HTnet: Transfer Learning for Golden Chip-Free Hardware Trojan Detection

Sina Faezi, Rozhin Yasaei, Mohammad Abdullah Al Faruque
Department of Electrical Engineering and Computer Science
University of California, Irvine, California, USA
(sfaezi,ryasaei,alfaruqu)@uci.edu

***Abstract*—Design and fabrication outsourcing has made integrated circuits (IC) vulnerable to malicious modifications by third parties known as hardware Trojans (HT). Over the last decade, the use of side-channel measurements for detecting the malicious manipulation of the ICs has been extensively studied. However, the suggested approaches often suffer from three major limitations: 1) reliance on a trusted identical chip (i.e. golden chip), 2) untraceable footprints of subtle hardware Trojans which remain inactive during the testing phase, and 3) the need to identify the best discriminative features that can be used for separating side-channel signals coming from HT-free and HT-infected circuits. To overcome these shortcomings, we propose a novel neural network design (i.e. HTNet) and a feature extractor training methodology that can be used for HT detection in run time. We create a library of known hardware Trojans and collect electromagnetic and power side-channel signals for each case and train HTnet to learn the best discriminative features based on this library. Then, in the test time we fine tune HTnet to learn the behavior of the particular chip under test. We use HTnet followed by an anomaly detection mechanism in run-time to monitor the chip behavior and report malicious activities in the side-channel signals. We evaluate our methodology using TrustHub [15] benchmarks and show that HTnet can extract a robust set of features that can be used for HT-detection purpose.**

***Index Terms*—hardware Trojan; neural networks; deep learning; transfer learning**

I. Introduction

Given the growing demand for low-cost integrated circuits (IC), companies tend to have their chips designed and fabricated by untrusted third-party entities over the globe. This has raised security concerns about intentional malicious modification of the integrated circuits, referred to as Hardware Trojans (HT).

The ICs are deployed in many sensitive applications, such as medical devices, critical defense technologies, and municipal support systems, which highlights the paramount importance of resolving this security issue. The attacker may tamper with the circuit in order to change the functionality, degrade performance, leak information, or deny service [16]. For instance, the malicious circuit manipulation in implantable medical devices can cause physical harm either by altering the functionality or through a Denial-of-Service attack. The HTs inserted in military devices, can leak sensitive data through side-channels and compromise the confidentiality of the information inside the chip such as encryption keys. In the last decade, HT detection has been deeply investigated in the literature; nevertheless, the proposed techniques have several shortcomings. The investigated techniques include destructive and non-destructive methods. Destructive methods have the IC decapsulated and delayered to obtain the images of physical layers of the chip and utilize reverse engineering techniques to detect malfunction in the circuit. Although this method is able to provide high levels of confidence, it is costly and time-consuming, and it can authenticate only a single chip which is often not usable after the invasive process. Thus, the non-destructive approaches, which rely on the HT impact on host circuit functionality or side-channel parameters, are typically better choice if proven to be effective. HTs are designed to have a small area and power consumption. They usually contain two primary part: trigger and payload. After a series of particular events in external or internal signals, the trigger activates the payload to perform malicious behavior. An inactive Trojan does not alter the functionality and its effect on chip parameters is not significant enough to be distinguished from process variation and measurement noise.

Side-channel signal analysis is another approach for HT detection based on the malicious circuit side-effects on the side-channel parametric profile of the chip. In this method, side-channel parameters are measured and compared to the expected values to identify the presence of additional structure. Most of the side-channel based works depend on a trusted chip, called golden chip, to create the reference model for comparison. However, in practice, the golden chip does not exist since the integrated circuits supply chain is susceptible to HT insertion in any stage [20].

Various works in the literature have studied golden chip-free approaches to address this limitation. In [7], the authors propose the idea of utilizing the Process Control Monitors (PCM) and statistical side-channel fingerprinting to construct the reference for chip verification. It is assumed that the PCM are reliable since manipulating them is very difficult for attackers and any systematic modification of them results in notable deviation. Another approach introduced in [9] is self-referencing. When the HT is triggered, it creates uncorrelated temporal variation in the transient current signature of the chip which is detectable by side-channel measurement.

Due to the stealthy nature of HTs, they may evade detection in the IC testing phase with drastic consequences that are not acceptable in critical application. Thereafter, the idea of run-time HT detection is discussed to act as the last line of defense. The chip behavior is monitored during run-time to guarantee proper and secure IC operation. For instance, the

978-3-9819263-5-4/DATE21/©2021 EDAA

method described in [3] provides an early alert for triggered HTs by monitoring performance counters. The advancement of semiconductor technology has brought about the issue of deterioration of circuit profile over time, known as circuit aging. This phenomenon can cause the run-time HT detection frameworks to fail since a predefined constant model cannot address the impact of the circuit aging on the circuit parameters and it is not considered in most approaches.

In this paper, we develop a novel neural network design (i.e. HTNet) and a training methodology for HT detection in run time without the golden-chip requirement. We create a library of known HTs and collect electromagnetic (EM) and power side-channel signals for each case and train HTnet to learn the best discriminative features based on this library. Then, during the testing phase, we fine tune the HTnet to learn the behavior of the particular chip under test. We use HTnet followed by an anomaly detection algorithm in run-time to monitor the side-channel signals emitted from the chip and report malicious behaviors related to a triggered HT.

A. Research Challenges

In a nutshell, the state-of-art approaches for HT detection often experience the following challenges:

- Reliance on a reference golden chip, which in practice is costly to obtain or unavailable in some HT threat models such as untrusted IPs.
- Detection of HTs often requires activating them while the probability of triggering an HT during testing is very low due to their stealthy nature.
- For each new IC design, a new feature space needs to be defined to extract the most relevant features in the side-channel signals that can be used for detecting HTs.

B. Technical Contributions

This paper makes the following technical contributions to address the aforementioned challenges:

- We devise a custom neural network architecture that performs the same or better than any available approach in the literature for HT detection.
- We propose a methodology to transfer the knowledge learned based on known HT benchmarks to a new circuit which may or may not contain an HT.
- With this work, we publish our collected electromagnetic and power side-channel data for hardware implementation of AES infected by various forms of HTs. We provide our source codes for our model implementation.[1]

II. Background

A. Related Works

The potential threat of HT has motivated many studies where side-channel analysis is a widely used strategy for HT detection. The idea is to measure side-channel parameters including supply power leakage [1], [11], [12], [19], transient current [9], delay [6], [21], temperature [5], or EM emission [17], [24] and determine if the IC is contaminated by comparing

[1]https://github.uci.edu/AICPS/HTnet

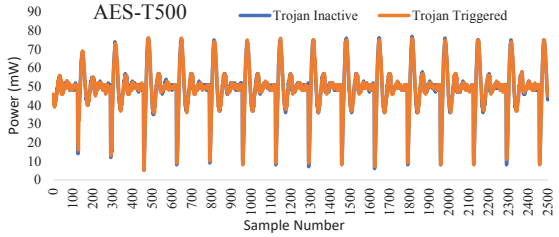

Fig. 1. Two power time series samples for a benchmark.

the measurements with the expected reference values. Although in some side-channel analysis approaches, golden chip-free methods such as temporal self-referencing [9] are utilized for HT detection, they mainly depend on availability of a golden chip. To overcome this shortcoming, reference-free methods are introduced which rely on the other characteristic of the chip. The authors of [14] use controllability and observability analysis of gate-level netlist along with unsupervised clustering to provide the HT related signal lists and assert the insertion of the HT. In [13], the authors propose that they used deep learning to extract Trojan features from gate-level netlist and detect infected nodes by clustering the nodes. The approach explained in [24] modifies the fill cells in a standard cell library to be observable in an optical image taken through the backside of the chip. Using the optical image, the circuit layout is extracted and used malicious manipulation detection.

B. Time Series Classification and Anomaly detection

1) Feature Engineering Based Models: As depicted in Figure 1, distinguishing between an inactive HT and a triggered HT signal is not a trivial task as the differences are very small. The majority of the machine learning approaches used in the literature such as Support Vector Machines (SVM) define a distance function for comparing the similarity/dissimilarity between two signals. Given that a side-channel signal with T samples is defined as $X = [x_1, x_2, ..., x_T]$, the distance functions are typically in the form of L_p norm defined as,

$$L_p = ||X - X'||_p = (\sum_i |x_i - x'_i|^p)^{1/p} \quad (1)$$

where $p = 1$ and $p = 2$ denote Manhattan distance and Euclidean distance, respectively. However, in practice, the length of the two signals might be different, small shifts in the signal may occur, and it is computationally expensive to determine the L_p value for large values of T. To overcome these challenges, various techniques have proposed using a set of features for representing the time series signal. Time domain features such as local min/max/average, frequency domain features extracted by Short-Time Fourier Transform (STFT), and features extracted by Discreet Wavelet Transform (DWT) have been among the most widely used features for the time series representation. Besides the spectral approaches, representative eigenvalue analysis methods such as Principle Component Analysis (PCA) [8] and Singular Value Decomposition (SVD) have been widely used in the literature to compress the signal and focus on the most relevant features in the signal. Often, some of these features are pruned with human supervision to select the best set of features. In most cases, the accuracy of the models heavily rely on the chosen features.

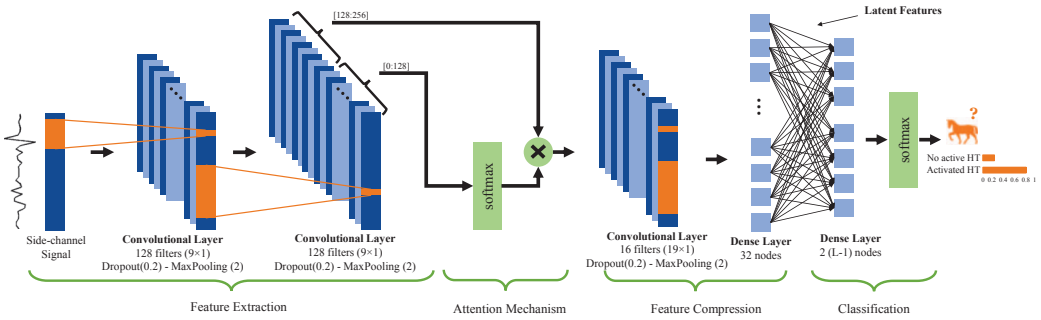

Fig. 2. HTnet structure when used as classifier side-channel signals.

2) End-to-end Models: In recent years, deep learning has gained incredible popularity. In particular, Convolutional Neural Networks (CNN) have been widely used in various applications due to their capability of extracting the most relevant features under most scenarios automatically. The details of CNN formulation for time series classification is provided in [22]. Under this formulation, each convolutional layer acts as a filter with trainable weights that can adjust to the input shape during the training phase. In contrast to the other methods, neither training nor using CNN based models does not involve a separate feature-engineering step; for this reason they are referred as end-to-end models.

III. Methodology

In this section, we describe our methodology for designing and training a neural network to extract the most relevant features from side-channel signals in a compact form. Then, we use a self-referencing approach to detect anomalous behavior.

A. Library Construction

The first step in our methodology is to gather a dataset of side-channel signals. We use TrustHub [18] benchmarks that are implemented in the FPGA platform as the reference circuits. For each benchmark, we collect N EM and power traces under two scenarios: HT-inactive and HT-triggered. In the HT-inactive scenario, the HT is disabled to stay inactive during data collection. In the HT-triggered scenario, we enforce the trigger condition and the HT is always active. The side-channel signal dataset collected for the j^{th} benchmark is defined as $D_j = \{(X_1^j, Y_1^j), (X_2^j, Y_2^j), ..., (X_{2N}^j, Y_{2N}^j)\}$ which is a collection of pairs (X_i^j, Y_i^j) where X_i^j is a univariate time series signal resulted from concatenation of various side-channel signals (EM and power for this paper) and $Y_i^j \in \{0, 1\}$ as it is corresponding label (0 - HT-inactive, 1 - HT-triggered).

B. HTnet Model design

To design a virtually perfect neural network structure for the detection of triggered HTs, we perform an exhaustive search on the available state-of-the-art architectures for CNNs to find the best design for classifying the collected signal into Trojan triggered and inactive classes. Figure 2 represents HTnet that is the result of this exhaustive search with fine-tuned hyperparameters. HTnet consists of two Convolutional Layers (CL) with ReLU activations, each immediately followed by a dropout and max-pooling mechanisms to avoid overfitting. The output of these CLs is a preliminary discriminative set of features from the input signal. To further purify the extracted features, we implement an attention mechanism after CLs. We divide the output of the second CL into half, apply a softmax on half of the features and multiply it to the other half. Afterward, a CL with a small number of filters followed by a dense layer with a small number of nodes is added to the network to compress the extracted features. For the rest of this paper, these compressed features will be referred to as latent features. In the end, there is one more dense layer with a softmax activation function for the classification purpose.

C. Modeling Side-channel Footprints of Known HTs

In this paper, we propose a transfer learning approach to avoid the need for a golden chip. Inspired by [10], we first train the HTnet H to learn our previous knowledge about the existing HTs. Assuming that we are going to evaluate our methodology over the m^{th} benchmark, we train HTnet to perform a $2(L-1)$ classification task over R_m defined as,

$$R_m = \bigcup_{i=0}^{i=L} D_i - D_m \tag{2}$$

where L is the total number of benchmarks (each benchmark has inactive and triggered samples). During training, the weights in H are tuned such that it extracts a set of descriptive latent features that can be used to discriminate between previously known HT-inactive and HT-triggered samples. We consider the weights of H as our initial knowledge for previously known HT-infected circuits.

D. Knowledge Transfer for One-class Feature Learning

Once an IC is manufactured, we can only acquire unlabeled side-channel signals from the chip. In the case of HTs with a trigger mechanism, it is often assumed that they will not get triggered during the chip test phase otherwise they would be detected. This means that during this phase we can assume that we have access to a number of side-channel signals that are mostly likely labeled as HT-inactive. Based on this assumption, we construct a model that learns to extract features for only one class of available data which is HT-inactive during the chip testing phase.

1) Architecture: As shown in Figure 3, we discard the last dense layer in H, call it H', and create two new models: a reference model H_r and a target model H_t. H_r consists of H' followed by a new softmax dense layer with $2(L-1)$ nodes. H_t consists of only H' which shares the same weights with H' part of the reference model and is responsible for one-class feature extraction. To train H_r and H_t we use a reference dataset $R'_m \subset R_m$, and a target dataset $D'_m \subset \{(X, Y) : (X, Y) \in D_m$ and $Y \neq 1\}$ accordingly.

2) Loss functions: The extracted features by the network should be both compact and descriptive. A compact feature space will result in a similar feature representation for samples of our target class (HT-inactive - D'_m). A descriptive feature space will produce distinct representations for samples of different classes (other benchmarks, HT-triggered, and HT-inactive -R'_m). To ensure these two qualities, we incorporate two loss functions: compactness loss (l_C) and descriptiveness loss (l_D). We formulate, our optimization objective to be,

$$\hat{g} = \min_g l_D(R'_m) + \lambda l_C(D'_m) \tag{3}$$

where λ is a constant. Here, l_C computes the mean squared intra-batch distance within a given batch of target class samples. Define $X = \{x_1, x_2, ..., x_n\} \in R^{n \times k}$ to be the input to the loss function, where n is number of samples in the batch and k is length of each sample. The distance of i^{th} sample with other samples can be defined as,

$$z_i = x_i - mean_i \tag{4}$$

where $mean_i = \frac{1}{n-1}\sum_{j \neq i} x_j$ is the mean of the rest of samples in the batch. From there, compactness loss is defined as average Euclidean distance as in,

$$l_C = \frac{1}{nk}\sum_{i=1}^{n} z_i^T z_i. \tag{5}$$

We calculate l_C over target network (H_t) outputs. To perform back-propagation using this loss function on this network, it is necessary to assess the contribution of each element over the final loss. Thereby, let $x_i = \{x_{i1}, x_{i2}, ..., x_{ik}\}$ and $m_i = \{m_{i1}, m_{i2}, ..., m_{ik}\}$. Then, the gradient of l_C with respect to the input x_{ij} can be calculated as,

$$\frac{\partial l_C}{\partial x_{ij}} = \frac{2}{(n-1)nk}[n \times (x_{ij} - m_{ij}) - \sum_{k=1}^{n}(x_{ik} - m_{ik})]. \tag{6}$$

For descriptiveness loss l_D we use classical cross-entropy formulation of multiclass classification tasks with regard to the reference dataset and calculate it over the reference network.

3) Training: Initially, both H_r and H_t contain the pre-trained model H weights. During training, the first two CLs in H' are frozen as commonly done for network fine-tuning and only later layers in the network can be trained. Furthermore, a relatively lower learning rate is chosen for the model training process to avoid sudden deviations from the initially trained model. During every iteration of training, in the forward pass, two sample batches each from the target dataset (D'_m) and reference dataset (R'_m) are feed to the H_t and H_r respectively to calculate l_C and l_D. Since the weights W are shared between the two networks, the composite loss can be defined as,

$$l(R'_m, D'_m) = l_D(R'_m|W) + \lambda l_C(D'_m|W). \tag{7}$$

During training, we back-propagate over two networks and learn the network parameters to minimize this composite loss function which leads to the minimization of the optimization objective in 3.

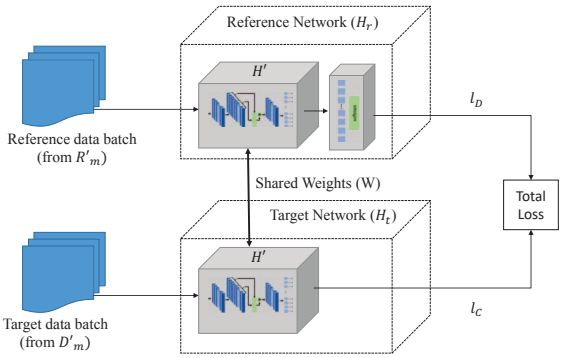

Fig. 3. Training reference and target networks.

E. HT detection through self-referencing

In this paper, our major contribution is constructing a model that can extract the most useful features from side-channel signals for HT detection. Once these features are extracted, any self-referencing anomaly detection approach can detect HTs with high accuracy based on these features. Here, we use One-class Support Vector machines (OC-SVM), K Nearest Neighbors (KNN), and Local Outlier Factor (LOF) as candidates for anomaly detection in the side-channel signal. These models are trained during the testing phase using our feature extractor and used in run time for HT detection.

IV. Results

A. Experimental Test Bed

We build an automated testbed (Figure 4) to measure the EM and power side-channel signals of various circuits. The device under test is a Sakura-G Board on which two Spartan-6 FPGA are integrated. One controller FPGA handles the USB connection with computer for data transmission and the main FPGA is used to implement the target circuit which is AES-128 encryption core along with a variety of hardware Trojan circuits in our tests. The board provides the power consumption of target circuit as an output voltage that is measured by the Tektronix TDS2022C oscilloscope. We use H-field probe to measure and amplify the magnetic emissions of FPGA. Then, the Agilent Spectrum Analyzer N1996A reads the signal. The computers sends random test vectors to AES core and collect the power and EM data from the devices. The measurements are synchronized by a trigger from board that determine the start of encryption process of each input.

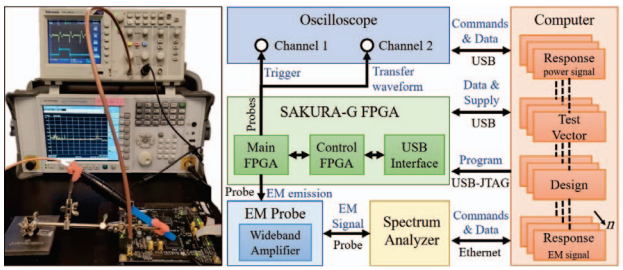

Fig. 4. The automated testbed for side-channel signals collection.

TABLE I
HTNET COMPARISON WITH THE METHODS IN [4] TO CLASSIFY POWER SIDE-CHANNEL SIGNALS.

Benchmark	SVM	Naive Bayes	Random Forest	HTnet
AES-T400	0.6646	0.7400	0.7012	0.8975
AES-T500	0.9967	0.9971	0.9900	0.9793
AES-T600	0.6738	0.7321	0.7175	1
AES-T700	1	1	1	1
AES-T800	1	1	1	1
AES-T1000	0.6800	0.7213	0.6925	0.7135
AES-T1100	0.6783	0.7371	0.6929	0.7758
AES-T1300	0.6746	0.9833	0.9721	1
AES-T1400	0.6738	0.7392	0.6704	0.8525
AES-T1600	0.6629	0.7421	0.7008	0.7570
AES-T1800	0.6596	0.7379	0.7204	0.9345
AES-T2000	0.6575	0.7267	0.7225	0.9295
Mean	0.7518	0.8214	0.7984	0.9033

B. Evaluation Method

For each benchmark, we collect 1000 HT-inactive and active samples ($N = 1000$) and utilize 80% of data for training the feature extractors and use the rest of the data to perform the tests (validation) in rest of this paper. We incorporate same number of HT-inactive and HT active samples for our evaluations. Hence, we simply report our results in terms of accuracy defined as,

$$Accuracy = \frac{N_{TP} + N_{TN}}{N_{TP} + N_{TN} + N_{FP} + N_{FN}} \quad (8)$$

where N_{TP}, N_{TN}, N_{FP} and N_{FN} are number of true positive, true negative, false positive and false negative accordingly. A *positive* side-channel signal segment means that an HT is triggered, and a *true* indication means that the detection method has correctly identified the type of that segment.

C. HTnet Evaluation

Given that we have access to both HT-inactive and HT active samples for each benchmark, we first evaluate the performance of HTnet to classify these two types of samples for each benchmark. Note that in this scenario, access to a golden chip is guaranteed and the purpose of this evaluation is to compare HTnet to the other methods available in the literature. Table I compares the performance of HTnet to the methods proposed in one of the latest works in the literature [4] (2020). This table shows the accuracy of each method using only power side-channel signals. To improve the accuracy of each of the rival methods, we first extract and select the best set of features using [2] and report their best performance. The results show that HTnet outperforms any of these methods over AES benchmarks even when the best type of features are extracted for them.

D. Golden Chip-Free Feature Extraction Evaluation

We set the value of λ to 0.1 and follow the training mechanism described in Section III. We use Stochastic Gradient Descent (SGD) optimizer with learning rate of 0.001 for training H and learning rate of 0.000001 for H_r and H_t. Figure 5 shows the outputs of H_t for some of the HT-inactive and the HT-triggered samples. As shown in this figure, each sample input (EM concatenated with power) is converted to a compact form of 32 features. In this feature space, all the HT-inactive samples posses a similar set of features while multiple features

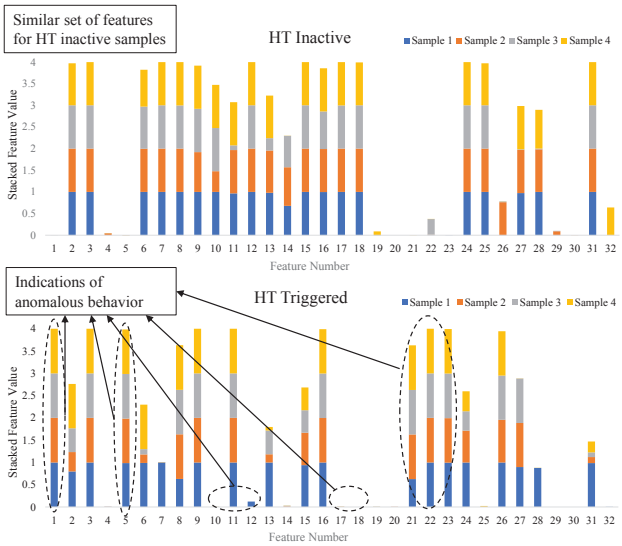

Fig. 5. Extracted features for four HT-inactive and HT-triggered samples from AES-500 benchmark.

of HT-triggered samples vary from the HT-inactive samples. These variations can be used for reporting the sample as a HT-triggered case using an anomaly detection mechanism.

E. HT Detection Evaluation

As described in Section III-E, we incorporate various off-the-shelf anomaly detection mechanisms from [23] to evaluate the effectiveness of our extracted set of features from EM and power side-channel signals for HT detection. As shown in Table II, our proposed methodology for extracting relevant features can increase the accuracy of anomaly detection methods in most of the cases. Particularly, LOF and OC-SVM cannot detect any anomalous behavior in side-channel signals without using our feature extraction mechanism for AES-1100 benchmark. Note that collecting EM side-channel signals highly depends on the probe placement over chip. Thereby, here, we only provide evaluations for benchmarks that we were able to collect reliable EM traces. As shown in II, EM and power side-channel signals fusion can boost the accuracy of our golden chip-free approach to pass the results presented in Table I in some cases.

F. Robustness of Extracted Features

The side-channel signals tend to vary over the life span of an IC due to various reasons, such as aging and temperature change. To evaluate the robustness of our extracted features, we add a synthetic white noise $\mathcal{N}(0, \sigma^2)$ to the scaled (0 to 1) test samples. Figure 6 shows the changes in the accuracy of

TABLE II
ACCURACY OF VARIOUS ANOMALY DETECTION ALGORITHMS OVER CONCATENATED EM AND POWER SIDE-CHANNEL SIGNALS.

	KNN		LOF		OC-SVM	
Model Input	HTnet	Raw	HTnet	Raw	HTnet	Raw
AES-T500	0.936	0.895	0.928	0.858	0.928	0.919
AES-T700	0.986	0.927	0.975	0.901	0.972	0.946
AES-T800	0.983	0.937	0.961	0.941	0.981	0.916
AES-T1000	0.972	0.914	0.975	0.953	0.983	0.962
AES-T1100	0.803	0.798	0.656	0.491	0.822	0.478
Mean	**0.936**	0.894	**0.899**	0.829	**0.937**	0.844

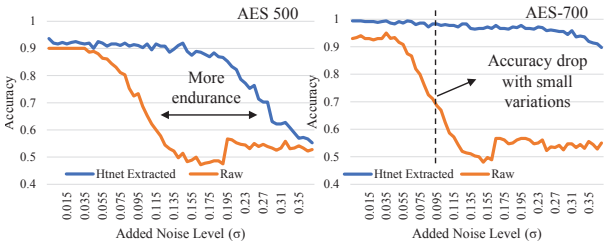

Fig. 6. Robustness evaluation of KNN with two different input types.

KNN when it is using HTnet extracted features versus when it is directly monitoring the input signal in two benchmarks. The results show that small variations in the side-channel signal can significantly affect a trained anomaly detector accuracy if it does not use our approach for feature extraction. In other terms, an HTnet enabled anomaly detection method can endure more noise in the collected side-channel signals during run time (more than two times resiliency for these two benchmarks).

V. Discussion

As a last line of defense, run-time HT detection approaches play a crucial role in detecting subtle HTs which are not detected in the verification phase. In this paper, we provide evidence that it is possible to extract robust features that can be used for HT detection using our previous knowledge about HTs. However, running a neural network along with the main circuit will consume considerable amount of energy which will make our approach mostly suitable for applications which do not rely on limited power resources. Furthermore, the self-referencing approach that we discussed in this paper is a naive approach which can be significantly expanded using the other works discussed in the related literature. Here, we provided a proof of concept for golden chip-free HT detection using recent developments in artificial intelligence. We leave the details of efficient implementation of HTnet and using more advanced self-referencing approaches to our future work.

VI. Conclusion

In this paper, we introduce HTnet which outperforms any existing approaches for detecting triggered HTs through side-channel signals given that a reference golden chip is available. Furthermore, we introduce a novel training methodology to utilize our previous knowledge about known HT-infected circuits to design a robust HT-related feature extractor. We show that it is possible to use off-the-shelf anomaly detection algorithms and incorporate HTnet as a feature extractor to detect triggered HTs in run time with 93.7% accuracy on average.

VII. Acknowledgment

This research was supported by the Office of Naval Research (ONR), award number N00014-17-1-2499. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect views of the funding agency.

References

[1] Alkabani, Y., and Koushanfar, F. Consistency-based characterization for ic trojan detection. In *Proceedings of the 2009 International Conference on Computer-Aided Design* (2009), ACM, pp. 123–127.

[2] Christ, M., Kempa-Liehr, A. W., and Feindt, M. Distributed and parallel time series feature extraction for industrial big data applications. *arXiv preprint arXiv:1610.07717* (2016).

[3] Elnaggar, R., Chakrabarty, K., and Tahoori, M. B. Run-time hardware trojan detection using performance counters. In *2017 IEEE International Test Conference (ITC)* (2017), IEEE, pp. 1–10.

[4] Gayatri, R., Gayatri, Y., Mitra, C., Mekala, S., and Priyatharishini, M. System level hardware trojan detection using side-channel power analysis and machine learning. In *2020 5th International Conference on Communication and Electronics Systems (ICCES)* (2020).

[5] Hu, K., Nowroz, A. N., Reda, S., and Koushanfar, F. High-sensitivity hardware trojan detection using multimodal characterization. In *Proceedings of the Conference on Design, Automation and Test in Europe* (2013), EDA Consortium, pp. 1271–1276.

[6] Jin, Y., and Makris, Y. Hardware trojan detection using path delay fingerprint. In *2008 IEEE International workshop on hardware-oriented security and trust* (2008), IEEE, pp. 51–57.

[7] Liu, Y., Huang, K., and Makris, Y. Hardware trojan detection through golden chip-free statistical side-channel fingerprinting. In *Proceedings of the 51st Annual Design Automation Conference* (2014), ACM.

[8] Liu, Y., Jin, Y., Nosratinia, A., and Makris, Y. Silicon demonstration of hardware trojan design and detection in wireless cryptographic ics. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems 25*, 4 (2016), 1506–1519.

[9] Narasimhan, S., Wang, X., Du, D., Chakraborty, R. S., and Bhunia, S. Tesr: A robust temporal self-referencing approach for hardware trojan detection. In *2011 IEEE International Symposium on Hardware-Oriented Security and Trust* (2011), IEEE, pp. 71–74.

[10] Perera, P., and Patel, V. M. Learning deep features for one-class classification. *IEEE Transactions on Image Processing 28*, 11 (2019).

[11] Potkonjak, M., Nahapetian, A., Nelson, M., and Massey, T. Hardware trojan horse detection using gate-level characterization. In *Proceedings of the 46th Annual Design Automation Conference* (2009), ACM, pp. 688–693.

[12] Rad, R. M., Wang, X., Tehranipoor, M., and Plusquellic, J. Power supply signal calibration techniques for improving detection resolution to hardware trojans. In *Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design* (2008), IEEE Press, pp. 632–639.

[13] Reshma, K., Priyatharishini, M., and Devi, M. N. Hardware trojan detection using deep learning technique. In *Soft Computing and Signal Processing*. Springer, 2019, pp. 671–680.

[14] Salmani, H. Cotd: Reference-free hardware trojan detection and recovery based on controllability and observability in gate-level netlist. *IEEE Transactions on Information Forensics and Security 12*, 2 (2017).

[15] Salmani, H., Tehranipoor, M., and Karri, R. On design vulnerability analysis and trust benchmarks development. In *2013 IEEE 31st international conference on computer design (ICCD)* (2013), IEEE.

[16] Shakya, B., He, T., Salmani, H., Forte, D., Bhunia, S., and Tehranipoor, M. Benchmarking of hardware trojans and maliciously affected circuits. *Journal of Hardware and Systems Security 1*, 1 (2017).

[17] Stellari, F., Song, P., and Ainspan, H. A. Functional block extraction for hardware security detection using time-integrated and time-resolved emission measurements. In *2014 IEEE 32nd VLSI Test Symposium (VTS)* (2014), IEEE, pp. 1–6.

[18] Tehranipoor, M., Karri, R., Koushanfar, F., and Potkonjak, M. Trusthub. *Available on-line: https://www.trust-hub.org* (2016).

[19] Wei, S., and Potkonjak, M. Scalable hardware trojan diagnosis. *IEEE Transactions on very large scale integration (VLSI) systems 20*, 6 (2012).

[20] Xiao, K., Forte, D., Jin, Y., Karri, R., Bhunia, S., and Tehranipoor, M. Hardware trojans: Lessons learned after one decade of research. *ACM Transactions on Design Automation of Electronic Systems (TODAES) 22*, 1 (2016), 6.

[21] Xiao, K., Zhang, X., and Tehranipoor, M. A clock sweeping technique for detecting hardware trojans impacting circuits delay. *IEEE Design & Test 30*, 2 (2013), 26–34.

[22] Zhao, B., Lu, H., Chen, S., Liu, J., and Wu, D. Convolutional neural networks for time series classification. *Journal of Systems Engineering and Electronics 28*, 1 (2017), 162–169.

[23] Zhao, Y., Nasrullah, Z., and Li, Z. Pyod: A python toolbox for scalable outlier detection. *Journal of Machine Learning Research* (2019).

[24] Zhou, B., Adato, R., Zangeneh, M., Yang, T., Uyar, A., Goldberg, B., Unlu, S., and Joshi, A. Detecting hardware trojans using backside optical imaging of embedded watermarks. In *Proceedings of the 52nd Annual Design Automation Conference* (2015), ACM, p. 111.