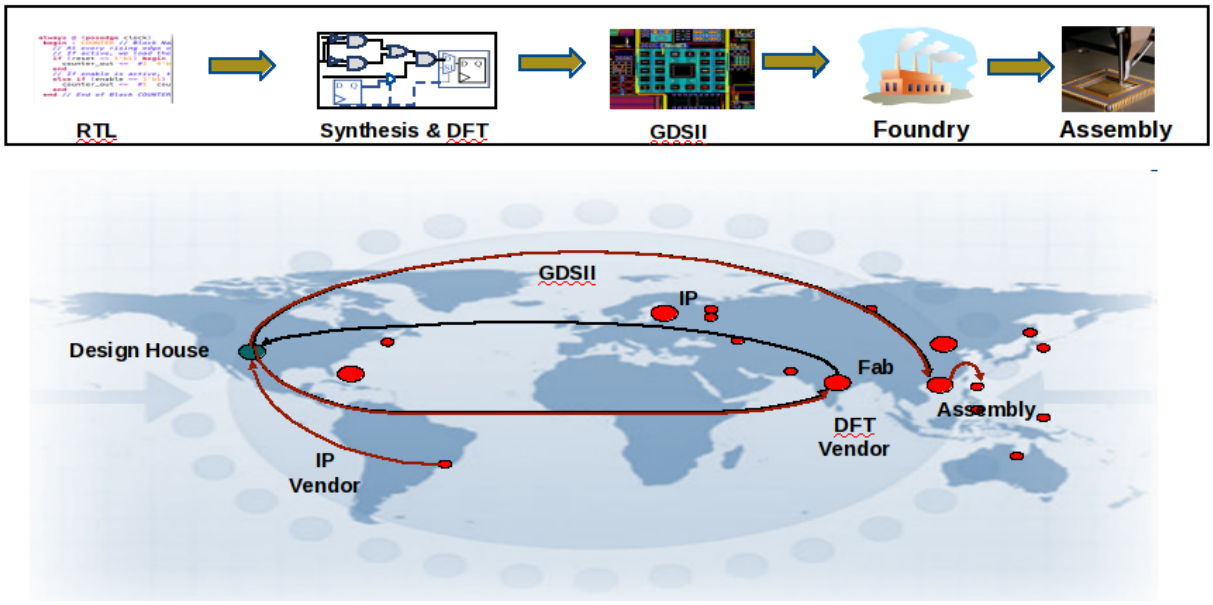

Figure 1.1: A view of a modern semiconductor supply chain [10].

With this rising need, some research in the field of hardware security has been focused on finding optimal methods of detecting and classifying Trojans. Initial research suggested the use of semi-invasive strategies such as scanning electron microscopy (SEM) for failure analysis. However, this is expensive and time consuming for it to be applied to every IC. Using netlist failure detection techniques was also unsuccessful due to Trojans that add functional logic remaining undetected.

The most promising technique relies on the use of side channel information as it is non-invasive and can be done quickly. By monitoring side channel information from an IC power grid it is possible to detect Trojans due to their additional activity [16]. In [14], the authors developed a ring oscillator network (RON) in a chip's power grid for hardware Trojan detection. The increased switching activity from Trojan activation will manifest itself

in decreased RO frequencies due to the variable voltage drop in the chip's power network. Using Principal Component Analysis (PCA) and convex hull classification ([20, 21]) they were able to achieve greater than 80% classification accuracy with a false positive rate of 50%.

This was improved upon in [17] using a genetic algorithm (GA) for feature reduction and a support vector machine (SVM) for classification. Feature reduction allows machine learning algorithms to reduce the feature space and decrease training time and the possibility of over fitting. The genetic algorithm is built upon the idea of natural selection where the best features will "survive" through each generation. When the end of the algorithm is reached you will be left with the optimal feature set. This in addition to the use of SVM resulted in 99.6% classification accuracy and a reduced false positive rate. However, [17] still suffers from a large FPR.

In this work, we present a supervised machine learning approach for the classification of Trojan free and infected ICs using a RON. The results show that we maintain similar accuracy to previous work in addition to reducing the FPR by using the K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Naive Bayes, and ensemble classification algorithms [24, 25, 26, 27]. Experimental results show detection accuracy $> 88\%$ with some classifiers even reaching 97.4%. Low false positive rates (FPR) were also achieved and in the case of two classifiers a $\sim 0\%$ FPR was reached.

The rest of this work will be laid out as follows: Chapter 2 will provide all necessary background information, Chapter 3 will discuss the proposed method of classification and results, Chapter 4 will discuss possible future work, and Chapter 5 will conclude the work.

Chapter 2

Background

2.1 Hardware Trojans

Hardware Trojans are malicious modifications made by attackers during the design and manufacturing process [2, 3, 6, 7]. Trojans can be used to degrade performance, steal information, or block functionality of an IC [2, 3]. These unwanted circuit additions are often hard to detect because they are not always triggered or activated by standard test procedures [2, 3]. Trojans also come in a wide variety of formats so no single filter can catch them all. These Trojans are unwanted and pose a risk to the chip owners of receiving secretly modified hardware that could lead to devastating consequences [2, 3].

2.1.1 Taxonomy

A hardware Trojan can be classified into three main categories according to their physical, activation, and action characteristics [2, 3, 15]. The first category based on physical characteristics of the Trojan classifies based on whether they are functional or parametric based. Functional Trojans are those that require the addition of functional logic in order to operate. Parametric, on the other hand, require only the modification of existing wires or layout. The next category analyzes the Trojan based on its activation characteristics [2, 3, 15]. In general this will determine whether a Trojan is activated by an internal or external trigger, and further classify based on the duration for which the Trojan is active.

Internally activated Trojans are used in the case where the malicious actor would like the Trojan to activate in known conditions. For example, this could be used to activate a Trojan to leak plaintext from an encryption step or to serve as a "timebomb" to destroy the circuit after a set amount of time has elapsed. The final category uses action characteristics to determine the classification. As stated above, Trojans can really perform three main functions: transmit information, modify the specification, or modify the function of the design [2, 3, 15]. While some Trojans fit neatly into these categories there are often Trojans that do not. This taxonomic scheme was designed to provide the means to also classify "hybrid" Trojans that are not as simple. A graphical overview of the taxonomic system can be seen in 2.1.

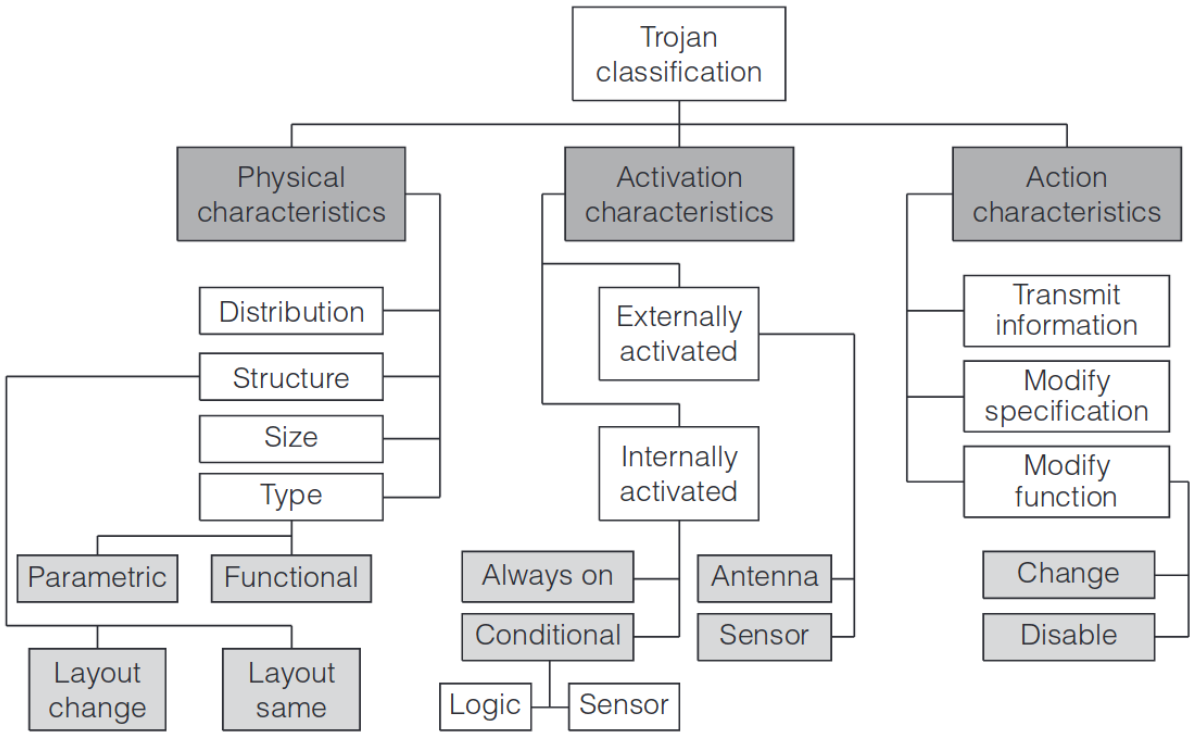

Figure 2.1: An overview of the hardware Trojan taxonomy. [15]

2.1.2 Prior Work in Trojan Detection

Initial research into the detection of hardware Trojans was through the use of physical inspection. These often consisted of semi-invasive techniques such as scanning electron or scanning optical microscopy. While these methods were feasible for smaller designs they failed to scale well. Modern integrated circuits can contain millions of transistors and viewing each and every IC after they are produced is simply too expensive and too slow.

Current Trojan detection methods largely focus on (i) functional testing and (ii) side channel analysis [7, 8, 14, 15, 16, 17]. Functional verification is the attempt to activate Trojans by applying test vectors and comparing the responses with the correct results [2, 16]. The difficulty with this approach is the rarity of which some hardware Trojans are activated. It is nearly impossible to explore every possible state of a circuit and search for Trojan activity [7]. Whereas, side-channel analyses detect the HT by analyzing the physical characteristics of the IC chip such as transient current, leakage current, delay, energy, heat generation, or EM radiation [7, 8, 13, 14, 15, 16]. In both approaches, the outputs of circuit under test are compared with the outputs of a golden circuit. Typically, the adversary would design a Trojan to evade detection by ensuring rare activation to evade logic testing and minimal physical characteristics, like size, to escape side channel based testing. Backside optical imaging of the fabricated chip enables extraction of the full standard cell layout of the chip with the watermarks, which in turn can be validated with image processing against the expected simulated layout to detect any changes made to accommodate hardware Trojans [13]. A challenge in backside imaging is obtaining a high enough spatial resolution for an accurate representation of a nanometer-scale circuit [13].

While all of these methods can provide accurate detection of hardware Trojans there is one issue that has inspired the use of machine learning and other mathematical based methods. Due to process variations, or the variation in manufactured ICs, it is difficult to discern differences in signatures due to process variations or the addition of Trojans. What could appear to be an infected IC based on its signature could in fact be caused by process

variations and vice versa. However, the recent popularity boom of machine learning has inspired the use of these algorithms for detection [17]. Machine learning algorithms are very useful for learning patterns in data that may not otherwise be visible to the human eye or even to simple algorithms. This results in the ability to find the details between a truly infected IC and one that had its signature changed due to process variations.

2.2 Ring Oscillator Network Architecture

As stated before there are many ways of using information from ICs for Trojan detection. In this work the central method of gathering information from ICs is the ring oscillator network. Initially proposed in [35], the goal of the ring is to provide a localized measurement of IC power consumption. Before explaining how the network is able to capture the activity of hardware Trojans it is first necessary to explain the concept of a ring oscillator. A ring oscillator is a simple structure created with an odd number of inverters as shown in 2.2. Since the inverters will output the inverse of their input the structure continuously alternatives between two voltage levels. However, the delay will depend on the number of stages and can be defined by as $2 * n * t_d$ where n is the number of stages and t_d is the delay for each stage [35]. From this we can derive the frequency of the entire ring oscillator as $f = \frac{1}{(2*n*t_d)}$.

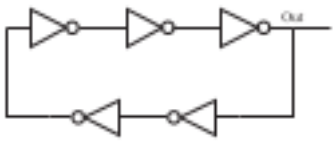

Figure 2.2: A simple ring oscillator [35]

Granted these properties alone are not enough to provide information about possible Trojans. However, integrated circuits have a relation between increased noise in the power supply and increased gate delays. Using these two properties and the fact that most Trojans will result in increased switching activity it is then possible to find infected chips with this information. By placing several ring oscillators in the power supply of an IC this noise can

be captured by measuring the ring oscillators frequency. If the current for a chip with no Trojans and the ring oscillators can be defined as follows [35]:

$$\frac{I_{total}}{f} = \sum_{i=0}^{i=N} \lambda_i * N * 2n * k_g \tag{2.1}$$

Then with Trojans present it will be defined as follows:

$$\frac{I_{total,t}}{f_t} = \sum_{i=0}^{i=N+n_t} \lambda_i * (N + n_t) * 2n * k_g * (1 + \alpha * \frac{\Delta V_t}{V_{dd} - \Delta V_t - V_{th}}) \tag{2.2}$$

By placing a network of ring oscillators in the power structure of the IC it is possible to have a localized global power measurement. The ring oscillator network structure also has the advantage of having a small size and power footprint. The measurement process consists of applying a series of test patterns through a linear feedback shift register and using a multiplexer to select which ring oscillator to measure. Once all the measurements are complete the ring oscillators are disabled. This prevents any extra power draw during the chips normal functionality. Additionally, the architecture is resistant to tampering as any change to even one of the ring oscillators will result in detection.

Chapter 3

Supervised Machine Learning Techniques for Trojan Detection with Ring Oscillator Network

3.0.1 Objectives

This chapter aims to provide a quantitative comparison of four popular supervised machine learning techniques in the application of hardware Trojan detection. While previous work has shown high classification accuracy is possible, some have also shown high false positive rates. Through this comparison the aim is to find a technique that will maintain the high classification accuracy (>90%) but reduce the false positive rate to under ∼10-15%. The four techniques were selected based not only on these objectives, but also on their other characteristics. For example, K-Nearest Neighbors is a simple and fast classifier that can perform well in many situations. The support vector machine is slightly slower, but has proven itself to be very accurate. Additionally, the Naive-Bayes classifier was chosen for its prior work in categorization in fields such as spam detection. The final technique of ensemble learning was chosen to try to extract the optimal characteristics of all these algorithms.

3.0.2 Supervised Learning

In the field of machine learning there are two main approaches in use: supervised and unsupervised learning [23]. Unsupervised learning is outside the scope of this work and will

not be discussed further for the sake of brevity. Supervised learning algorithms work under the assumption the training data is labeled before being processed by the algorithm. By labeling the data, the algorithm then knows the desired output for the given input set and can create a hypothesis for determining the desired output for future inputs [23]. Within the topic of supervised learning exist two problem types: regression and classification. Regression algorithms are going to map input values to a real output value, e.g., predicting stock market prices given a feature set. On the other hand, classification algorithms will place a set of input data points into one or more "classes", e.g. Trojan free or Trojan infected.

In this work, a binary classification ([22, 23]) approach is used to classify each IC as either Trojan free or Trojan infected. In order to properly train the classifier we must operate under the assumption we have data from both Trojan free and Trojan infected circuits. Obtaining known Trojan free ICs is a challenge in and of itself, but knowing which ICs are infected with Trojans will require some other method of detection until enough data can be collected to train a classifier.

3.0.3 K-Nearest Neighbors

One of the simplest and yet most popular machine learning algorithms is k-nearest neighbors (KNN). When used for classification the k nearest training samples in the feature space are used to classify the new point through a simple majority vote. This simplicity does come with the cost of longer classification times for larger data sets. The value k is usually defined as a positive integer, and in the case of binary classification it is useful to set k as an odd number to prevent a split decision. The distance metric can be any method of calculating distance, but Euclidean distance is often used. It can be defined as follows:

$$d(x, y) = \sqrt{(x_1 - y_1)^2 + \cdots + (x_n - y_n)^2} \tag{3.1}$$

As can be seen in 3.1, the value of k will have an effect on the classifiers performance. By comparing the mean error and accuracy values for a series of k values it is possible to find the most optimal k value for a data set.

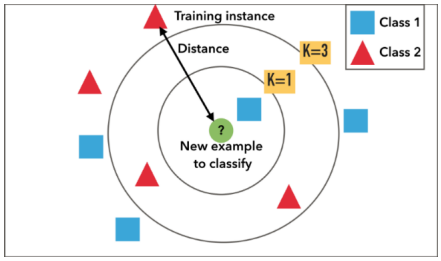

Figure 3.1: An example of the KNN algorithm showing the effect the value of k has on classification. Notice that the new example will be classified as class 1 if $k = 1$, but class 2 if $k = 3$ [18].

The classifier was trained using a range of k values from 1 to 40 and the value with the best FPR and accuracy was selected without being over fitted. It is usually safe to select a value of k near the square root of the number of training samples. However, low values of k will lead to a classifier that performs worse with noisy data, and high values of k can lead to over fitting the classifier to the training data.

3.0.4 Support Vector Machine

Another popular machine learning algorithm is known as the support vector machine (SVM) [25]. While not as simple as KNN it is much more powerful for classification and regression applications. The training of the SVM consists of finding the optimal hyperplane that will linearly classify data points with the largest margin possible between the two classes of data points. However, not all data can be linearly separated by a hyperplane in which case we must apply a "kernel trick" to transform the feature space.

If we define our training data as a set of points in the form of $(\vec{x_1}, y_1), \ldots, (\vec{x_n}, y_n)$, where $\vec{x_i}$ is a vector and y_i is -1 or 1 to represent the class of $\vec{x_i}$, then we can define our hyperplane

as satisfying the following equation [25]:

$$\vec{w} \cdot \vec{x} - b = 0 \tag{3.2}$$

If the data set is linearly separable then one class can be defined as anything on or above the boundary $\vec{w} \cdot \vec{x} - b = 1$ and the other class can be defined as anything on or below $\vec{w} \cdot \vec{x} - b = -1$. Now in order to train the SVM we want to minimize the difference between these two hyperplanes so that the margin between the two classes is maximized. Thus the problem simplifies down into:

$$\min_{n^1 \ldots n^i} \text{ for } y_i(\vec{w} \cdot \vec{x_i} - b) \geq 1 \tag{3.3}$$

The classifier will then be defined by $\vec{w}$ and b. An example of a hyperplane used to separate two classes of data can be seen in 3.2.

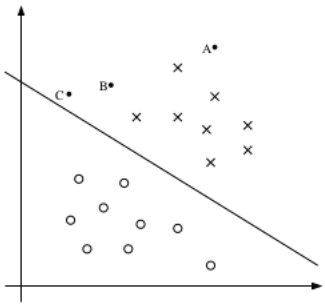

Figure 3.2: Example of a SVM hyperplane separating two classes of data. In this example A, B, and C would be classified by computing the dot product. They all wold be classified as class 'x' [19]

This method will only work for linearly separable data and the classification of other data requires the replacement of the dot product with a nonlinear kernel function, thus the name "kernel trick". By using the kernel function we can now put a hyperplane in our higher dimensional nonlinear feature space. In this work, a Gaussian radial base function was used.

$$k(\vec{w_i}, \vec{x_j}) = exp(-\gamma \|\vec{x_i} - \vec{x_j}\|^2) \text{ for } \gamma > 0 \tag{3.4}$$

3.0.5 Naive Bayes

The construction of classifiers using Naive Bayes is a relatively simple process that can produce highly accurate and fast classification results using a probabilistic approach[26]. Using Bayes theorem we can generate the probability that a data point will belong to a class C_k given the presence of one of the features of the data [26].

$$P(C_k|x) = \frac{P(x|C_k)P(C_k)}{P(x)} \tag{3.5}$$

However, when trying to build a classifier we are interested in the probability a data point belongs in class given multiple features. Using the chain rule Bayes theorem can be expanded to account for this. Assuming the n features in the data set can be represented as $X = (x_1, x_2, ..., x_n)$ then it follows [26]:

$$P(C_k|x_1, ..., x_n) = \frac{P(x_1|C_k)P(x_2|C_k)...P(x_n|C_k)P(C_k)}{P(x_1)P(x_2)...P(x_n)} \tag{3.6}$$

Now if we assume the conditional independence of the features in the set:

$$P(C_k|x_1, ..., x_n) = \frac{P(C = C_k)\prod_i P(x_i|C = C_k)}{\sum_j P(C = C_j)\prod_i P(X_i|C = C_j} \tag{3.7}$$

Finally, to create a classification rule we must have a way of making decisions. Using a *maximum a posteriori* rule, or simply stated choosing the most probable outcome, we can decide how to assign class labels to data points.

$$y = argmaxP(C_k)\prod_{i=1}^{n} P(x_i|C_k) \tag{3.8}$$

Naive Bayes can be applied in one of three ways to estimate the likelihood of the features. A Gaussian classifier will assume the features are distributed on a Gaussian distribution, Multinomial will assume multinomially distributed data, and Bernoulli will assume binary-valued features. The Gaussian classifier was selected in this work due to the continuous

nature of the data set.

3.0.6 Ensemble Learning

Ensemble learning is another technique for producing better prediction results using machine learning algorithms [27]. It operates under the assumption that by combining the predictive power of single algorithms it is possible to increase the overall possible predictive power. Several popular strategies include voting, bagging, stacking, boosting, and "bucket of models". In this work, we will only be implementing a simple voting method. This is done by taking the output of each of the three classifiers and using it as a vote. The class with the most votes will then be the output of the classifier. Theoretically, given an odd number of classifiers in the ensemble a decision should be made every time that represents the best of each classifier [27]. However, if the ensemble is made up of an even number of classifiers situations can arise resulting in split decisions. This is said to be an unstable decision. This can be mitigated through the use of either an odd number of classifiers or using weighted voting to reduce the possibility of a split decision.

3.0.7 Ring Oscillator Network and Trojan Detection

Recent work has shown that a ring oscillator (RO) network (RON) connected to the power supply structure of an IC can be used to detect hardware Trojan activity. As shown in 3.3, ROs consisting of inverters and a NAND gate for activation control are placed in a vertical orientation within the power structure of an IC. The ROs are then provided test patterns from a linear feedback shift register and a decoder. These outputs are then selected using a multiplexer and a counter registering the number of oscillations from the selected RO. The RO's frequency can then be derived from the number of oscillations. Any Trojan inserted into an IC will result in extra noise in the power supply structure that would not otherwise be present in a "golden" chip. By injecting the same test patterns into every IC the Trojans should at least partially active and thus cause extra noise. Since a RO's frequency is directly

related to its power supply voltage this Trojan caused power supply noise should propagate to the RO's frequency and result in differing measurements between clean and infected ICs [14, 17].

However, the frequency differences are not always discernible to the human eye nor to simple algorithmic classification strategies due to process variations and other factors. In [14] Principal Component Analysis (PCA) was used as a means of feature reduction. The data set contained the frequency data from 8 ROs, but through feature reduction could be accurately represented with just 3. A simple convex hull classification method was then used to classify each IC as either Trojan free or into one of the 23 Trojan categories. While the RON is successful at detecting the difference between Trojan free and Trojan infected circuits the false positive rate was nearly 50%. Using the data collected from the RON we will try to improve on this false positive rate while maintaining above 90% classification accuracy.

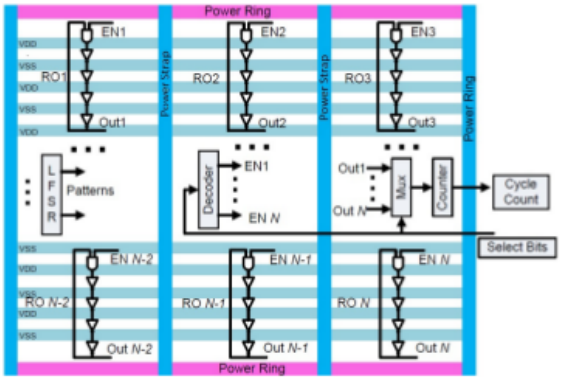

Figure 3.3: The ring oscillator network used for Trojan detection. While 8 ROs are used in this configuration the structure will differ based on the power network of the IC you are trying to protect [14, 17]

3.0.8 Experimental Set-up

We conducted our experiments on eight FPGA boards (Nexys4 DDR development board [28]). Each FPGA board is divided into four separate regions to increase the sample size. Each region is considered as an individual IC and Trojan, and the RON architecture is

implemented in only a single portion at a time in order to make sure that one portion (or an individual IC) does not interfere another. We used a total of eight 41-stage ROs in each portion (i.e., IC). We distributed combinational and sequential Trojans ([29]) in one portion randomly. We used several Trojan benchmarks from Trusthub [29]. We measured the average RO frequency at room temperature and nominal operating voltage from 50 measurements (with Trojan and without Trojan) to cancel out the measurement noise. We included ITC-99 ([30]) benchmarks for normal operation.

3.1 Method

The method we will use is to use the four previously discussed supervised classification approaches and optimize them for accuracy and a low FPR. The main motivation for this is to reduce potential waste of Trojan free ICs that would otherwise be discarded due to being classified as infected. However, accuracy must be maintained to prevent Trojans from being introduced into the supply chain.

In order to do this, from the collected data, each chip has readings for two "golden" or Trojan free samples and 23 Trojan inserted samples. The data was collected using the test setup described in 3.0.8. This data was then be labeled accordingly and used to train the classifiers.

The KNN classifier will then be optimized by finding the best k value for maintaining accuracy and minimizing the FPR. By training the KNN classifier on a range of k values and different training sample sizes we were able to select the best value for our data set. The SVM classifier will be optimized using two slack values pertaining to the Gaussian kernel function, C and γ. C can be considered the weight correct classification has over maximizing the margin between the two classes. Gamma is the inverse of the variance of our Gaussian function. Thus, a small γ will lead to a large variance and points could be similar even if they are not close together and vice versa. In order to find the optimal γ and C values we have

used a grid search method in which a given set of values is exhaustively run through until the best values for the data set are found. The Naive Bayes Gaussian classifier will not be tuned using any parameters. Each of the three classifiers will then be combined in a simple voting ensemble in the following combinations: KNN+SVM+Naive Bayes, KNN+SVM, KNN+Naive Bayes, and SVM+Naive Bayes. The KNN and SVM classifiers will retain the same optimization parameters as they had being trained individually.

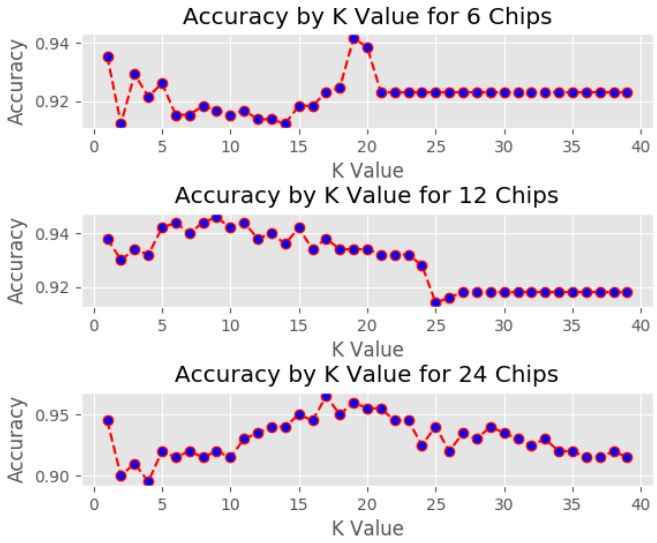

Figure 3.4: A plot showing the effect the value of k has on the classifier accuracy. As can be seen a k value of 2 provides enough accuracy without being over fitted.

3.2 Results

Following the method above each classifier was trained and optimized for three different sized data sets consisting of 6 chips, 12 chips, and 24 chips. Each sample size was then repeated for 20 trials and the average accuracy, false positive rate (FPR), false negative rate (FNR), true negative rate (TNR), and true positive rate (TPR) were calculated and recorded as follows:

$$TPR = \frac{TP}{TP + FN} \tag{3.9}$$

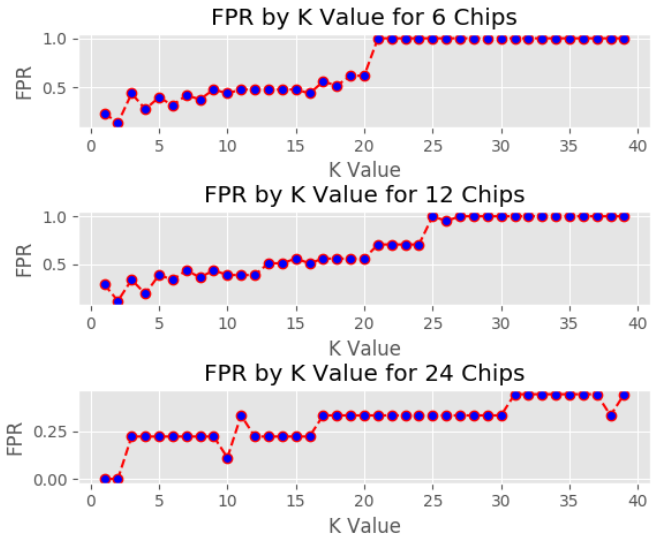

Figure 3.5: A plot showing the effect the value of k has on the classifier false positive rate. As shown higher values of k will lead to over fitting and higher false positive rates.

$$TNR = \frac{TN}{TN + FP} \tag{3.10}$$

$$FNR = \frac{FN}{FN + TP} \tag{3.11}$$

$$FPR = \frac{FP}{FP + TN} \tag{3.12}$$

The optimization step of the training led to the discovery of useful properties of our data set. Initial estimates for the value of k when training the KNN classifier used the square root of the number of samples in the training data set. While this resulted in a very accurate classifier it came at the cost of a FPR greater than or equal to 50%. This is most likely a result of the data set having little noise and being very prone to over fitting. In Figure 3.4, the accuracy for a range of k values is depicted, note that as the value increases the accuracy tends to plateau as a result of over-fitting. Figure 3.5 shows the same plateau for the FPR. Since we want to avoid over-fitting and lower k values perform well we can assume the data set is not noisy. This led to the decision to use a k value of 2 for every sample size. This maintained the greater than 90% accuracy benchmark and had a best case FPR of only 9.4%, a near 40% decrease compared to PCA and convex hull classification. Even

with small training sets the KNN maintained a FPR under 20% (Table 3.1).

Table 3.1: KNN Classifier Results

Metric	**Sample Size**		
	6 Samples	**12 Samples**	**24 Samples**
TNR	0.813	0.815	0.906
FPR	0.187	0.185	0.094
FNR	0.075	0.063	0.051
TPR	0.916	0.927	0.745
Accuracy	0.916	0.927	0.945

Optimizing the SVM proved to be more difficult than the KNN classifier. The grid search was quick to converge on a C value of 1 and γ value of 0.1, but the FPR left much to be desired. As can be seen in Table 3.2, the SVM is very accurate but when trained on fewer samples it struggles with a high FPR. Using a balancing optimization it was still able to achieve a 97.4% classification accuracy and a 7.1% FPR (Table 3.2) and outperform convex hull and approach the results achieved in [17]. This leads me to believe that with a larger data set and increased training set sizes the SVM could become more accurate and reduce the FPR even further. Unfortunately, it is not always possible to have large data sets due to factors outlined above.

Table 3.2: SVM Classifier Results

Metric	**Sample Size**		
	6 Samples	**12 Samples**	**24 Samples**
TNR	0.445	0.605	0.929
FPR	0.555	0.355	0.071
FNR	0.017	0.023	0.023
TPR	0.983	0.977	0.977
Accuracy	0.940	0.946	0.974

Despite the many operating assumptions the Naive Bayes classifier is a very powerful but simple and fast method. With no optimization the classifier produced results that were slightly less accurate compared to the other classifiers. At the 6 chip sample size the

classifier was 88.3% accurate but had only a 6.9% FPR. The accuracy only dropped 0.1% when increasing the training sample size to 12 chips, but the FPR dropped to 6.1%, the lowest FPR of any non-ensemble classifier (Table 3.3). The Naive Bayes classifier produced the best results in term of FPR but was held back by a higher FNR which led to reduced accuracy. In theory, this could be reduced by tuning the decision threshold, but would most likely result in the FPR increasing.

Table 3.3: Naive Bayes Gaussian Classifier Results

Metric	**Sample Size**		
	6 Samples	**12 Samples**	**24 Samples**
TNR	0.931	0.955	0.939
FPR	0.069	0.045	0.061
FNR	0.121	0.124	0.127
TPR	0.879	0.876	0.873
Accuracy	0.883	0.882	0.873

When using ensemble learning the hope is the results are better than that of each of the individual classifiers by themselves. However, it also runs the risk of the opposite occurring. We encountered both situations while training the ensembles. The ensemble containing all three classifiers performed better than the lone SVM classifier at the lower training sample sizes. Yet, it was outperformed at the 24 chip sample size (Table 3.4). The Naive Bayes and KNN/SVM ensembles had the lowest overall FPRs of all classifiers, but struggled to beat the desired 90% binary classification accuracy threshold (Tables 3.6 & 3.7). This can be attributed to the Naive Bayes classifier's characteristics dominating those of the other classifiers. Despite the lower accuracy, at the 24 chip training sample size both ensembles had a 0% FPR. Overall, the best ensemble method was the combination of the SVM and KNN classifiers (Table 3.5). At the lower training sample sizes the FPR was only 19.6% and 16.4%, but kept an accuracy of 92.1% and 93.0% respectively. At the 24 chip training sample size the FPR was 0.03% higher than the SVM alone but with a 3.4% accuracy loss.

Considering the results, the choice for the best approach is very dependent on the data

Table 3.4: SVM+KNN+NB Ensemble Classifier Results

Metric	**Sample Size**		
	6 Samples	**12 Samples**	**24 Samples**
TNR	0.785	0.796	0.908
FPR	0.215	0.204	0.092
FNR	0.066	0.062	0.055
TPR	0.934	0.938	0.945
Accuracy	0.922	0.927	0.943

Table 3.5: SVM+KNN Ensemble Classifier Results

Metric	**Sample Size**		
	6 Samples	**12 Samples**	**24 Samples**
TNR	0.804	0.836	0.926
FPR	0.196	0.164	0.074
FNR	0.069	0.062	0.058
TPR	0.931	0.938	0.942
Accuracy	0.921	0.930	0.940

Table 3.6: SVM+NB Ensemble Classifier Results

Metric	**Sample Size**		
	6 Samples	**12 Samples**	**24 Samples**
TNR	0.939	0.953	1.000
FPR	0.061	0.047	0.000
FNR	0.125	0.127	0.129
TPR	0.875	0.873	0.871
Accuracy	0.880	0.879	0.881

Table 3.7: KNN+NB Ensemble Classifier Results

Metric	**Sample Size**		
	6 Samples	**12 Samples**	**24 Samples**
TNR	0.982	0.993	1.000
FPR	0.018	0.007	0.000
FNR	0.122	0.126	0.137
TPR	0.878	0.874	0.863
Accuracy	0.886	0.883	0.873

set and desired outcomes. The Naive Bayes and KNN classifiers are extremely fast, simple, and do well at maintaining low FPRs and moderate accuracy throughout the sample sizes. Combining the SVM and KNN classifiers in an ensemble allowed the classifier to maintain greater than 90% accuracy, but kept the FPR lower compared to using a SVM alone. Thus, with very little data the best classification performance will come from a Naive Bayes or ensemble containing the Naive Bayes classifier such as the KNN and NB ensemble. However, with sufficient data the SVM classifier alone still provides the best trade off between accuracy and FPR.

3.3 Summary of Contribution

.....The major contributions of this thesis have been highlighted below.

- Using RON Architecture to Detect Hardware Trojan: The FPGA implementation of the RON architecture was successfully able to collect frequency data for Trojan free ICs as well as combinational and sequential Trojan infected ICs.

- Comparison of Different Machine Learning Algorithms on Detecting Hardware Trojan: The quantitative comparison of four different machine learning algorithms showed several possible best options for a range of training set sizes. Large training sizes favored the SVM classifier while the Naive-Bayes and ensemble classifiers showed promise with smaller training sets.

- Validation with Silicon Results: The use of the the Nexys4 DDR FPGA board provided a data set that validates the results of the classifiers for detecting hardware Trojans in actual ICs.

Chapter 4

Future Work

As seen in the previous chapter and in other works, supervised machine learning methods can provide highly accurate results with the proper tuning and data sets. However, there is an inherent problem in that gathering large data sets is difficult. For an untrusted supply chain it requires the gathering data about a set of ICs before reverse engineering them and finding the known "golden" chips. This is often an expensive and laborious process. It also makes training effective classifiers difficult. As can be seen in the previous chapter the most effective classifiers performed well when they had larger training sets. Unfortunately in an untrusted supply chain there are not many workarounds to such a problem.

With this in mind, I have begun preliminary investigations into the use of unsupervised methods of classification. However, the results were insufficient for publication at this time. Unsupervised methods operate using a data set that is not marked whether the chips are Trojan free or infected. Rather, they use solely the features of the data set to learn about underlying patterns. Theoretically, this would allow the classification of chips with very little "golden" chip data. By looking at the suggested classifications and mapping it with the limited "golden" chips it may prove to be a viable option.

Additionally, generative models have proven to be very promising in the field of machine learning. One of the most difficult problems in the field is creating algorithms that are

truly learning and "understanding" the data. Generative models aim to do this by taking input data and using a neural network to create similar data to the original data set. The theory behind the generative approach is a model that can create similar data has effectively captured the features of the original data set. In the case of the generative adversarial network (GAN) the generated data is then fed to a adversarial network that makes decisions on whether the data came from the initial distribution or was created by the generative network. Not only does this create a network that is extremely efficient at generating data but one that is also a capable classifier. In the case of hardware Trojans, this may provide a means of accurate classification with very limited data sets.

Chapter 5

Conclusion

5.1 Conclusion

While the field of hardware Trojan detection is relatively new, it is not without very difficult problems. Trojan developers are continuously working on their approaches and becoming better at hiding their malicious designs. This will continue to drive the need for efficient and accurate methods of Trojan detection. In this work, a quantitative comparison of four supervised machine learning algorithms' performance when classifying ICs based on their ring oscillator network frequencies was presented. This method was able to achieve 97.6% binary classification accuracy and a FPR of just 7.1% when using a SVM classifier, and ensemble approaches achieved ∼88% accuracy with nearly no false positives. Despite these promising results, supervised learning approaches are often impractical in a real supply chain. As discussed in 4, finding proven 'Golden chips' is a challenge and knowing which chips are infected at the scaled assumed in the data set is near impossible. The discussed future work aims to serve as a workaround for this problem.

5.1.1 Acknowledgment

Portions of this work was supported in parts by the National Science Foundation under Grant Number CNS-1850241 and UAH NFR. We would like to thank Dr. Tehranipoor and Dr. Forte for sharing the resources on Trusthub [29].

Bibliography

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

[2] M. Tehranipoor and F. Koushanfar, "A Survey of Hardware Trojan Taxonomy and Detection," in IEEE Design & Test of Computers, vol. 27, no. 1, pp. 10-25, Jan.-Feb. 2010.

[3] Shakya, Bicky, Tony He, Hassan Salmani, Domenic Forte, Swarup Bhunia, and Mark Tehranipoor. "Benchmarking of hardware Trojans and maliciously affected circuits." Journal of Hardware and Systems Security 1, no. 1 (2017): 85-102.

[4] Mark Tehranipoor and Hassan Salmani, "Trojan Benchmarks" Available: https://www.trust-hub.org/benchmarks/trojan"

[5] Jyothi, Vinayaka, and Jeyavijayan JV Rajendran. "Hardware Trojan Attacks in FPGA and Protection Approaches." In The Hardware Trojan War, pp. 345-368. Springer, Cham, 2018.

[6] Salmani, Hassan. "Trusted Testing Techniques for Hardware Trojan Detection." In Trusted Digital Circuits, pp. 109-119. Springer, Cham, 2018.

[7] Cui, Xiaotong, Kaijie Wu, and Ramesh Karri. "Hardware Trojan detection using path delay order encoding with process variation tolerance." In 2018 IEEE 23rd European Test Symposium (ETS), pp. 1-2. IEEE, 2018.

[8] Plusquellic, Jim, and Fareena Saqib. "Detecting Hardware Trojans Using Delay Analysis." In The Hardware Trojan War, pp. 219-267. Springer, Cham, 2018.

[9] M. T. Rahman, D. Forte, Q. Shi, G. K. Contreras and M. Tehranipoor, "CSST: Preventing distribution of unlicensed and rejected ICs by untrusted foundry and assembly," 2014 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), Amsterdam, 2014, pp. 46-51.

[10] Rahman, Md Tauhidur, Domenic Forte, and Mark M. Tehranipoor. "Protection of Assets from Scan Chain Vulnerabilities Through Obfuscation." In Hardware Protection through Obfuscation, pp. 135-158. Springer, Cham, 2017.

[11] Rahman, Md Tauhidur, Domenic Forte, Quihang Shi, Gustavo K. Contreras, and Mohammad Tehranipoor. "CSST: an efficient secure split-test for preventing IC piracy." In Test Workshop (NATW), 2014 IEEE 23rd North Atlantic, pp. 43-47. IEEE, 2014.

[12] Banga, Mainak, and Michael S. Hsiao. "Hardware IP Trust." In The Hardware Trojan War, pp. 75-100. Springer, Cham, 2018.

[13] Boyou Zhou et al., "Detecting Hardware Trojans using backside optical imaging of embedded watermarks," 2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC), San Francisco, CA, 2015, pp. 1-6.

[14] Shane Kelly, Xuehui Zhang, Mohammed Tehranipoor, and Andrew Ferraiuolo, "Detecting Hardware Trojans using On-chip Sensors in an ASIC Design." Journal of Electronic Testing 31, no. 1 (2015): 11-26.

[15] Tehranipoor, M., & Koushanfar, F. (2013). "A Survey of Hardware Trojan Taxonomy and Detection." IEEE Design & Test, 1–1.

[16] Wang, Xiaoxiao & Tehranipoor, Mark & Plusquellic, J. (2008). Detecting malicious inclusions in secure hardware: Challenges and solutions. 15-19. 10.1109/HST.2008.4559039.

[17] Karimian, Nima & Tehranipoor, Fatemeh & Forte, Domenic & Rahman, Md Tauhidur. (2015). Genetic Algorithm for Hardware Trojan Detection with Ring Oscillator Network (RON).

[18] Bronshtein, A, "A quick introduction to K-Nearest Neighbors Algorithm". 2017.

[19] Ng, A. "CCS229 Lecture Notes: Support Vector Machines". Stanford University, 2018.

[20] Lever, Jake, Martin Krzywinski, and Naomi Altman. "Points of significance: Principal component analysis." (2017): 641.

[21] Tang, Min, Jie-yi Zhao, Ruo-feng Tong, and Dinesh Manocha, "GPU accelerated convex hull computation." Computers & Graphics 36, no. 5 (2012): 498-506.

[22] Yang, Zhiguang, and Haizhou Ai. "Demographic classification with local binary patterns." In International Conference on Biometrics, pp. 464-473. Springer, Berlin, Heidelberg, 2007.

[23] Chapelle, Olivier, Bernhard Scholkopf, and Alexander Zien. "Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]." IEEE Transactions on Neural Networks 20, no. 3 (2009): 542-542.

[24] Denoeux, Thierry. "A k-nearest neighbor classification rule based on Dempster-Shafer theory." IEEE transactions on systems, man, and cybernetics 25, no. 5 (1995): 804-813.

[25] Suykens, Johan AK, and Joos Vandewalle, "Least squares support vector machine classifiers." Neural processing letters 9, no. 3 (1999): 293-300.

[26] McCallum, Andrew, and Kamal Nigam. "A comparison of event models for naive bayes text classification." In AAAI-98 workshop on learning for text categorization, vol. 752, no. 1, pp. 41-48. 1998.

[27] Dietterichl, Thomas G. "Ensemble learning." (2002).

[28] https://store.digilentinc.com/nexys-4-ddr-artix-7-fpga-trainer-board-recommended-for-ece-curriculum/

[29] Trusthub https://trust-hub.org/benchmarks/trojan

[30] https://www.cerc.utexas.edu/itc99-benchmarks/bench.html Accessed 23 Apr. 2019].

[31] Lyngaas, S. (2018). *NIST wants to the federal government to pay more attention to the supply chain.* [online] CyberScoop. Available at: https://www.cyberscoop.com/nist-supply-chain-risk-management-framework/ [Accessed 23 Apr. 2019].

[32] D. Agrawal et al., *Trojan Detection Using IC Fingerprinting*, Proc. IEEE Symp. Security and Privacy(SP07),IEEECSPress, 2007, pp. 296-310

[33] Li and J. Lach, *At-Speed Delay Characterization for IC Authentication and Trojan Horse Detection* Proc.IEEE Int'l Workshop Hardware-Oriented Security andTrust(HOST 08), IEEE CS Press, 2008, pp. 8-14.

[34] Ghohroud, N. and Hessabi, S. (2017). A Bio-Inspired Method for Hardware Trojan Detection. In: 19th International Symposium on Computer Architecture and Digital Systems. [online] IEEE. Available at: https://ieeexplore.ieee.org/document/8310672.

[35] Xuehui Zhang, Andrew Ferraiuolo, and Mohammad Tehranipoor. 2013. Detection of trojans using a combined ring oscillator network and off-chip transient power analysis. J. Emerg. Technol. Comput. Syst. 9, 3, Article 25 (October 2013), 20 pages. DOI=http://dx.doi.org/10.1145/2491677