Supervised and Unsupervised Machine Learning for Side-Channel based Trojan Detection

Dirmanto Jap
School of Physical and Mathematical Sciences,
Nanyang Technological University, Singapore
Email: djap@ntu.edu.sg

Wei He, Shivam Bhasin
Temasek Laboratories,
Nanyang Technological University, Singapore
Email: {he.wei, sbhasin}@ntu.edu.sg

***Abstract*—Hardware Trojan (HT) has recently drawn much attention in both industry and academia due to the global outsourcing trend in semiconductor manufacturing, where a malicious logic can be inserted into the security critical ICs at almost any stages. HT severity mainly stems from its low-cost and stealthy nature where the HT only functions at a strict condition to purposely alter the logic or physical behavior for leaking secrets. This fact makes HT detection very challenging in practice. In this paper, we propose a novel HT detection technique based on machine learning approach. The described solution is constructed over one-class SVM and is shown to be more robust compared to the template based detection techniques. An unsupervised approach is also applied in our solution for mitigating the golden model dependencies. To evaluate the solution, a practical HT design was inserted into an AES coprocessor implemented in a Xilinx FPGA. Based on the partial reconfiguration, the HT size can be dynamically changed without altering cipher part, which helps to precisely evaluate the HT influence. The experimental results have shown that our proposed detection technique achieve a high performance accuracy.**

I. Introduction

In modern days, the term Trojan is normally used to refer to Software Trojan, which is, as the name might imply, a malicious program that presents itself as harmless and tricks the innocent user to run or install it, and performs malicious actions on the user's device. In the context of hardware, with the growth of the semiconductor technologies, the chips are getting smaller and more complex, and also the resources required for the chip manufacturing are increasing. Hence, nowadays, most of the chip manufacturing tasks are being delegated or outsourced to other companies, rather than being done solely by the company that sells those chips. However, this opens up a new possibility for malicious parties to modify or tamper with the design of the chip, since it is hard to check and control each individual chip being manufactured in different foundries during the manufacturing process. Even if the device is supposed to be secure by design, a small alteration during the manufacturing process can easily compromise the integrity of the chip. This maliciously alteration introduces a new concept, which is now commonly referred as Hardware Trojan (HT) [1].

HT has received significant attentions from both industry and academia communities in the past decades, due to its critical threats imposed to security-sensitive hardware environments [2]. Once properly implemented, HT can be very hard to detect using conventional detection methods, because of its extremely tiny size compared to the infected design and the hibernation state during majority of its lifetime. Normally, HT will be inserted only on a small subset of the chip manufactured, which is only known to the adversary, and hence, distinguishing the HT infected chip might be a challenging task. A wide spectrum of approaches have been proposed for detecting the inserted HT from designs, which can be mainly classified into destructive and non-destructive methods, which can be further classified into invasive and non-invasive methods [3].

Most of the HT detection methods have been focused on the non-invasive methods [4]. They can be mainly classified into side-channel based approaches and logic (functional) testing based approaches. In side-channel based methods, the physical leakage will be observed, to see if there is any anomaly, with respect to the reference from the original chip or "golden" chip. This is a problem by itself because the golden chip might not always be available. Another problem is that the impact of the HT can be easily hidden within the process variation or random noise of the device, especially if the HT are inserted carefully. A skillful adversary could cleverly insert the HT such that any physical changes caused by the HT will be hidden within those random noises.

In logic testing based methods, random test vectors are used as inputs to the suspected chip in order to trigger the activation of the HT. One of the main concern is that the HT are not necessarily activated with the applied trigger pattern, so its application is quite restricted on trigger based HT. Also, the size of the patterns for testing tends to grow exponentially with increased input number. However, it has advantage in the sense that it is not affected by the process variation even if the HT size is small and carefully inserted.

Our Contributions: In this paper, we propose a new HT detection method based on the machine learning approach to exploit the anomaly in the side-channel leakage of the chip. We show that this method could achieve a high accuracy for the HT detection, even with the presence of significant noise. We also apply an unsupervised method to mitigate the analysis pitfalls with the golden chip.

The paper is structured as follows: In Section II, we describe the relevant HT detection methodologies that had been previously proposed in literatures. Next, the theoretical background

of machine learning approach and its practical incorporation to HT detection are described in Section III. In Section IV, the experimental setup and EM measurements are described. The detailed experimental results are presented in Section V. Finally, we provide some discussion and conclude the paper in Section VI.

II. Previous Works

The techniques of HT detection was first discussed by Agrawal *et al.* in [5], where a solution was proposed based on the generation of chip fingerprint of its power characteristics. This fingerprint was constructed by transforming the power traces measured from the *genuine* chip using Karhunen-Loève transform. Authors practically verified that the power fingerprints of the HT infected chips inevitably behave differently, as they carry additional leakages from the inserted HT circuit. Relying on the transformation, the differences could be magnified for observation. Different sizes of HT could then be detected using this method.

Kutzner *et al.* [6] evaluated the fingerprint based HT detection method on a fully functional lightweight HT embedded on PRESENT block cipher. They showed that in real measurements, due to noise and process variation, the HT could be undetectable by this method. Instead, they introduced a new detection approach, called Difference of Probability Distribution (DPD), which was also based on the difference in the side-channel characteristics. However, this solution exploits statistical properties of the probability distribution functions, built from side-channel measurements of an investigated IC. More specifically, they constructed the probability distribution from multiple time samples, over the encryption executions. Then, they calculated and combined the difference of cumulative distribution between each pair of tested designs. They showed that the difference between HT-infected and genuine design is significantly more observable as opposed to between genuine designs.

The original proposition [5] and the experimental verification [6] of the fingerprint based HT detection method were formalized by Ngo *et al.* [7]. In this work, they formulated the theoretical model for HT effects, to determine the false positive rate for the HT detection, taking into account the process variation and properties of the HT. It was assumed that the HTs will shift the average leakage of the chip, and thus, it allowed the estimation of the false positive (and negative) based on the difference of mean between the genuine and the HT infected device. This approach is similar to the Template Attacks [8], but there is only the genuine class, and the device will be deemed suspicious when its probability falls outside user defined boundary. By profiling the device, it is possible to distinguish a HT infected chip during the test. Hence, one possible question is whether it is feasible to use an alternative method to perform a more precise profiling of the chip. We consider the work of [7] as a reference technique.

A. Template based HT Detection

In previous work [7], the effects of HTs on the chip or device are modeled and formulated. The main assumption is that the side-channel leakage is following the (univariate or multivariate) normal distribution. Assuming all other experimental parameters stay constant, the side-channel leakage will still be sensitive to process variation or noise. This variation can modeled as a random noise following a normal distribution.

Under the normality assumption, when the adversary inserts the HT, it is usually small enough to avoid being detected by conventional methods. The HT can be denoted as the additional circuitries (it can also be defined using deletion or substitution of circuitries, but the main principle remains the same). The variation of the leakage caused by the HT is usually negligible relative to the process variation or noise and the main effect is the change in the leakage, which can be modeled as constant offset shift from the genuine circuit in the side-channel leakage (total leakage = HT leakage + genuine leakage + noise $\approx$ genuine leakage + noise).

Then, they used the Template based method for classification, by building the profiles using the genuine and infected classes. The profiling is done by computing the mean and the covariance of respected classes. However, for most of the time the profiling of the infected chips might not be possible since the mean for the HT infected chips is unknown. Hence, for classification of the device, the decision boundary is constructed based on user defined threshold around the mean of the genuine class, and the device needs to be classified as genuine or HT infected, if the leakage falls outside the threshold.

To tackle the problem with the noise or process variation, they introduced two methods to preprocess the traces: the sum of absolute difference and the threshold techniques. For the first approach, the sum of absolute difference, the mean of genuine traces from N FPGAs ($\vec{t_g} = \frac{1}{N}\sum_{i=1}^{N}\vec{t_i}$) is used for reference. And the sum of absolute difference will be performed for all traces ($S_i = \sum_{\#samples}|\vec{t_i} - \vec{t}|$). For the threshold technique, instead of directly summing up all features, a heuristic approach is first used to choose relevant features used for summation.

- First, the matrix $\mathbf{D}_g$ and matrix $\mathbf{D}_t$ are computed. The former defines the absolute difference of the genuine traces with their means, while the latter defines the absolute difference of tested traces with the mean of genuine trace.
- For each trace $\vec{t_i}$ from FPGA i, for each j-th feature, compare $D_{t_{i,j}}$ with max(D_{g_j}) (maximum value of $\mathbf{D}_g$ in column j).
- If $D_{t_{i,j}} < \max(D_{g_j})$, set $D_{t_{i,j}} = 0$, otherwise, keep the value.
- If the value of $D_{t_{i,j}}$ is 0 for all j, then FPGA i is classified as HT free.
- Otherwise, redo the sum of all difference method on the updated $\vec{D_{t_i}}$ and updated $\mathbf{D}_g$ (set all the columns in $\mathbf{D}_g$

to 0 where $\vec{D_{t_i}}$ are 0).

In this way, the irrelevant features can be discarded and hence removing one of the source of noise.

B. Challenges in HT Detection

Several practical problems still exist in prior HT detection techniques. Normally, most methods try to observe the physical anomaly of the hidden HTs in chip. In side-channel based methods, the techniques try to detect any suspicious side-channel activities. However, they will need a genuine or golden chip as a reference or fingerprint. Also, it is hard to distinguish HT related unusual activity from usual noise and process variation. More generally, it is not possible to develop a universal detection approach, since HTs are structured differently with varying physical characteristics.

III. Machine Learning for HT Detections

In this work, we investigate the detectability of the HT by observing the side-channel leakage (such as the electromagnetic emanation (EM)) and adopt a machine learning approach for single class classification, as an alternative tool for HT detection. The general concept of this learning algorithm is that it requires only the representation of the genuine data to construct a classification model, which is suitable for the HT detection, due to the unknown behavior of the HT.

Using the side-channel leakage of the genuine device for training, the profile can be constructed and used for determining whether the tested device is suspicious or not. Another possible scenario with this method is that the given data for training might already contain HTs. Hence in this case, the algorithm can be trained and tested using the data measured from the same set of devices, similar to the unsupervised learning concept, and return the prediction for each device, either genuine or infected. In this scenario, the method can be used for the case when there is no golden reference available.

A. Proposed Method based on One-class SVM

One-class Support Vector Machine (SVM) is a method which is based on SVM algorithm [9]. As the name implies, it constructs the classification model based on only one class, and build the separator hyperplane around the boundary of the training data. Once the model has been constructed by the algorithm, the new data will be verified using this model. Hence, if any new data is different from the training data, it will fall outside the boundary and could be considered as outliers, see Figure 1.

The idea of the method to separate the training data from the origin in some feature space. Intuitively, the origin acted as the representative of the other class, in order to simulate a binary classification problem, based of only one class. The separation is done by constructing a hyperplane as in SVM and the aim is to maximize the distance from the origin. Based on this construction, a region is formed by the hyperplane covering most of the training data. The classification is done as follow: if the data falls within the region, it returns +1 which means it might not be outlier, otherwise, it returns -1 which means that the data might be suspicious.

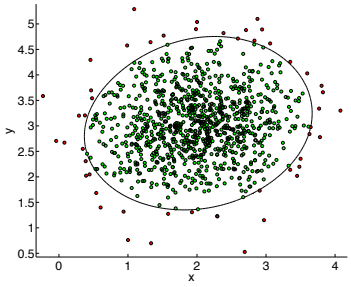


Fig. 1: Illustration of One-class SVM with RBF kernel.

In general, one class SVM is formulated by the following equation, which is similar to normal SVM:

$$\arg\min_{w,\xi,\rho} \frac{1}{2}\|\vec{w}\|^2 + \frac{1}{\nu n}\sum_{i=1}^{N}\xi_i - \rho \tag{1}$$

$$\text{subject to: } (\langle \vec{w}, \phi(\vec{t_i})\rangle) \geq \rho - \xi_i,\ \xi_i \geq 0\ \forall i,$$

The decision function is then formulated as follow (based on the same method to solve SVM):

$$f(\vec{t}) = \text{sgn}(\sum_{i=1}^{N}(\alpha_i)K(\vec{t_i}, \vec{t}) - \rho). \tag{2}$$

For one-class SVM, the parameter ν denotes the upper bound of the misclassification or margin error of the training data (rates of outliers assumed to be in the data) as well as the lower bound of number of training data that are used as support vectors. If the value of ν is set to be too small, it is assumed that the number of outliers are small and it could result is high false negative rate (outliers deemed as good or genuine data). On the other hand, if ν is set to be too large, most of the training data and thus large portion of the genuine data will be deemed as outliers, resulting in high false positive (false alarm) rate. Other parameters work the same, just like in SVM, and the kernel method still applies (see Table I).

This method can be used for the anomalies or outliers detection [11]. Thus, it can also be used for HT detection by modeling the HTs as outliers. The argument is that in the machine learning problem, it is easy to gather the training data, however, abnormal data or outliers might be expensive or impossible to collect. The reason is that it is hard to simulate all possible cases for the abnormality of the data. Thus, this can be a problem if the outliers detection problem is to be solved using the conventional classification methods. The single class machine learning methods is then used to

TABLE I: Commonly used kernels for SVM [10]

Kernel name	Kernel function
Linear	$K(t_i, t_j) = {t_i}^T t_j$
Radial basis function	$K(t_i, t_j) = \exp(\gamma\|t_i - t_j\|^2)$
Polynomial	$K(t_i, t_j) = (t_i \cdot t_j)^d$

generalize the distribution density of the training data. So rather than separating the training data from the outliers, the learning algorithm defines the structure of the training data. The concept described is similar to the HT detection problems. The training data used can be obtained from the verified genuine chip or golden model (supervised case), or the set of chip under test (unsupervised case).

B. Related Works

Previously, one-class SVM method had been used by Gustin *et al.* [12] to detect modification in the IP core. The power signature was constructed from the genuine IP, and the validation was done by comparing the generated data with the reference signature. It was used to distinguish the implementation of PRESENT lightweight cipher from other lightweight ciphers. However, in this case, the compared power signatures were based on different implementation of ciphers, and hence the difference might be easily observable compared to small modifications caused by the HT.

Later, Bao *et al.* [13] proposed a reverse engineering approach to identify HT-free IC using one-class SVM approach. They extracted several features from the high resolution image obtained using scanning electron microscope (SEM) on the exploited layer of the IC and used them to train the classifier. The constructed model was then used to clarify whether the IC is HT infected or not. However, this approach was invasive in nature. The technique proposed in this paper is non-invasive and thus it is more applicable to a wider spectrum of scenarios.

IV. Evaluation on FPGA

In our experiments, EM based analyses are performed on FPGA implemented cipher primitive with different HT sizes. The detailed HT implementation and EM measurement preparations on the exemplary FPGA will be described in this section.

A. HT Insertion on Partial Reconfiguration

To emulate the real-world device scenario, we rely on the `Partial Reconfiguration` technique on a Xilinx FPGA to insert a HT design into a post P&R AES cipher [14]. The circuit is separated into `static` and dynamic `Partial Reconfigurable (PR)` parts, where the dynamic part can be refreshed in real-time with a new bitstream, and the main static part stays intact. The advantage here is significant for our HT detection experiments, since we can ensure the absolutely identical cipher (static) with different HT (dynamic) sizes. So the variances only come from the HT circuits and hence permit fair evaluations of influence from the HT. The region constraint for dynamic and static part of the FPGA are applied at the early design stage and is shown in Figure 2.

The cipher algorithm is implemented on a Xilinx Spartan-6 FPGA, soldered on Sakura-G Board, similar to the setup described in [7], running at a frequency of 25 MHz.

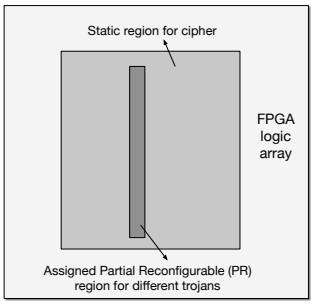


Fig. 2: HT inserted into the cipher using partial reconfiguration to ensure that the cipher is identical under different HT schemes.

B. Platform Setup and Cipher Implementation

The measurement platform is shown in Figure 3, where the EM radiation of the FPGA internal core from the decoupling capacitor (C40) is measured by a high-precision EM probe. The environmental noise from the surrounding on-board components and IO banks are shielded by an EM-proof cover, which helps to increase the signal-to-noise ratio (SNR) for our measurement.

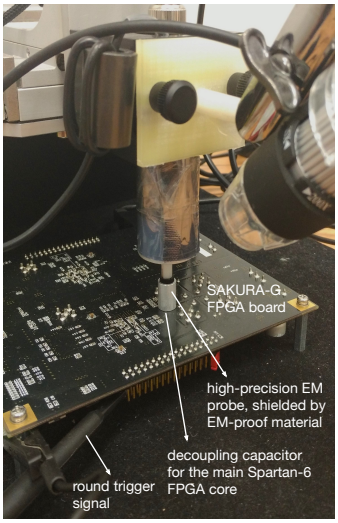


Fig. 3: EM measurement on decoupling capacitors of SAKURA-G FPGA board.

The insertion of the HT is similar to the one mentioned earlier which was highlighted in Figure 4. The HT [15] is used to inject a fault on the specific round to enable fault attacks on AES, though in our experiment, we do not trigger the HT (it remains dormant). Thus the presented method can be applied even if the HT is always dormant by minor leakage from the triggering circuit.

The HT is triggered only in second last round of AES, when N least significant bits (LSB) of 128 bits at the `AddRoundKey` at value 1. The payload of this HT is an exclusive-OR (XOR) gate. If the HT is triggered, it will inject fault at the 8-th round, to serve Piret's DFA [16]. We use

2 different versions of the HTs, varying the value of N (=8 and =128). The HT itself is placed within the boundary of AES crypto-processor. Here, the placement and routing in the original circuit are kept the same for both genuine circuit and HT circuit, as seen in Figure 5. We observe the EM activity of triggering circuit or payload which is enough to detect HT.

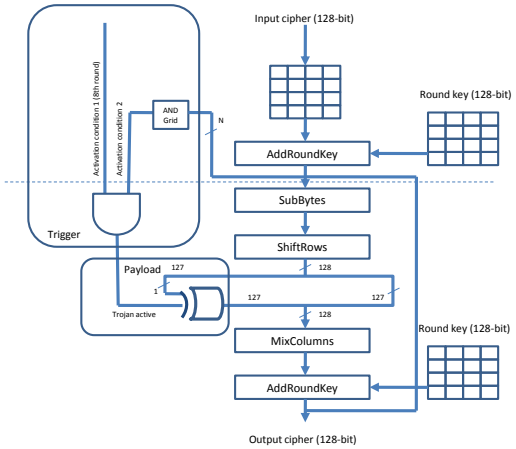
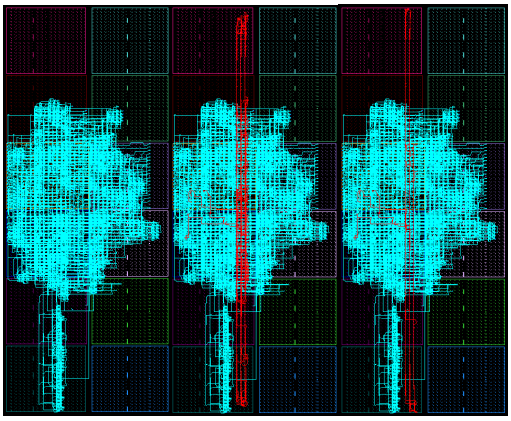


Fig. 4: Example of HT that induces fault on AES [15].

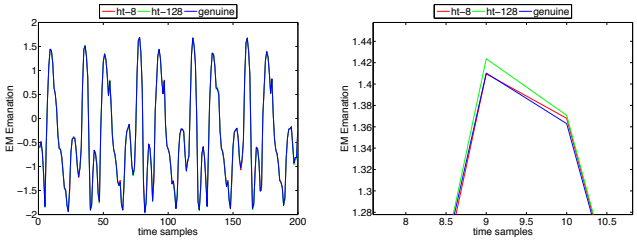
C. EM Measurement

We collect traces for 100 different plaintexts which are randomly chosen with uniform distribution. To reduce measurement noise, each trace is averaged 256 times before being saved for the experiments. The signal from the EM probe is recorded with WaveRunner 610Zi oscilloscope from LeCroy. Here, only the EM traces dropping within the interesting computation rounds are recorded (from sixth rounds onwards). However, we could argue that since the HT is not triggered, recording the traces over longer timing would not affect the overall performance.



(a) genuine design (b) 128-bit HT (c) 8-bit HT

Fig. 5: HT uninfected and infected AES implementations of different trojan size using Partial Reconfiguration in Spartan-6 FPGA.



(a) the EM trace (genuine and HT infected) (b) the EM traces (zoomed)

Fig. 6: EM trace (genuine and HT infected) with zoomed version.

Algorithm 1: One-class SVM for HT detection

Training phase:
  Record power or EM traces $\mathbf{T}_g$ from the genuine devices (supervised),
    or $\mathbf{T}$ from the tested devices (unsupervised) under the same plaintext.
  Perform feature selection method to obtain $\mathbf{T}'_g$ or $\mathbf{T}'$.
  Determine user defined parameter ν and γ.
  Construct the decision boundary f_g or f (Eq. 2) based on $\mathbf{T}'_g$ or $\mathbf{T}'$.
Testing phase:
  Record power or EM traces $\mathbf{T}_A$ from the tested devices with same plaintext for training. Perform feature selection to get $\mathbf{T}'_A$.
  Predict the temporary class labels using f_g or f on $\mathbf{T}'_A$.
  Repeat the previous steps with different plaintexts to obtain predictions.
  Perform majority voting from the collected prediction under different plaintexts to determine the final class labels.

In total, each trace has 500 sample points. We repeat the trace measurement 200 times from the genuine device, where 150 are used for training and remaining 50 for testing. For the circuit under test, 50 traces (averaged 256 times) each for same 100 plaintexts are collected. One sample of the EM trace is shown in Figure 6, where it can be seen that the HT impact is almost negligible (which might also be attributed to influence of the noise).

D. Preprocessing for HT Detection

In the supervised setting, we first split the genuine traces for training and testing. For training phase, we train the one-class SVM using different parameters on RBF kernel. We use the following sets of features when representing the data for training:

1) the sum of absolute difference [7];
2) the thresholding technique [7];

3) all the sample points in the traces;
4) the local peaks in the traces.

The first 2 methods have been aforementioned in Section II-A. The third one is basically to keep all the time sample points in the traces for training, since we assume that the impact of the HT might be hidden anywhere in the trace. The last feature selection method is done by selecting the sample points l_i in the traces with $|l_i| \geq [\mu_t + c \times \sigma_t]$ (μ_t is the mean of the trace, c, is predefined threshold, and σ_t is the standard deviation of the trace). It is similar to the idea of thresholding technique that the HT impact might be more observable in the peaks. The difference is that it selects multiple local peaks in the traces as features, only with respect to the training data, without the influence of testing data, and it does not sum all the selected points.

To improve the performance, we adopt the majority voting to determine the class, based on the classification results on different plaintexts. Here, for each plaintext, we obtain the predicted class for each trace. The process is repeated with the traces from different plaintexts. Then, based on the prediction from multiple plaintexts, majority voting will be done to determine the final prediction for the class. For comparison, we adopt the Template based approach, and same setting are used for classification. Different thresholds are used for determining the decision boundary for Template based attacks, and similarly, different learning parameters are used for construction of the boundary for one-class SVM. In Algorithm 1, we describe the procedure for conducting the experiments.

Normally, the learning parameters need to be tuned beforehand, by splitting the training data into training and validation data. The performance of different parameters will be checked by performing cross validation. Here, different learning parameters are used for training and tested with validation data, where parameters that lead to the best performing accuracy will be chosen. This is to ensure that there is no overfitting, where the algorithm captures the noise of the data, and makes a poor generalization, such that it fits the training data, but performs poorly on testing data. However, for the single class classification, this process might be difficult, since there is only one class, the genuine class. The validation data typically consist of genuine data and a small number of anomalous (HT) data for evaluating the performance. The performance is evaluated using the accuracy metric (ratio in which the predicted label match the actual label). The presence of the anomalous data in the validation set is generally done by generating the data artificially. However, sometimes, this might affect the performance of the learning algorithm by assuming some priori about the HT.

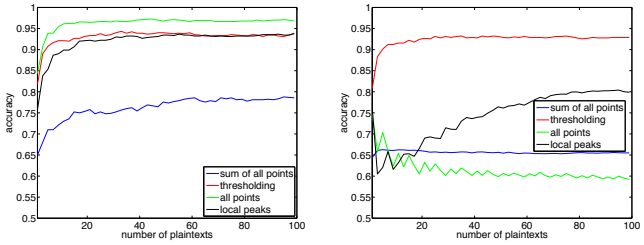
V. Experimental Results

In this section, the experiments with and without golden chip will be analyzed respectively. Some extended discussions will also be given for enhancing the merits of our contributions.

A. Experiment with Golden Chip

In this experiment, we assume that there exist a golden chip, on which the training phase will be conducted. We start the experiment with the larger HT ($N = 128$). In general, we check the performance accuracy of one-class SVM from different learning parameters in the validation phase, and based on the observation, we choose the better performing parameters (ν and γ for RBF kernel) for the testing phase.

During the testing phase, new sets of traces will be collected. The measurement is conducted from the set of suspected devices. Using the parameters obtained from the validation experiments, we conduct the experiments on the sets of testing data to check the performance accuracy. For the last feature selection method, we choose the value of v to be 1.5, which allows us to keep enough sample points for training. For the Template based detection method, in order to classify each trace, a parameter which is the user defined boundary or threshold needs to be defined when classifying each trace. We then perform similar training phase on the data (training and validation phase) to determine the boundary parameter. For each experiment, we then repeat it 50 times and take the average of the results. We then show the comparison of the results from the one-class SVM and the Template based attacks in Figure 7.



(a) one class SVM (b) template based

Fig. 7: Result on 128-bit HT - supervised scenario.

From the experiment, we can see that the one-class SVM approach is performing better than the Template based method. For both cases, the performance results for the threshold technique are quite similar for both methods. For one-class SVM approach, using all points can help increasing the accuracy, whereas in Template based method, adding more points or features will just add more noise. Also, for one-class SVM, the trend is that by increasing the number of plaintexts and adopting the majority voting approach, the accuracy results can be increased. The results when using only single traces and majority voting are compared and presented at Table. II.

In Table. III, we included the false positive and false negative rate for the experiments. We can see for most cases, that the false positive is higher than the false negative. In the context of security, this is somehow desirable, because, we want to minimize the probability of undetected HT. Since, the trend is more or less similar for all the experiments, and the overall result can be captured on the accuracy metric. For the

TABLE II: Accuracy results on 128-bit HT (supervised) with single data and majority voting

One-class SVM		
	1 trace	maj. voting
1	68.1%	83.6%
2	85.3%	93.4%
3	95.3%	99.4%
4	80.6%	95.0%

Template based		
	1 trace	maj. voting
1	66.4%	70.2%
2	83.8%	92.2%
3	74.7%	59.2%
4	76.2%	79.8%

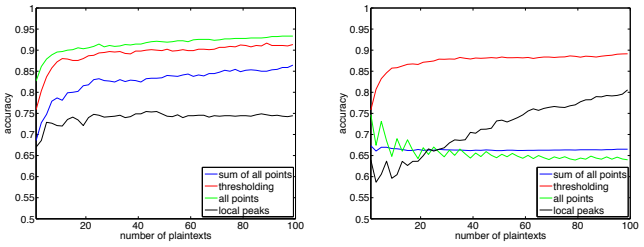
rest of experimental results, we only highlight the performance accuracy.

TABLE III: False positive and False negative on 128-bit HT (supervised) with majority voting

One-class SVM		
	FP	FN
1	14.4%	18.4%
2	13.1%	0.0%
3	1.1%	0.0%
4	10.0%	0.0%

Template based		
	FP	FN
1	6.0%	63.1%
2	14.2%	0%
3	81.5%	0%
4	40.2%	0%

To further verify the results, we investigate a smaller HT size with $N = 8$ in the next experiments. In Figure 8, we show the experimental results and similar trend can be observed, however, with lower accuracy. In this experiment, we show that when using the local peaks features, it fails to increase the performance accuracy, performing the worst, among different feature selection methods. Again, the threshold technique method performs similarly for both cases, and can achieve the best results for Template based attacks. For detailed comparison, the results are presented at Table. IV.



(a) one class SVM (b) template based

Fig. 8: Result on 8-bit HT - supervised scenario.

In general, from different experiments with varying HT sizes, we can see that one-class SVM is performing better than

TABLE IV: Accuracy results on 8-bit HT (supervised) with single data and majority voting

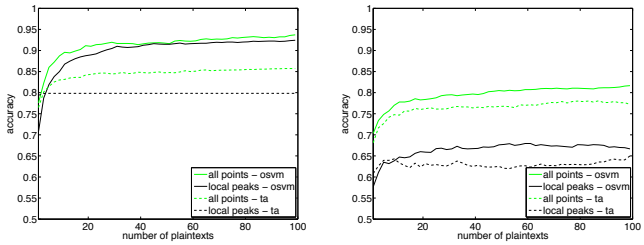
One-class SVM		
	1 trace	maj. voting
1	68.6%	86.4%
2	75.9%	91.3%
3	82.6%	93.3%
4	67.0%	74.4%

Template based		
	1 trace	maj. voting
1	67.3%	66.5%
2	75.8%	89.2%
3	74.6%	64.0%
4	63.9%	80.6%

the Template based approach. Also, different feature selection methods are more suitable for different methods, namely using all points for one-class SVM and thresholding for Template based. The main problem with the former is obviously when the number of samples grow, more training data are required. We also show that the majority voting based approach can improve the overall classification accuracy. By observing the results, for one-class SVM case, the performance with more plaintexts are better than using only a single plaintext. This is not true for the Template based case, as the results sometimes are getting worse when adding more information from more plaintexts.

B. Experiment without Golden Chip

For the unsupervised scenario, we also show that the one-class SVM can achieve good accuracy performance. In this case, we first collect the traces from the set of suspected devices and use all the traces for training. It is assumed that in the set of suspected devices, a few of the devices are infected with the HTs, and hence, the training data will contain some portions of the outliers. Then, for validation and testing, we collect a new set of data, from the same suspected devices, and use the data for testing. In Figure 9, we show the results from the experiments. The solid lines denote the one-class SVM approach and the dashed lines denote the Template based methods. The detailed results are presented at Table. V.



(a) 128-bit HT (b) 8-bit HT

Fig. 9: Result on unsupervised scenario.

For this experiments, since the mean of the genuine traces is unknown, we only consider two feature selection approaches, either using all points, or using only the local peaks based on heuristic methods. The experimental results show that using all points could lead to higher performance result. However, the performance of the heuristic method is getting worse with smaller size of HT as compared to larger HT. This might mean that for the smaller HT, the distinguishing points are not located in the peak of the traces, but rather during the period where the device is less active.

In this case, we can see that when the HT size is large, the accuracy of the Template based method is not as good as the one-class SVM method. This could be due to the reason that the distinguishability of the HT is better and by incorporating it on the training data, it disturbs the construction of the template. In one-class SVM based methods however, by adjusting the

TABLE V: Accuracy results on unsupervised with single data and majority voting

One-class SVM (128-bit)		
	1 trace	maj. voting
3	77.6%	93.7%
4	70.1%	92.4%
One-class SVM (8-bit)		
	1 trace	maj. voting
3	70.3%	81.6%
4	57.6%	66.7%

Template based (128-bit)		
	1 trace	maj. voting
3	74.6%	85.7%
4	79.4%	79.8%
Template based (8-bit)		
	1 trace	maj. voting
3	68.1%	77.2%
4	60.8%	65.0%

learning parameters, the additional noise from HT can be minimized. When the HT is smaller, the leakage is behaving more like genuine (smaller offset between the genuine and HT infected) and hence, Template based method could approach the performance of the one-class SVM.

VI. Discussions and Conclusions

In [7], the authors described a metric for detecting HTs using side-channel leakage under process variation, using Template-Like method for HT detection. In this paper, we proposed an alternative based on one-class SVM approach for detecting the HT. We adopt the single class learning algorithm, one-class SVM, which is commonly used for outlier detection. So rather than separating the training data from the outliers, the learning algorithm defines the structure of the training data. Based on this concept, using the side-channel leakage of the genuine device, supervised learning approach is adopted and used for determining if the tested device is suspicious or not.

In this method, we have modeled the HTs detection as outliers detection problem, and discuss some of the impact of the learning parameters on the performance of the algorithm, especially the ν parameter and the kernel parameter (γ). The choice of parameter ν can be determined by using cross validation methods. However, normally, during the validation phase, it is assumed that the data contain both HT (artificial or prior knowledge) and genuine data. Whereas, the HT data might not be available in practice. Hence, a more sophisticated parameter selection method could be used in the future.

Regarding the choice of features, we compare 4 different methods. For the two methods from the previous works [7], as they sum all the points, it adds more noise to the computation. However, the threshold technique still allows good classification for the Template based approach. For the next method, we use all the sample points, and the one-class SVM can extract the necessary information which allows good classification on the data. In the last method, we choose the local peaks in the traces, however, some information might not be in the peaks, which limits the accuracy of the prediction.

Another possible scenario with outliers detection is that the given data for training might already contain outliers. Hence in this case, the algorithm could be trained and tested using the data measured from the same device, using unsupervised learning, and return the prediction for each device, genuine or infected. Though the performance is not as good as the supervised scenario, it can still be an alternative, especially when dealing with the case where the golden chip is not available.

To improve the overall accuracy of the predictions, it is shown that by combining with majority voting method, the performance of the algorithm can be improved. Also, to further improve the success of HT detection, this method could also be integrated together with other state-of-the-art methods, such as using logic testing based approach (classical HT detection approach), or using some methods in side-channel analysis for increasing SNR in the traces.

Regarding the issues with noise and process variation, this method can still achieve the theoretical boundary shown in previous work. For the future work, we would like to test the proposed method based on different ICs, and a deeper investigation of the impact from noise and silicon process variation constitutes another part of the following work.

References

[1] DARPA, "TRUST in Integrated Circuits program," 2007.

[2] S. Adee, "The Hunt For The Kill Switch," *IEEE Spectr.*, vol. 45, no. 5, pp. 34–39, 2008.

[3] R. S. Chakraborty, S. Narasimhan, and S. Bhunia, "Hardware Trojan: Threats and Emerging Solutions," in *HLDVT*. IEEE, 2009, pp. 166–171.

[4] S. Narasimhan, D. Du, R. S. Chakraborty, S. Paul, F. G. Wolff, C. A. Papachristou, K. Roy, and S. Bhunia, "Hardware Trojan Detection by Multiple-Parameter Side-Channel Analysis," *Computers, IEEE Transactions on*, vol. 62, no. 11, 2013.

[5] D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi, and B. Sunar, "Trojan Detection using IC Fingerprinting," in *IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2007, pp. 296–310.

[6] S. Kutzner, A. Y. Poschmann, and M. Stöttinger, "Hardware Trojan Design and Detection: A Practical Evaluation," in *Proceedings of WESS*. New York, NY, USA: ACM, 2013, pp. 1:1–1:9.

[7] X. T. Ngo, Z. Najm, S. Bhasin, S. Guilley, and J.-L. Danger, "Method taking into account process dispersions to detect hardware trojan horse by side-channel," *Journal of Cryptographic Engineering*, pp. 1–9, 2016. [Online]. Available: http://dx.doi.org/10.1007/s13389-016-0129-2

[8] S. Chari, J. R. Rao, and P. Rohatgi, "Template Attacks," in *CHES*, ser. LNCS, vol. 2523. Springer, 2002, pp. 13–28.

[9] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt, "Support Vector Method for Novelty Detection," in *Advances in Neural Information Processing Systems 12, [NIPS Conference]*. The MIT Press, 1999, pp. 582–588.

[10] C. Cortes and V. Vapnik, "Support-Vector Networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[11] K. A. Heller, K. M. Svore, A. D. Keromytis, and S. J. Stolfo, "One Class Support Vector Machines for Detecting Anomalous Windows Registry Accesses," in *In Proc. of the workshop on Data Mining for Computer Security*, 2003.

[12] L. Gustin, F. Durvaux, S. Kerckhof, F.-X. Standaert, and M. Verleysen, "Support Vector Machines for Improved IP Detection with Soft Physical Hash Functions," in *COSADE*, 2014.

[13] C. Bao, D. Forte, and A. Srivastava, "On Application of One-class SVM to Reverse Engineering-based Hardware Trojan Detection," in *Fifteenth ISQED 2014*. IEEE, 2014, pp. 47–54.

[14] J. Daemen and V. Rijmen, *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer, 2002.

[15] S. Bhasin, J.-L. Danger, S. Guilley, X. T. Ngo, and L. Sauvage, "Hardware Trojan Horses in Cryptographic IP Cores," in *FDTC*, W. Fischer and J.-M. Schmidt, Eds. IEEE, 2013, pp. 15–29.

[16] G. Piret and J. Quisquater, "A Differential Fault Attack Technique against SPN Structures, with Application to the AES and KHAZAD," in *CHES 2003, 5th International Workshop*, ser. LNCS, vol. 2779. Springer, 2003, pp. 77–88.