Towards AI-Enabled Hardware Security: Challenges and Opportunities

Hossein Sayadi[1], Mehrdad Aliasgari[1], Furkan Aydin[2], Seetal Potluri[2], Aydin Aysu[2], Jack Edmonds[3], Sara Tehranipoor[4]

[1]Department of Computer Engineering and Computer Science, California State University, Long Beach, CA, USA
[2]Department of Electrical and Computer Engineering, North Carolina State University, NC, USA
[3]Department of Electrical and Computer Engineering, Santa Clara University, CA, USA
[4]Department of Computer Science and Electrical Engineering, West Virginia University, WV, USA

Abstract—**Recent developments in Artificial Intelligence (AI) and Machine Learning (ML), driven by a substantial increase in the size of data in emerging computing systems, have led into successful applications of such intelligent techniques in various disciplines including security. Traditionally, integrity of data has been protected with various security protocols at the software level with the underlying hardware assumed to be secure. This assumption however is no longer true with an increasing number of attacks reported on the hardware. The emergence of new security threats (e.g., malware, side-channel attacks, etc.) requires patching/updating the software-based solutions that needs a vast amount of memory and hardware resources. Therefore, the security should be delegated to the underlying hardware, building a bottom-up solution for securing computing devices rather than treating it as an afterthought. This paper highlights the growing role of AI/ML techniques in hardware and architecture security field and provides insightful discussions on pressing challenges, opportunities, and future directions of designing accurate and efficient machine learning-based attacks and defense mechanisms in response to emerging hardware security vulnerabilities in modern computer systems and next generation of cryptosystems.**

I. Introduction

For the past decades, cybersecurity has been at the forefront of worldwide attention as a serious threat to the infrastructures of information technology. Attackers are more driven and equipped to compromise computational infrastructure, including software and hardware systems. Recent developments have demonstrated that attackers are able to breach systems and carry out destructive actions by exploiting newly discovered hardware vulnerabilities. Particularly, a computer system's security can be jeopardized at the hardware level by a variety of attacks, such as running malicious programs (a.k.a. malware) to infect the target host and/or deploying Side-Channel Attacks (SCAs) to deduce the consent of users to share sensitive information [1], [2], [3], [4], [5], [6], [7]. Malware and SCAs are now important dangers to system security due to the information technology industry's rapid growth.

The prevalence of emerging computing platforms in the embedded systems and Internet-of-Things (IoT) industries has amplified the negative impact of new security vulnerabilities [8]. Essentially, the proliferation of new malware and side-channel attacks at the hardware level necessitates patching or updating software-based malware detection solutions (such as commercial anti-virus software), which is not practical for emerging computing systems, especially in embedded mobile and IoT devices [9]. The modern embedded systems, which cover a wide range of applications, are commonly resource-constrained, making it difficult to deploy the traditional software-based techniques used for detecting and containing malware in general-purpose computing systems [10], [11]. Moreover, most of these advanced analysis techniques are architecture-dependent i.e., dependent on the underlying hardware. Hence, this makes existing traditional software-based detection techniques difficult to apply effectively to emerging embedded computing devices.

The complexity of security attacks changes as quickly as innovation and the expansion of information technology infrastructure, creating a never-ending arms race between security defenses and attackers. A significant rise in the volume of data in new computing systems has spurred recent advances in Artificial Intelligence (AI), Machine Learning (ML), and Deep Learning (DL) which have resulted in effective applications of these intelligent approaches in a variety of fields, including security. A growing body of academic, commercial, and governmental research is being done on AI-enabled hardware security, which spans a variety of computing domains including mobile platforms, embedded systems, IoT devices, and high performance systems. Due to its capacity to keep up with evolving threats, an important part of current research has concentrated on the development and deployment of machine learning approaches for hardware security.

In this paper, we present an in-depth analysis of recent trends in AI-enabled hardware (cyber)security methods and their challenges and opportunities, and further provide insights on the future direction of this research area. In Section II, we present recent trends and advances on application of machine learning for hardware-assisted intrusion detection, and further highlight the essential challenges and opportunities in the field. Next, in Section III we present an overview of DL-enabled side-channel attacks analysis in next-generation cryptosystems along with its challenges and potential opportunities. Furthermore, Section IV provides implementation results of machine learning-based side-channel attacks on AES-128 where we present our findings for evaluation of the Advanced Encryption Standard, particularly unmasked and masked AES-128 implemented in software on microcontroller units and unmasked AES-128 implemented in hardware on field-programmable gate arrays (FPGAs), against machine learning-based side-channel analysis. We expect that our work provides the foundations for and facilitates future studies on exploring the application of AI/ML techniques to cope with increasingly complex cyber-attacks at the computer systems' hardware level in various application domains.

II. Intelligent Hardware-Assisted Intrusion Detection: A Retrospective and Look Ahead

In traditional intrusion detection techniques such as anti-virus tools scanning and analyzing take a considerable amount of time degrading the performance and speed of the computer

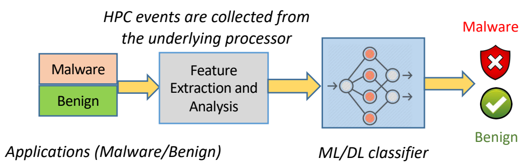
HPC events are collected from the underlying processor
Malware
Benign
Feature Extraction and Analysis
Malware
Benign
Applications (Malware/Benign)
ML/DL classifier

Fig. 1: Process of hardware-assisted malware detection using ML/DL.

system. In addition, the analysis of malware samples and the construction of rules/patterns for recognizing the unknown malware is often error-prone, making the manual heuristic analysis of applications limited [12]. Behavior-based detection also increases the number of false positives, blocking harmless programs from working and restricting the normal operation of the system [13], [14]. Therefore, to overcome the drawbacks of conventional intrusion detection methods, researchers have shifted their attention towards benefiting the power of AI/ML techniques and realizing the potential of hardware-related information for distinguishing malicious programs from benign.

A. Hardware Performance Counters for Security

Performance Monitoring Unit (PMU), a logical part of modern microprocessors, is in charge of tracking and measuring numerous performance-related events that are derived from Hardware Performance Counter (HPC) registers. HPCs are essentially a group of special-purpose registers included into contemporary microprocessors to record the trace of hardware-related events like executed instructions, cache misses, or incorrectly anticipated branching for an active program [1], [4], [15]. HPCs are programmed to issue an interrupt when a counter overflows or even be set to start the counter from the desired value. HPCs are easily programmable and found their ways in various processor platforms from high-performance to low power embedded processors, IoT, and biomedical devices.

As a result, hardware-assisted intrusion detection using underlying hardware related information (e.g., application logs) and ML/DL techniques has emerged as a promising solution for improving upon the software-based solutions and reducing the latency of detection process with small cost and no hardware redesign effort [1], [4], [16]. Figure 1 shows the general process of employing ML/DL applied on low-level hardware events for distinguishing intrusion (e.g., malware) from benign applications. Hardware detectors offer fast real-time detection, efficient resource utilization, and invulnerability from getting infected by attackers, which make them suitable for mitigating newer threats. Machine learning-based detectors using HPC data can be implemented in microprocessor hardware with significantly low overhead as compared to the software-based methods, as detection inside the hardware is very fast (few clock cycles). Such methods will not only allow to capture the malcode pattern more rapidly, it may also be feasible to deploy the computing system on end hosts [4], [11].

B. Brief Summary of Prior Works

The work in [1] used supervised ML classifiers on collected performance counters traces of both malware and benign programs and demonstrated that the running applications can be categorized with high level of accuracy. The authors illustrated the suitability of employing HPC information in detecting malware at the Linux OS level such as Linux rootkits and cache side-channel attacks on Intel and ARM processors. In a different study, Tang et al. [17] further discussed the feasibility of unsupervised learning that employs low-level HPCs features for detecting return-oriented programming (ROP) and buffer overflow attacks by finding anomalies in HPC information.

Authors in [15] is a recent work on HMD that deploys machine learning algorithms applied on synthetic traces of HPC features for detection of kernel-level rootkit attacks. The paper achieved high prediction accuracy in detecting five self-developed synthetic rootkits models. The research in [4] proposed ensemble learning techniques based on AdaBoost and Bagging to facilitate run-time hardware-assisted malware detection and improved the performance of HMD by accounting for the impact of reducing the number of HPC features on the performance of malware detectors to address the challenge of limited availability of on-chip HPC registers. In addition, a recent work in [16] proposed a two-stage machine learning-based approach for run-time malware detection based on variety of ML classifiers to detect and classify the malware signature with high accuracy and efficiency.

The deployment of ML algorithms are further offered in recent studies to accurately and efficiently detect the signature of emerging microarchitectural side-channel attacks [7], [18], [19], [20], [21]. ML-based microarchitectural SCAs detectors that are also based on low-level events captured from processors' HPC registers have considered collecting hardware events of victim applications (cryptographic application, e.g. RSA, AES and etc.) and attack applications to build a large dataset and apply ML/DL to build the detection system.

C. Application of ML for Hardware Intrusion Detection

Machine learning-based countermeasures often includes different stages such as monitoring the application to profile the HPC data, feature analysis, training, and testing the ML-based detector using the collected features. The ML models trained by low-level microarchitectural features continuously learn by analyzing the HPCs data to identify the malicious patterns and protect the processor architecture against possible malware and side-channel attacks.

1) Feature Selection: Analysis of Key Events: Identifying the prominent low-level features is an important step for developing accurate ML-based countermeasures [4], [16], [22]. There exists numerous microarchitectural events with different functionality available to collect from running programs in modern microprocessors. Counting all possible features would result in a high dimensional data which increases computational complexity and induces delay. Moreover, including irrelevant features could reduce the accuracy of classifiers [23], [24]. Two research questions are raised by this. Which low-level features should be used first in order to identify and categorize a certain type of malicious attack? Second, how can data collection be feature reduced to save unneeded computing overheads? Therefore, a minimal collection of HPCs that can accurately reflect the malware or side-channel attack application behavior is determined in order to execute an efficient efficient ML-based security countermeasure to identify security threats with low overhead. For effective run-time detection in resource-limited systems (e.g. embedded devices) which have limited number of HPC registers that can be concurrently captured at run-time, feature selection even plays a more important role in determining the minimal set of critical HPCs to collect the required data in a single run [4], [9], [15], [16]. There have been different feature selection techniques that are dominant in prior ML-based detection work such as correlation attribute evaluation, principle component analysis (PCA), information gain ratio (IGR), and Fisher Score. Selected HPC features are then used to train each ML-based detector in which the classifier attempts to find a correlation between the feature values and the application behavior to predict the existence of malicious patterns (benign or attack).

2) Performance and Efficiency Evaluation: There are several metrics that may be used in machine learning and statistics to assess a prediction model's performance in order to demonstrate its robustness and accuracy. Table II provides a summary of the standard evaluation metrics used for performance analysis of ML/DL-based security countermeasures utilized for detection and classification activities.

In addition, recently Convolutional and Deep Neural Network models have provided to be state-of-the-art across many applications. However, CNNs and DNNs are characterized by their large size which is not suitable for resource-constrained embedded and IoT devices. There is a trade-off between the accuracy and overhead of ML techniques and developers must take it into account when they intend to integrate a ML-based detector into the recourse-constrained devices, such as mobile devices, IoT sensors, biomedical and low-power wearable devices [11], [22]. Essentially, as highlighted in recent works such as [4], [16], [22] hardware implementation overhead, particularly area, latency (delay), and power consumption are as critical as accuracy in the performance analysis of ML-based security countermeasures. This is because ML classifiers are chosen in real time based on cost efficiency, an especially important consideration due to the limited resources of the microprocessors found in today's computing systems.

3) ML Techniques for Malware and SCAs Detection: To categorize the unknown applications into either benign or malicious software, the classification process can be divided into two stages including training and testing. First, we need to construct the classification model by training the ML classifiers using the extracted data (the HPCs information) for malware detection. The extracted features are then converted to vectors in the training set. Both the feature vectors and the class label of each sample (i.e., malicious or benign) are used as inputs for a classification algorithm .

By analyzing the training file samples, the deployed ML classification algorithm constructs a classifier capable of detecting the patterns of malicious samples with some level of accuracy and performance detection. Next, during the testing stage, first the vectors of the new file samples are first extracted using the same feature extraction techniques as in the training phase. This unseen data then is fed to the trained classifier to examine the detection rate of malware detection process. The classifier attempts to classify the new file samples based on the extracted feature vectors. Table I shows the ML classification techniques where we briefly describe the machine learning classification models that are typically used for hardware-assisted intrusion detection.

D. Performance Monitoring Tools

In order to monitor applications behavior and collect hardware-related events that assist in application performance analysis and tuning various performance monitoring tools have been used in prior works. These tools include Perf [25], Pin [26], PAPI [27], Intel VTune [28], and Intel PCM [29]. All these tools are available for Linux systems while only Intel VTune and Intel PCM are able to monitor HPCs in Windows and macOS systems. Perf, PAPI, and Pin demand some knowledge of command lines for users due to the lack of GUI interface. Perf tool is a Linux-based low-level performance monitoring tool that can instrument CPU performance counters, tracepoints, kprobes, and uprobes (dynamic tracing) [30]. Its monitoring granularity scales as least as $10ms$ without customization. Pin tool collects various program's ISA-dependent features such as instruction mix, instruction-level parallelism, register traffic, branch predictability, etc. to examine the applications behavior [26].

Performance Application Programming Interface (PAPI) [31] provides a cross-platform interface for monitoring hardware performance counters on processors that are equipped with specific registers for hardware events. To help with discovering and resolving performance bottlenecks in running programs for tweaking and debugging purposes, Intel has developed a licensed-based tool called Vtune [28]. It can record and show performance-related information. It offers a robust GUI interface and supports a wide range of profiling, including HPCs, call graphs, performance bottlenecks, and hotspot hunting, in comparison to the previous tool. Last but not least, PCM [29], [32] is the performance monitoring units (PMU) implemented in Intel's processors (e.g., Xeon, Atom, and Xeon Phi) that help to monitor performance and energy-related metrics in both Windows and Linux environments. Compared to Perf and PAPI tools, Intel PCM supports both core and uncore events monitoring in real-time.

E. Challenges and Opportunities

In this section, we determine several research challenges of hardware-assisted malware and side-channel attacks detection using ML/DL algorithms.

1) Detection of Stealthy Malware: Stealthy attack is a type of cyber security attack in which the malicious code is hidden inside the benign application for performing harmful purposes [33]. In real-world scenarios malware can also be embedded in a benign application which makes the detection task more challenging. Recent works [34], [35] have shown that that embedded malware (a category of stealthy malware) is not detected by standard ML models; hence, proposed a time-series deep learning-based technique to detect embedded malware Given the importance of this area, there is still a critical research demand for artificial intelligent based methods for detecting various sophisticated stealthy cyber attacks including advanced persistent threats (APT) in various computing domains including high-performance, IoT, biomedical, and cloud.

2) Clear Architectural Implications of HPC Events: Despite the wide application of HPC registers for security, obtaining a deep knowledge of micro-architectural features interactions with the malware behavior, which is an essential step for various types of security analysis, is a challenging problem that makes the potential gains quickly diminish. Hence, a serious concern to delve into in the area of AI-enabled hardware intrusion detection would be lack of proper explanation about the reasons that a specific set of features enables to properly characterize the malicious traits of samples.

3) Validation of HPCs for Security Analysis: The challenges and limitations of using hardware performance counters registers to train AI/ML models for security inferences need to be considered. This is due to a reason that security analysis was not the original intent for designing such registers in modern CPUs. While modern microprocessors have continued to increase in the peak performance and scalability, accurate

TABLE I: Frequently used ML models for hardware-assisted malware and SCAs detection

ML Classifier	Description	Example
Bayesian Network (BN)	Probabilistic graphical model that aims to model conditional dependence and causation, by representing a set of variables and conditional dependencies by edges in a directed graph.	BayesNet (BN), NaiveBayes (NB)
Neural Network (NN)	Consists of units (neurons), arranged in layers, which convert an input vector into some output. Each unit takes an input, applies a (often nonlinear) function to it, and then passes the output on to the next layer.	Multi-Layer Perceptron (MLP)
Decision Tree (DT)	Sequential models, known as "*divide and conquer*" algorithms, which logically combine a sequence of simple tests where a numerical attribute is compared against a threshold value or against a set of possible values.	REPTree (RT), J48, PART
Rule-Based	ML models that identify, learn, and evolve a set of relational rules that collectively represent the knowledge captured by the system.	OneR, JRip
Linear Regression	A statistical model used to determine the extent to which there is a linear relationship between a dependent variable and one or more independent variables.	simple linear regression, multiple linear regression
Logistic Regression (LR)	A statistical method in which its goal is to find the best fitting model to describe the relationship between dependent variable (response or outcome variable) and a set of independent (predictor or explanatory) variables.	Simple Logistic (SL), Multinomial Logistic Regression (MLR)
Ensemble Learning	A branch of ML which is used to improve the accuracy and performance of general ML classifiers by generating a set of base learners and combining their outputs for final decision.	Boosting, Bagging, Random Forest (RF)
Boosting	One of the most commonly used ensemble learning in which each base classifier is trained on a weighted form of the training set in which the weights depend on the performance of the previous base classifier.	AdaBoost
K Nearest Neighbor (KNN)	A supervised classification algorithm that takes labeled points and uses them to learn how to classify other points by looking at the labeled points closest to that new point (nearest neighbors), and has those neighbors vote (the "k" is the number of neighbors it checks).	KNN (k= 1,2,...),
Bagging	An ensemble meta-estimator where a value like a mean is estimated from multiple random samples of training data which are drawn with replacement. Each ML model is exploited to make a prediction and the results are averaged to give a more robust and generalized prediction.	Bagging/Bootstrap Aggregation

TABLE II: Performance metrics for analyzing ML/DL models

Performance Metric	Description
True Positive (TP)	Correct positive prediction
False Positive (FP)	Incorrect positive prediction
True Negative (TN)	Correct negative prediction
False Negative (FN)	Incorrect negative prediction
Specificity: True Negative Rate	$TNR = TN/(TN + FP)$
False Positive Rate	$FPR = FP/(FP + TN)$
Precision	$P = TP/(FP + TP)$
Recall: True Positive Rate	$TPR = TP/(TP + FN)$
F measure (F score)	$Fmeasure = 2 \times (P \times R)/(P + R)$
Detection Accuracy	$ACC = (TP + TN)/(TP + FP + TN + FN)$
Error Rate	$ERR = (FP + FN)/(P + N)$
Area Under the Curve	$AUC = \int_0^1 TPR(x)dx = \int_0^1 P(A > \tau(x))dx$

performance monitoring remains an important issue as an area of dissatisfaction between researchers and designers who are interested in delivering a more realistic realization of the interactions of hardware and software. As low-level features frequently have ambiguous documentation and indefinite inference, an expert level of perception is necessary to accurately execute the required applications (benign and malicious) on the target processor, and capture the information from HPC registers for further analysis [36], [37]. Hence, the researchers studying in this domain are urged to attempt to validate and verify their extracted hardware performance counter-assisted information on different microprocessor architectures.

4) Assessing Offensive Machine Learning: Despite wide applications in various domains, some recent researches substantiated that ML/DL brings new security challenges itself, such as an adversarial attacks [38], [39], [40]. In particular, he exposed vulnerabilities have indicated that the outcome of ML classifiers can be modified or controlled by adding specially crafted perturbations to the input data. There exists a few studies on adversarial learning for hardware-based intrusion detection. For instance, the work in [41] presented an adversarial attack on the ML-based intrusion detection systems to evade the security mechanism by injecting the perturbations in the HPC traces by using an adversarial sample generator application. On the defense side, the research in [42] proposed a moving target defense (MTD)-based method in response to the basic adversarial attack by developing multiple ML classifiers trained on different sets of HPCs. Due to emergence of these types of adversaries and vulnerabilities in ML/DL-based detectors along with ever-growing complexity of attacks and wide adoption of such mechanisms in different domains (e.g., embedded, IoT, biomedical devices, etc.) it is necessary to explore more extensible, generalized, and adaptable adversary-resilient learning-based countermeasures to address this important challenge across various application domains.

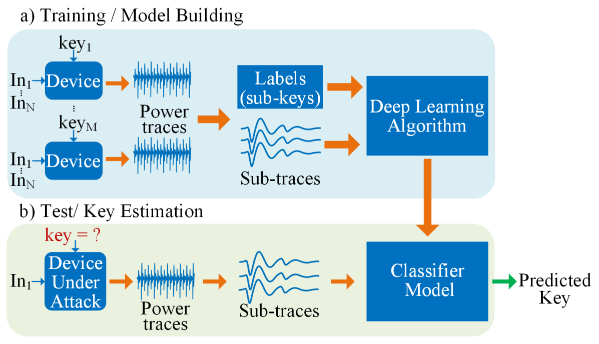

Fig. 2: Deep Learning-based side-channel analysis illustrating a) training and b) testing phases [47].

III. Learning-Based Side-Channel Attacks

Side-channel attacks are superior to classical cryptanalysis because of the additional leakages of the computing device while interacting with the physical environment and their strong correlation to the secret key. The possible side-channels include timing, power [43], electromagnetic radiation [44], temperature [45], or even sound [46]. Broadly speaking, SCAs can be categorized as *profiling-based* and *non-profiling based*. The most powerful amongst them is the class of *profiling-based* side-channel attacks.

Profiling-based attacks use two steps: the adversary first procures a copy of the target device and uses it to create *templates* that characterize the physical leakage for different key combinations; and subsequently, it performs a key recovery using *template* matching and analysis on the target device. However, it is computationally impossible to exhaustively create a dictionary of all *templates* due to the sheer complexity of the key search space. As a result, although *profiling-based* attacks are superior to their *non-profiling based* SCA counterparts, they can suffer a loss in accuracy with an increase in the cipher size. This motivates the development of more advanced techniques that can intelligently prune the search space and improve key decryption accuracy.

The idea of creating *templates* matches the idea of creating labels during supervised classification, thus motivating researchers to explore the powerful ML or Deep Learning (DL) techniques for *profiled-based* SCA. Figure 2 outlines the method of such profiled attacks. The adversary profiles the device using different keys and inputs during the training phase and builds a DL-based classifier. The adversary can then use this classifier in the field to estimate the secret key by correctly

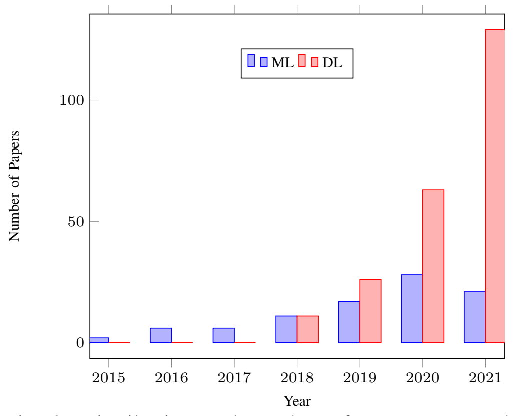

Fig. 3: Distribution and number of papers on machine learning and deep learning-based side-channel analysis. The Cambrian explosion of learning-based approaches is also visible in the context of side-channel analysis.

labeling traces with an unknown key.

In recent years, there has been tremendous growth in the creation, analysis, and successful deployment of a wide variety of powerful DL algorithms on various challenging problems including image recognition, natural language processing, DNA sequencing, etc. [48]. This has impacted SCA research too—indeed, there is a significant body of existing work [49] that has the utilization of a rich collection of supervised and unsupervised learning algorithms. Figure 3 quantifies these numbers based on our findings on Google Scholar and further splits them into ML vs. DL in terms of the learning techniques being used.

The key advantages of ML or DL-based approaches over the conventional techniques are threefold. First, learning-based techniques can automatically explore and apply suitable pre-processing techniques on obtained side-channel traces—whether these techniques are "good enough" is obviously a separate discussion which we will discuss below. But such pre-processing techniques can be useful if there is a mismatch between obtained/profiled traces vs. the test traces that can happen either naturally, due to aging, voltage or temperature variations, or due to an employed defense. Second, the target device may not be the same as the profiled devices. In such so-called cross-device attacks, ML can automatically generalize at a higher dimension if the learning process is capable. Third, and arguably most importantly, there are far fewer side-channel experts compared to ML/DL experts. Therefore, ML/DL-based attacks allow re-purposing the existing ML infrastructures and expertise towards SCA.

A. *Brief Summary of Prior Works*

A survey of published works on ML-based SCA prior to the application of DL is already available in prior work [49]. We will summarize some key papers here and refer interested readers to the extended survey paper for a more thorough analysis. Maghrebi et al. proposed the first application of DL in general and convolutional neural networks (CNNs) in particular for profiled SCA [50]. Earlier papers used multilayer perceptrons (MLPs), but since they do not report the number of hidden layers or this number is set to one, those works are generally not considered DL-based SCA in the hardware security community [51]. The first application of the *non-profiling* DL-based SCA is proposed in [52]. In [53], the authors proposed for the first time the application of reinforcement learning to

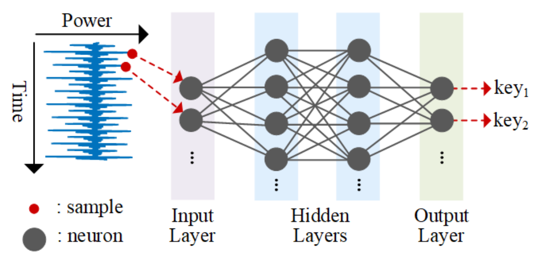

Fig. 4: Deep Learning-based classifier (Deep Neural Network) taking samples of power trace as inputs to the input-layer and providing the label-probabilities as outputs of the output-layer [67].

select model hyperparameters for DL-based SCA. Carbone et al. were the first to consider DL-based *profiling* SCA [54] in the context of public-key implementations. The first works in the direction of SCA and AI/DNN explainability aimed at interpreting neural network decisions by using heatmapping techniques [55], and the first step in this direction uses the Singular Vector Canonical Correlation Analysis tool to explain what neural networks learn while training on different side-channel datasets [56].

B. *DL-based SCAs*

Traditional machine learning approaches such as support vector machines and random forests have advantages over DPA and TA [57], [58]. However, with the improvements in available compute power, there has been a surge in using DL-based SCA techniques to attack cryptographic devices. Figure 2 outlines the method of such profiled attacks. During the training phase, the adversary profiles the device using different keys and inputs and builds a DL-based classifier. The adversary can then use this classifier in the field to estimate the secret key. DL techniques manage to filter and align traces automatically, which was normally dealt with by specialized methods known only to side-channel experts. Previous work has shown how different DL techniques can be used to extend and enhance SCA [59], [60], [61], [62], [63], [64], [65].

1) *Neural Network Classification:*

The objective of a classifier is to predict one of K discrete-classes $\mathcal{C}_k$, where $k = 1, 2, \dots, K$, given some input vector $\mathbf{x}$. Figure 4 shows how a DL-based classifier can be used by an adversary to determine the secret key, where $\mathbf{x}$ corresponds to power traces and $\mathcal{C}_k$ refers to sub-keys, which are used to create a DL model to attack the device.

Neural networks (NNs) consist of multiple layers, namely an *input layer*, an *output layer*, and *hidden layers*. The class of NNs with multiple hidden layers are referred to as deep neural networks (DNNs). The different layers consist of neurons with activation functions, which tend to be the rectified linear unit (*ReLU*) to capture the non-linearity [66].

The dimensions of the input layer vary with the dimension of the input data, e.g., the number of time steps in a power trace. The output layer consists of K neurons, one each for the K discrete prediction classes. The number of hidden layers and the neurons per hidden layer varies based on the task that the model has to perform and are considered hyper-parameters. For instance, to capture complex features for image recognition tasks, we might want to increase the hidden layers and neurons in them. However, increasing the layers and neurons may lead to over-fitting on the training set and thus we need to strike a careful balance.

C. Challenges and Opportunities in AI-Enabled SCAs

We summarize the key challenges and the opportunities for the use of learning techniques in the context of SCA.

1) Challenge #1: Curse of Dimensionality: Although ML-based techniques have been proposed in the past, the high dimensional data makes it exponentially more difficult to learn a classifier that generalizes well on unseen examples, a challenge that is also known as the curse of dimensionality [68]. The curse of dimensionality limits the practical applicability of many linear regressive and classification-based ML models. Advanced ML classifiers like SVMs reduce the dimension by first defining basis functions that are centered on the training data points and then selecting a subset of these during a non-linear optimized training. However, the number of basis functions in SVMs is often still relatively large and typically increases with the size of the training set [68]. Neural networks solve this challenge by using a fixed number of *parametric* basis functions, where the *parameters* are adapted during training. Unlike SVMs, NNs involve nonconvex optimization during training, which is costly but worth investing in during the training phase to obtain a compact model that is fast at processing new data [68].

The initial works in DL-based SCA [50] have used MLP but it has multiple limitations including high guessing entropy [69] and high accuracy degradation (up to 74.8%) in cross-device settings [70]. This has led to the development and application of more powerful DL techniques to SCA, including supervised ones like CNNs, semi-supervised ones like recurrent neural networks (RNNs), and unsupervised ones like long short term networks (LSTMs).

2) Challenge #2: Extending Beyond AES: The large majority of the existing learning-based attacks have focused on the AES. While AES is an important target, it is definitely not the only one. Moreover, AES implementations by default allow an adversary to collect as many traces as desired, limiting the impact of increasing the attack's capability by a few percentages. By contrast, there are other applications using other cryptographic tools that can fundamentally limit the number of observable test traces.

One such use case is the key-exchange protocols that limit the adversary to a single measurement trace. Our work has shown that learning-based approaches [47], [67] can outperform classical solutions [71], [72], [73] in the context of single-trace attacks. These works have, furthermore, extended the analysis to next-generation cryptosystems including lattice-based cryptography.

3) Opportunity #1: Leveraging the Developments in AI/ML: AI/ML research is still rapidly evolving in various aspects. These create an opportunity to advance learning-based SCA. In particular, the data augmentation approaches, explainability, fairness, and methods to accelerate the learning process with fewer data are among the important things that are under constant improvement. Likewise, the new types of learning methods or network topologies improve the success rate. Naturally, the advances in this sub-domain will improve the side-channel research. But the improvement is non-trivial and requires domain expertise in both side-channels and ML theory. Workshops targeting this particular sub-field could help bolster the capabilities of the hardware security community.

4) Opportunity #2: Cross-Device Attacks: Cross-device attacks are arguably where the learning-based approaches shine given the generalizability properties of ML. Of particular importance is the heterogeneous attacks [74] where the learning can generalize across different devices—for example, the training can occur on a 16-bit architecture and the attack can execute on a 32-bit architecture from a different vendor with a different ISA. These types of attacks can be critical for military applications where the target device may be unique that is not available in the commercial space. The limits of these cross-device attacks and their comparison with *non-profiling learning-based* approaches are largely unknown. Therefore, there are research opportunities in this direction.

IV. Machine-learning based SCA on AES-128

All cryptographic algorithms, while ideally mathematically secure, are almost certain to have flaws when implemented in hardware, and an important question is whether or not the effort required to break them is worth the reward an attacker would gain in doing so. Attempting to break them non-maliciously allows us to gain a sense of this trade-off and decide whether or not a method of data protection will be secure enough for a proposed use case, along with potentially resulting in the future development and implementation of preventative countermeasures to various forms of attack.

While we hope that our research will provide new insights and paths forward regarding machine learning-based side-channel analysis (MLSCA), we are certainly not the first to have investigated this topic. Many prior publications on it exist including [75], [76], [77], and [78]. In this section, we present our findings of an evaluation of the Advanced Encryption Standard, particularly unmasked and masked AES-128 implemented in software on a pair of STM32F415 microcontroller units (MCUs) and unmasked AES-128 implemented in hardware on a pair of Artix-7 XC7A100T field-programmable gate arrays (FPGAs), against machine learning-based side-channel analysis (MLSCA). 12 machine learning classifiers were used in combination with a side-channel leakage model in the context of different scenarios involving targets and keys.

A. Methodology

The following section discusses the steps we took to perform our evaluation of the MLSCA resistance of AES-128 implemented on the STM32F415 MCUs and XC7A100T FPGAs. We start with a description of our test setup followed by a discussion of our procedure, the machine learning models used, and finally the rank metric.

1) Experimental Setup: To begin, we first needed to collect power consumption data associated with encryptions performed by the two targets of each device type. More specifically, we measured the voltage across a shunt resistor inserted in the power supply rail of a target with the help of a CW1173 ChipWhisperer-Lite. As stated on their site, "The ChipWhisperer® ecosystem presents the first open-source, low-cost solution to expose weaknesses in embedded systems all around us" [79]. The ChipWhisperer-Lite essentially operates like an oscilloscope, while also allowing for communication between a laptop and a target board using an open-source Python package for control. Our evaluation setup consists of a MacBook Pro, ChipWhisperer-Lite, two STM32F415 MCUs capable of being mounted on a CW308 UFO Board to facilitate connections, and two XC7A100T FPGAs can be seen in Figures 5 and 6. On the software side, we used Python packages including but not limited to the ChipWhisperer, Keras/TensorFlow, Scikit-learn, NumPy, Pandas, and Matplotlib libraries in combination with a Jupyter Notebook to carry out our data processing and analysis.

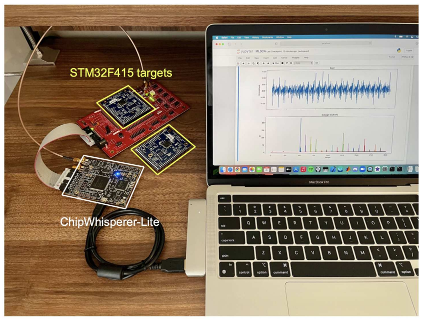

Fig. 5: Evaluation setup with a MacBook Pro, ChipWhisperer-Lite, and STM32F415 targets with a CW308 UFO Board.

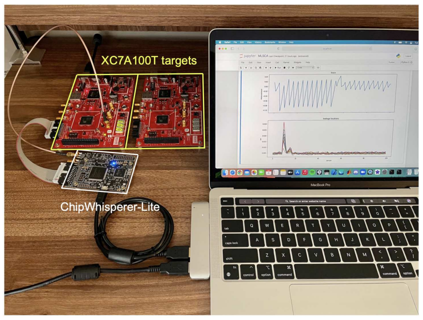

Fig. 6: Evaluation setup with a MacBook Pro, ChipWhisperer-Lite, and XC7A100T targets.

For each device type (MCU and FPGA), we performed an evaluation of four scenarios: profiling one target and key and attacking the same target with the same key, profiling one target and key and attacking the same target with a different key, profiling one target and key and attacking a different target with the same key, and profiling one target and key and attacking a different target with a different key.

2) *STM32F415 MCU and XC7A100T FPGA Evaluation Procedures:* Beginning with a standard unmasked software implementation of AES-128, we downloaded the program to the first of our two MCUs and triggered a series of encryptions in order to capture enough power consumption traces for the profiling set for our machine learning classifiers. We decided to collect 60,000 traces consisting of 2,000 samples each, corresponding to 60,000 different encryptions of randomly generated 128-bit plaintext with a static 128-bit key. Through trial and error, 2,000 samples turned out to be enough to capture the operation of interest to us in the first round of encryption. We also collected four additional sets of 10,000 traces each to serve as our attack sets. The first two of these four sets were collected from the same target that was profiled, with one set of encryptions using the same key that was used in the profiling set and the other using a different key. The second two attack sets were collected from the second STM32F415 MCU, with one set again using the same key that was profiled on the first target and the second using a different key. This same general process was used when collecting traces for datasets associated with a masked software implementation of AES-128 on the STM32F415 MCUs. We saved the total 100,000 collected traces for each implementation along with the associated 128-bit key, plaintext, and ciphertext values from each encryption for later analysis.

After collecting our datasets for the evaluation, we then applied a side-channel leakage model to the profiling set in an attempt to determine which samples in our collected traces may correspond to the processing of an operation of interest to us as attackers. For attacking both the unmasked and masked software implementations of AES-128 on the MCUs, the operation of interest to us was the "substitute bytes" or S-box operation in the first round. Only a few operations occur prior to this in the encryption process - those being the generation of the round keys derived from the original key and an "add round key" operation that consists of an XOR between the bytes of the zeroth round key (or the key itself) and the input plaintext. It's been shown prior to us that the power consumption corresponding to the processing of the S-box operation may be correlated to its output values and that if you're able to predict what those values are, it's possible to work backward to the secret key used. This is assuming that as an attacker you kept track of what random plaintext you used for each encryption, as there is only one value out of 256 possible values for the first key byte that when XORed with the first plaintext byte will result in the predicted first output byte of the S-box operation and so forth. A quick for loop can be used to figure out what the value is for each of the 16 key bytes.

For evaluating the FPGAs, we collected the same number of traces for our profiling and attack sets as we did for the MCUs, but only 100 samples were captured per trace, as the hardware implementation of AES-128 ran significantly faster.

B. *Evaluation Metric - Rank*

The method we used to determine whether or not a machine learning classifier was successful in predicting the key we were looking for was by using a metric known in the side-channel analysis research community as "rank" or "byte rank." As stated previously for our MCU evaluation, the training and test sets for the 16 bytes were labeled with the correct S-box output bytes for each encryption, but the accuracy of correct prediction based on a single trace was found to be unreliable. Rather, the predictions from many traces needed to be taken into account in order to have a fighting chance of determining the correct key.

In the case of a byte, there are 256 values it can take, so we asked our machine learning classifiers to return an array of 256 probabilities corresponding to each of the potential S-box output values (0-255). Then, we looped over all of the traces' returned probability arrays, and for each one, we determined what key byte value was associated with each of the possible S-box values using the plaintext associated with each trace. Once we figured out what S-box value probability was associated with each of the 256 possible key byte values, we added the probabilities to an array of sums corresponding to key byte value probabilities. As we looped over all of the traces' probability arrays and continued to add up probabilities, we hoped (as an attacker) that eventually the key byte with the highest total value after summing up all probabilities would be the correct one. We also kept track of where the correct value stood in relation to the 255 incorrect values over the course of this process. That way we could plot how its rank changed over time, and how many trace predictions it took before reaching the top rank or highest likelihood. In our case, we chose 0 to represent the highest rank and 255 for the lowest.

C. *Results*

The following section discusses what we found in performing our evaluation of the MLSCA resistance of AES-128 implemented on the STM32F415 and XC7A100T.

Classifier Rank Comparison for the STM32F415 and XC7A100T												
Machine Learning Classifier	Unmasked Software AES-128 (MCU) Average Number of Attack Traces to Rank 0				Masked Software AES-128 (MCU) Average Number of Attack Traces to Rank 0				Unmasked Hardware AES-128 (FPGA) Average Number of Attack Traces to Rank 0			
	SD & SK	SD & DK	DD & SK	DD & DK	SD & SK	SD & DK	DD & SK	DD & DK	SD & SK	SD & DK	DD & SK	DD & DK
AdaBoost	69	49	10	12	8	N/A	9	N/A	N/A	N/A	N/A	N/A
Convolutional Neural Network	83	83	7	8	13	N/A	17	N/A	304	573	414	585
Decision Tree	90	105	40	28	31	N/A	44	N/A	N/A	N/A	N/A	N/A
Gaussian Naive Bayes	88	129	15	13	20	N/A	37	N/A	449	583	527	712
K-Nearest Neighbors	8	7	3	4	5	N/A	7	N/A	1484	1849	1161	3147
Linear Discriminant Analysis	118	136	6	4	8	N/A	19	N/A	251	508	312	524
Logistic Regression	109	98	6	5	7	N/A	15	N/A	249	505	312	522
Long Short-Term Memory Network	15	13	6	5	7	N/A	11	N/A	283	448	293	472
Multilayer Perceptron Network	110	133	12	8	11	N/A	26	N/A	574	996	594	1695
Restricted Boltzmann Machine	156	149	48	57	38	N/A	41	N/A	N/A	N/A	N/A	N/A
Random Forest	33	52	25	24	30	N/A	36	N/A	N/A	N/A	N/A	N/A
Support Vector Machine	7	9	5	4	4	N/A	16	N/A	314	612	316	672

Fig. 7: Average byte rank 0 convergence for classifiers with 20,000 profiling and 1,000 attack traces for the MCUs and 60,000 profiling and 10,000 attack traces for the FPGAs (SD = same device/target that was profiled, SK = same key that was profiled, DD = different device/target than what was profiled, DK = different key than what was profiled).

1) Unmasked Software AES-128 on the STM32F415: To begin with the unmasked software implementation of AES-128 on the MCUs, we found that in all four attack scenarios previously discussed (involving two targets and two keys), we were able to recover the full encryption key with all 12 of our chosen machine learning classifiers in combination with a first-round S-box leakage model. The classifiers were easily able to do this with little to no tuning, rendering this combination of device and AES variant very vulnerable to this form of attack.

2) Masked Software AES-128 on the STM32F415: Concerning the masked software implementation, we found that in two out of the four attack scenarios, the machine learning classifiers in combination with the same leakage model were again easily able to recover the secret key. However, in the two scenarios involving a different attack key than the profiling key, all classifiers failed at predicting it. In rare cases we were able to recover partial keys, but never the full 16 bytes. Given that these two scenarios (attacking the same target that was profiled with a different key and attacking a different target than what was profiled with a different key) are likely the most realistic scenarios out of the four, we think that masking can indeed be an effective countermeasure against MLSCA.

3) Unmasked Hardware AES-128 on the XC7A100T: For the unmasked hardware implementation of AES-128 on the FPGAs, we found that we were able to reliably predict full encryption keys with 8 out of 12 of the machine learning classifiers, with a caveat. We were unable to recover bytes 0, 4, 8, and 12 when using a separately trained classifier to predict each of the 16 bytes in the two scenarios involving different profiling and attack keys for the FPGA. Interestingly, these four bytes make up the first column of the AES state array. Why exactly this is the case, we are not entirely sure at the time of writing this paper, but future investigation may reveal the reason. In any case, we found a way around this issue from an attacking standpoint and discovered that if we used a single model that was able to predict its corresponding byte successfully to predict the other 15 bytes as well, we were in fact able to predict the full key reliably with the 8 successful classifiers. This showed that training separate models for each byte was not actually necessary and that in a case like this it could potentially hinder the attack. While using a single model to predict all bytes, in general, led to an increase in the number of traces required in the prediction stage (calculating rank) for the 15 bytes it was not trained on, it was still a faster evaluation process overall, as 15 out of the 16 machine learning classifiers that previously consumed training time were no longer relevant.

4) Classifier Comparison: In terms of how the 12 machine learning classifiers compared to each other performance-wise, the table in Figure 7 displays the average number of attack traces required per byte before reaching a rank of 0 for all 12 classifiers in the various scenarios. It should be noted that for the results in the STM32F415 table, we used reduced profiling and attack sets (20,000 and 1,000 traces, respectively). These tables should not be considered an end-all for determining which classifier is the best for MLSCA in general, as how well a classifier performs can vary depending on the situation. Take the K-nearest neighbors classifier for instance, which was one of the best for the MCUs and the worst for the FPGAs. This being said, in the context of evaluating AES-128 on the STM32F415, our long short-term memory, K-nearest neighbors, and support vector machine classifiers tended to require the least number of traces to rank the correct key byte values as most probable, and for the XC7A100T, our convolutional neural network, linear discriminant analysis, logistic regression, long short-term memory, and support vector machine classifiers were all very comparable. The SVM classifier did however take considerably longer to run than any of the others, so while it did tend to produce a low-rank convergence, it wasn't nearly as time-efficient as the rest. It should also be noted that the ensemble/tree-based machine learning algorithms failed in predicting keys for the FPGAs, along with our restricted Boltzmann machine classifier.

V. Conclusion

For the past years, given the rising amount of documented attacks on the hardware, the presumption of secure hardware is no longer valid. The requirement to patch/update software-based solutions due to the advent of new security threats (such as malware, side-channel attacks, etc.) necessitates a significant amount of memory and hardware resources. As a result, rather than considering security as an afterthought, it should be entrusted to the underlying hardware when designing a bottom-up approach to safeguarding computer systems. Research advancements have shown that hardware systems enable a swift analysis of applications' trends and learn from them with the use of intelligent machine learning-based security countermeasures, allowing them to successfully avoid future attacks and respond pro-actively to changing behavior in real-time. This work emphasizes the expanding role of AI/ML techniques in hardware and architecture security and offers insightful discussions on urgent challenges, opportunities, and future directions of developing precise and effective machine learning-based attacks and defense mechanisms in response to emerging hardware security vulnerabilities in contemporary computer systems and next-generation cryptosystems.

VI. Acknowledgment

This work is supported in part by the National Science Foundation under Award No. 2139034, and upon work supported in part by the National Science Foundation under Grant No. CNS #16-2137283 - Center for Advanced Electronics through Machine Learning (CAEML) and its industry members.

References

[1] J. Demme *et al.*, "On the feasibility of online malware detection with performance counters," in *ISCA'13*. ACM, 2013, pp. 559–570.

[2] P. Kocher *et al.*, "Spectre attacks: Exploiting speculative execution," in *2019 IEEE Symposium on Security and Privacy (SP)*, 2019, pp. 1–19.

[3] M. Lipp *et al.*, "Meltdown: Reading kernel memory from user space," in *27th USENIX Security Symposium (USENIX Security 18)*, Baltimore, MD, Aug. 2018, pp. 973–990.

[4] H. Sayadi *et al.*, "Ensemble learning for effective run-time hardware-based malware detection: A comprehensive analysis and classification," in *DAC'18*, 2018, pp. 1–6.

[5] A. Aysu, Y. Tobah, M. Tiwari, A. Gerstlauer, and M. Orshansky, "Horizontal Side-channel Vulnerabilities of Post-Quantum Key Exchange Protocols," in *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2018, pp. 81–88.

[6] F. Regazzoni *et al.*, "Machine learning and hardware security: Challenges and opportunities -invited talk-," in *IEEE/ACM International Conference On Computer Aided Design, ICCAD 2020, San Diego, CA, USA, November 2-5, 2020*. IEEE, 2020, pp. 141:1–141:6.

[7] H. Wang *et al.*, "Hybrid-shield: Accurate and efficient cross-layer countermeasure for run-time detection and mitigation of cache-based side-channel attacks," in *ICCAD'20*, ser. ICCAD '20, 2020.

[8] A. Mosenia and N. K. Jha, "A comprehensive study of security of internet-of-things," *IEEE Transactions on Emerging Topics in Computing*, vol. 5, no. 4, pp. 586–602, Oct 2017.

[9] H. Sayadi, H. Makrani, O. Randive, S. M. P D, S. Rafatirad, and H. Homayoun, "Customized machine learning-based hardware-assisted malware detection in embedded devices," *The 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications (IEEE TrustCom-18)*, 2018.

[10] G. Kornaros, "Hardware-assisted machine learning in resource-constrained iot environments for security: Review and future prospective," *IEEE Access*, 2022.

[11] S. M. P. Dinakarrao *et al.*, "Lightweight node-level malware detection and network-level malware confinement in iot networks," in *DATE'19*, March 2019, pp. 776–781.

[12] Y. Ye, T. Li, D. Adjeroh, and S. S. Iyengar, "A survey on malware detection using data mining techniques," *ACM Computing Surveys*, vol. 50, no. 3, pp. 1–40, 2017. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3101309.3073559

[13] M. Schmall, "Heuristic techniques in av solutions: An overview," in *https://www.symantec.com/connect/articles/heuristic-techniques-av-solutions-overview*, 2002.

[14] Z. Bazrafshan, H. Hashemi, S. M. H. Fard, and A. Hamzeh, "A survey on heuristic malware detection techniques," in *The 5th Conference on Information and Knowledge Technology*, May 2013, pp. 113–120.

[15] B. Singh *et al.*, "On the detection of kernel-level rootkits using hardware performance counters," in *ASIACCS'17*, 2017, pp. 483–493.

[16] H. Sayadi *et al.*, "2smart: A two-stage machine learning-based approach for run-time specialized hardware-assisted malware detection," in *DATE'19*, March 2019, pp. 728–733.

[17] A. Tang *et al.*, "Unsupervised anomaly-based malware detection using hardware features," in *RAID'14*. Springer, 2014, pp. 109–129.

[18] T. Zhang, Y. Zhang, and R. B. Lee, "Cloudradar: A real-time side-channel attack detection system in clouds," in *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer, 2016, pp. 118–140.

[19] M. Chiappetta, E. Savas, and C. Yilmaz, "Real time detection of cache-based side-channel attacks using hardware performance counters," *Applied Soft Computing*, vol. 49, pp. 1162–1174, 2016.

[20] M. Mushtaq, J. Bricq, M. K. Bhatti, A. Akram, V. Lapotre, G. Gogniat, and P. Benoit, "Whisper: A tool for run-time detection of side-channel attacks," *IEEE Access*, vol. 8, pp. 83 871–83 900, 2020.

[21] M. Mushtaq *et al.*, "Nights-watch: A cache-based side-channel intrusion detector using hardware performance counters," in *Proceedings of the 7th International Workshop on Hardware and Architectural Support for Security and Privacy*. ACM, 2018, p. 1.

[22] Y. Gao *et al.*, "Adaptive-hmd: Accurate and cost-efficient machine learning-driven malware detection using microarchitectural events," in *IOLTS'21*. IEEE, 2021, pp. 1–7.

[23] H. Liu *et al.*, *Feature selection for knowledge discovery and data mining*. Springer Science & Business Media, 2012, vol. 454.

[24] L. Yu *et al.*, "Feature selection for high-dimensional data: A fast correlation-based filter solution," in *Proceedings of the 20th international conference on machine learning (ICML-03)*, 2003, pp. 856–863.

[25] "Intel performance monitoring unit," in *https://software.intel.com/en-us/articles/intel-performance-counter-monitor*.

[26] V. J. Reddi, A. Settle, D. A. Connors, and R. S. Cohn, "Pin: a binary instrumentation tool for computer architecture research and education," in *Proceedings of the 2004 workshop on Computer architecture education: held in conjunction with the 31st International Symposium on Computer Architecture*, 2004, pp. 22–es.

[27] P. J. Mucci, S. Browne, C. Deane, and G. Ho, "Papi: A portable interface to hardware performance counters," in *Proceedings of the department of defense HPCMP users group conference*, vol. 710, 1999.

[28] J. Reinders, "Vtune performance analyzer essentials," *Intel Press*, 2005.

[29] T. Willhalm, R. Dementiev, and P. Fay, "Intel performance counter monitor-a better way to measure cpu utilization," *Dosegljivo: https://software. intel. com/en-us/articles/intelperformance-countermonitor-a-better-way-to-measure-cpu-utilization.[Dostopano: September 2014]*, 2012.

[30] *https://perf.wiki.kernel.org/index.php*, last accessed: 20-Feb-2019.

[31] K. London, S. Moore, P. Mucci, K. Seymour, and R. Luczak, "The papi cross-platform interface to hardware performance counters," in *Department of Defense Users' Group Conference Proceedings*, 2001, pp. 18–21.

[32] T. Willhalm, R. Dementiev, and P. Fay, "Intel® performance counter monitor - a better way to measure cpu utilization," *https://software.intel.com/content/www/us/en/develop/articles/intel-performance-counter-monitor.html*, 2017.

[33] S. J. Stolfo *et al.*, "Towards stealthy malware detection," in *Malware Detection*. Boston, MA: Springer US, 2007, pp. 231–249.

[34] H. Sayadi *et al.*, "Stealthminer: Specialized time series machine learning for run-time stealthy malware detection based on microarchitectural features," in *GLSVLSI'20*, 2020, p. 175–180.

[35] H. Sayadi, Y. Gao, H. Mohammadi Makrani, J. Lin, P. C. Costa, S. Rafatirad, and H. Homayoun, "Towards accurate run-time hardware-assisted stealthy malware detection: a lightweight, yet effective time series cnn-based approach," *Cryptography*, vol. 5, no. 4, p. 28, 2021.

[36] S. Das *et al.*, "Sok: The challenges, pitfalls, and perils of using hardware performance counters for security," in *IEEE SP*, 2019, pp. 20–38.

[37] B. Zhou *et al.*, "Hardware performance counters can detect malware: Myth or fact?" in *ASIACCS'18*, 2018, pp. 457–468.

[38] O. Suciu, S. E. Coull, and J. Johns, "Exploring adversarial examples in malware detection," in *2019 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2019, pp. 8–14.

[39] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *2016 IEEE European Symposium on Security and Privacy (EuroS P)*, 2016, pp. 372–387.

[40] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," in *5th International*

Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net, 2017. [Online]. Available: https://openreview.net/forum?id=Sys6GJqxl

[41] S. M. P. Dinakarrao *et al.*, "Adversarial attack on microarchitectural events based malware detectors," in *Proceedings of the 56th Annual Design Automation Conference 2019*, 2019, pp. 1–6.

[42] A. P. Kuruvila, S. Kundu, and K. Basu, "Defending hardware-based malware detectors against adversarial attacks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 9, pp. 1727–1739, 2021.

[43] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in cryptology–CRYPTO'99*. Springer, 1999, pp. 789–789.

[44] J.-J. Quisquater and D. Samyde, "Electromagnetic analysis (ema): Measures and counter-measures for smart cards," in *Smart Card Programming and Security*. Springer Berlin Heidelberg, 2001, pp. 200–210.

[45] M. A. Islam, S. Ren, and A. Wierman, "Exploiting a thermal side channel for power attacks in multi-tenant data centers," in *ACM Conference on Computer and Communications Security*, 2017, p. 1079–1094.

[46] D. Genkin, A. Shamir, and E. Tromer, "Acoustic cryptanalysis," *J. Cryptol.*, vol. 30, no. 2, pp. 392–443, 2017.

[47] F. Aydin, P. Kashyap, S. Potluri, P. Franzon, and A. Aysu, "Deeparsca: Breaking parallel architectures of lattice cryptography via learning based side-channel attacks," in *Embedded Computer Systems: Architectures, Modeling, and Simulation*, A. Orailoglu, M. Jung, and M. Reichenbach, Eds. Cham: Springer International Publishing, 2020, pp. 262–280.

[48] Y. LeCun, Y. Bengio, and G. E. Hinton, "Deep learning," *Nat.*, vol. 521, no. 7553, pp. 436–444, 2015.

[49] B. Hettwer, S. Gehrer, and T. Güneysu, "Applications of machine learning techniques in side-channel attacks: a survey," *J. Cryptogr. Eng.*, vol. 10, no. 2, pp. 135–162, 2020.

[50] H. Maghrebi, T. Portigliatti, and E. Prouff, "Breaking cryptographic implementations using deep learning techniques," in *SPACE*, ser. Lecture Notes in Computer Science, C. Carlet, M. A. Hasan, and V. Saraswat, Eds., vol. 10076. Springer, 2016, pp. 3–26.

[51] "Sok: Deep learning-based physical side-channel analysis," in *IACR Cryptol. ePrint Arch.*, 2021, p. 1092.

[52] B. Timon, "Non-profiled deep learning-based side-channel attacks with sensitivity analysis," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2019, no. 2, p. 107–131, Feb. 2019.

[53] J. Rijsdijk, L. Wu, G. Perin, and S. Picek, "Reinforcement learning for hyperparameter tuning in deep learning-based side-channel analysis," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2021, no. 3, p. 677–707, Jul. 2021.

[54] M. Carbone, V. Conin, M.-A. Cornélie, F. Dassance, G. Dufresne, C. Dumas, E. Prouff, and A. Venelli, "Deep learning to evaluate secure rsa implementations," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2019, no. 2, p. 132–161, Feb. 2019.

[55] B. Hettwer, S. Gehrer, and T. Güneysu, "Deep neural network attribution methods for leakage analysis and symmetric key recovery," in *Selected Areas in Cryptography – SAC 2019*, K. G. Paterson and D. Stebila, Eds. Cham: Springer International Publishing, 2020, pp. 645–666.

[56] M. Raghu, J. Gilmer, J. Yosinski, and J. Sohl-Dickstein, "Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 6078–6087.

[57] A. Heuser and M. Zohner, "Intelligent Machine Homicide," in *Constructive Side-Channel Analysis and Secure Design*. Springer, 2012, pp. 249–264.

[58] L. Lerman, G. Bontempi, and O. Markowitch, "Power Analysis Attack: An Approach Based on Machine Learning," *International Journal of Applied Cryptography*, vol. 3, no. 2, pp. 97–115, Jun. 2014.

[59] R. Gilmore, N. Hanley, and M. O'Neill, "Neural Network based Attack on a Masked Implementation of AES," in *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2015, pp. 106–111.

[60] J. Kim, S. Picek, A. Heuser, S. Bhasin, and A. Hanjalic, "Make Some Noise. Unleashing the Power of Convolutional Neural Networks for Profiled Side-channel Analysis," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2019, no. 3, pp. 148–179, 2019. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/8292

[61] H. Maghrebi, T. Portigliatti, and E. Prouff, "Breaking Cryptographic Implementations Using Deep Learning Techniques," in *Security, Privacy, and Applied Cryptography Engineering (SPACE)*, 2016, pp. 3–26.

[62] S. Picek, A. Heuser, A. Jovic, S. A. Ludwig, S. Guilley, D. Jakobovic, and N. Mentens, "Side-channel Analysis and Machine Learning: A Practical Perspective," in *International Joint Conference on Neural Networks (IJCNN)*, 2017, pp. 4095–4102.

[63] E. Prouff, R. Strullu, R. Benadjila, E. Cagli, and C. Dumas, "Study of Deep Learning Techniques for Side-Channel Analysis and Introduction to ASCAD Database," *IACR Cryptology ePrint Archive*, vol. 2018, p. 53, 2018. [Online]. Available: http://eprint.iacr.org/2018/053

[64] B. Timon, "Non-Profiled Deep Learning-based Side-Channel attacks with Sensitivity Analysis," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2019, no. 2, pp. 107–131, 2019.

[65] S. G. B. Hettwer, T. Horn and T. Güneysu, "Encoding Power Traces as Images for Efficient Side-Channel Analysis," in *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2020, pp. 1–10.

[66] V. Nair and G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines," in *International Conference on International Conference on Machine Learning (ICML)*. PMLR, 2010, pp. 807–814.

[67] P. Kashyap, F. Aydin, S. Potluri, P. D. Franzon, and A. Aysu, "2deep: Enhancing side-channel attacks on lattice-based key-exchange via 2-d deep learning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 6, pp. 1217–1229, 2021.

[68] C. M. Bishop, *Pattern recognition and machine learning, 5th Edition*, ser. Information science and statistics. Springer, 2007.

[69] E. Prouff, R. Strullu, R. Benadjila, E. Cagli, and C. Dumas, "Study of deep learning techniques for side-channel analysis and introduction to ASCAD database," *IACR Cryptol. ePrint Arch.*, vol. 2018, p. 53, 2018.

[70] K. Ramezanpour, P. Ampadu, and W. Diehl, "Scaul: Power side-channel analysis with unsupervised learning," *IEEE Transactions on Computers*, vol. 69, no. 11, pp. 1626–1638, 2020.

[71] F. Aydin, E. Karabulut, S. Potluri, E. Alkim, and A. Aysu, "Reveal: Single-trace side-channel leakage of the seal homomorphic encryption library," in *2022 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2022, pp. 1527–1532.

[72] F. Aydin, A. Aysu, M. Tiwari, A. Gerstlauer, and M. Orshansky, "Horizontal side-channel vulnerabilities of post-quantum key exchange and encapsulation protocols," *ACM Trans. Embed. Comput. Syst.*, vol. 20, no. 6, oct 2021.

[73] A. Aysu, Y. Tobah, M. Tiwari, A. Gerstlauer, and M. Orshansky, "Horizontal side-channel vulnerabilities of post-quantum key exchange protocols," in *2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2018, pp. 81–88.

[74] F. Zhang, B. Shao, G. Xu, B. Yang, Z. Yang, Z. Qin, and K. Ren, "From homogeneous to heterogeneous: Leveraging deep learning based power analysis across devices," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, 2020, pp. 1–6.

[75] H. Maghrebi, T. Portigliatti, and E. Prouff, "Breaking cryptographic implementations using deep learning techniques," in *International Conference on Security, Privacy, and Applied Cryptography Engineering*. Springer, 2016, pp. 3–26.

[76] H. Maghrebi, "Deep learning based side channel attacks in practice." *IACR Cryptol. ePrint Arch.*, vol. 2019, p. 578, 2019.

[77] B. Sönmez, A. A. Sarıkaya, and Ş. Bahtiyar, "Machine learning based side channel selection for time-driven cache attacks on aes," in *2019 4th International Conference on Computer Science and Engineering (UBMK)*. IEEE, 2019, pp. 1–5.

[78] R. Benadjila *et al.*, "Study of deep learning techniques for side-channel analysis and introduction to ascad database," *ANSSI, France & CEA, LETI, MINATEC Campus, France. Online verfügbar unter https://eprint. iacr. org/2018/053. pdf, zuletzt geprüft am*, vol. 22, p. 2018, 2018.

[79] "Chipwhisperer." [Online]. Available: https://www.newae.com/chipwhisperer