Practical Approaches Toward Deep-Learning-Based Cross-Device Power Side-Channel Attack

Anupam Golder, *Student Member, IEEE*, Debayan Das, *Student Member, IEEE*, Josef Danial, *Student Member, IEEE*, Santosh Ghosh, Shreyas Sen, *Senior Member, IEEE*, and Arijit Raychowdhury, *Senior Member, IEEE*

***Abstract*— Power side-channel analysis (SCA) has been of immense interest to most embedded designers to evaluate the physical security of the system. This work presents profiling-based cross-device power SCA attacks using deep-learning techniques on 8-bit AVR microcontroller devices running AES-128. First, we show the practical issues that arise in these profiling-based cross-device attacks due to significant device-to-device variations. Second, we show that utilizing principal component analysis (PCA)-based preprocessing and multidevice training, a multilayer perceptron (MLP)-based 256-class classifier can achieve an average accuracy of 99.43% in recovering the first keybyte from all the 30 devices in our data set, even in the presence of significant interdevice variations. Results show that the designed MLP with PCA-based preprocessing outperforms a convolutional neural network (CNN) with four-device training by ∼20% in terms of the average test accuracy of cross-device attack for the aligned traces captured using the ChipWhisperer hardware. Finally, to extend the practicality of these cross-device attacks, another preprocessing step, namely, dynamic time warping (DTW) has been utilized to remove any misalignment among the traces, before performing PCA. DTW along with PCA followed by the 256-class MLP classifier provides ≥10.97% higher accuracy than the CNN-based approach for cross-device attack even in the presence of up to 50 time-sample misalignments between the traces.**

***Index Terms*— Cross-device attacks, deep learning, dynamic time warping (DTW), principal component analysis (PCA), profiling attacks, side-channel analysis (SCA).**

I. Introduction

A. Motivation

In today's world, to establish secure communication between two parties, the use of cryptographic algorithms is commonplace. Although these mathematically secure crypto algorithms cannot be broken by means of brute-force attack, there have been numerous accounts of breaking the secret key by utilizing side-channel information in the form of power consumption [1], electromagnetic radiation [2]–[4], and optical [5], [6], or acoustic [7] vibrations captured from hardware implementations of the algorithms.

In this paper, we focus on power side-channel analysis (SCA)-based attacks, and specifically on profiling-based attacks [8]. In traditional nonprofiled attacks, such as differential and correlation power analysis (DPA [1]/CPA [9]) attacks, the attacker gathers power traces from a target device and uses statistical techniques, such as the difference of mean (DOM) traces or Pearson correlation coefficient to break the secret key. On the other hand, profiling-based attacks [8], [10], [11] assume the worst case scenario from the perspective of the target crypto engine, where the adversary is assumed to possess an identical device to profile the leakage patterns for all possible combinations for a keybyte (profiling or training phase), and use this prior knowledge to identify the secret key of the victim's crypto engine (online or test phase). Such an attack has been demonstrated on a commercially available contactless smart card in [12].

Traditionally, profiling attacks are performed by generating templates (hence called *template attack* [8], [13]–[15]) for different keys by utilizing a multivariate Gaussian distribution approximation of the preidentified points-of-interest (POIs). Recently, the hardware security research community has focused its attention to the machine learning (ML)-based profiled attacks using support vector machine (SVM) [16], [17], random forest (RF) [17], as well as deep-learning-based attacks [10], [18]–[20]. The advantage of deep-learning-based attacks is that not only do they perform as good as the template-based attacks, they do not require extensive statistical analysis to identify POIs. Moreover, as the number of dimensions increase, ML-based attacks start to gain interest, because an increase in number of noninformative points in POIs degrades performance of template attacks [11]. However, these ML-based approaches mentioned above rely on the assumption that the leakage profile from the profiling and the target devices are similar.

Recently, Das *et al.* [21] showed the first cross-device profiling-based attack using neural networks. However, the attack scenario was limited to a small sample of devices taken from the same batch.

Manuscript received February 2, 2019; revised May 23, 2019; accepted June 8, 2019. Date of publication July 26, 2019; date of current version November 22, 2019. This work was supported in part by the National Science Foundation (NSF) under Grant CNS 17-19235. The work of A. Golder was supported by NSF through the Center for Advanced Electronics through Machine Learning (CAEML) and its industry members under Grant CNS 16-24810. *(Corresponding author: Anupam Golder.)*

A. Golder and A. Raychowdhury are with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: anupamgolder@gatech.edu; arijit.raychowdhury@ece.gatech.edu).

D. Das, J. Danial, and S. Sen are with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907 USA (e-mail: das60@purdue.edu; jdanial@purdue.edu; shreyas@purdue.edu).

S. Ghosh is with the Intel Labs, Intel Corporation, Hillsboro, OR 97124 USA (e-mail: santosh.ghosh@intel.com).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2019.2926324

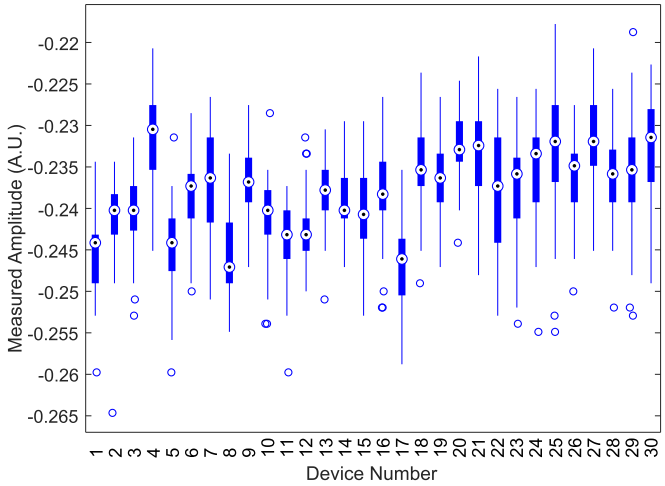

Fig. 1. Box-and-whisker plots showing a distribution of 180 observations of sample #96 of the power traces obtained from 30 identical microcontroller devices in two separate batches [CW308T-XMega, Batch1: Date Code: 1848, Lot Code: 0412—Devices (1–20), Batch2: Date Code: 1830, Lot Code: 0350—Devices (21–30)], running an AES encryption operation for a fixed plaintext and a fixed subkey 0×00 for the first keybyte. Amplitudes are reported in Arbitrary Unit (A.U.), as the ChipWhisperer capture platform does not report any unit for the measured power. The shift in medians and interquartile ranges can be attributed to manufacturing and packaging variations of different devices and circuit boards. Outliers, which fall outside the interquartile range for a specific distribution, are plotted in blue circles.

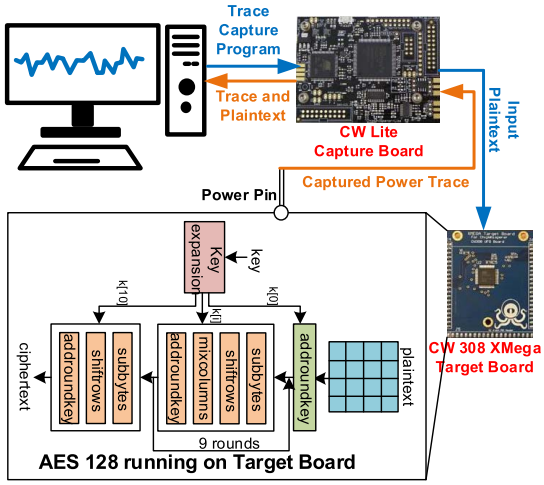

Fig. 2. ChipWhisperer platform for capturing and recording power traces from an AVR XMega microcontroller running AES-128 encryption algorithm. This platform supports programming the microcontroller to execute AES-128, modification of keys and plaintexts, and capturing of power traces with a perfectly triggered circuit to ensure alignment between traces.

Hence, the test accuracy for cross-device attack obtained using a multilayer perceptron (MLP) without any preprocessing was very optimistic. Also, Carbone *et al.* [22] have evaluated a secure RSA implementation using deep learning where the training, validation, and testing data were collected from three different smart cards only.

In a practical attack scenario, the target device may come from a different batch other than the one on which the ML-based classifier has been trained. Hence, the interdevice variations may be significantly higher, making cross-device attacks more difficult. This paper thoroughly analyzes the performance of the MLP classifier by obtaining 30 devices from two different batches (Fig. 1) and shows that using the MLP even when trained with four devices, the accuracy across the test devices may be as low as $\sim$ 8% for the devices from a different batch or from the same batch with high interdevice variation. Although this work is an extension of [21], it presents a significant advancement in generalizing cross-device attack by utilizing principal component analysis (PCA)-based preprocessing to project trace samples to their principal subspace [23], along with the multidevice training of the MLP classifier, which together provide a drastic improvement in minimum value of test accuracy by boosting it up to $\sim$90% from $\sim$8%. In addition, to extend and improve the practicality of this attack, we utilize dynamic time warping (DTW), preceding the PCA stage, to align the traces in case of desynchronization. Hence, DTW and PCA followed by the MLP classifier outperform the convolutional neural network (CNN)-based classifier by $>$10% in the test accuracy, even in the presence of up to 50 time sample misalignments among the traces. This extended paper includes the following key improvements over [21]: 1) a more detailed analysis on the performance of deep-learning techniques in cross-device attack scenario using traces collected from 30 devices; 2) a plausible explanation of improvement in test accuracy using multidevice training (Section III-C); 3) use of preprocessing to improve test accuracy instead of using multiple traces as presented in [21] (Sections IV and V); and 4) a comparative analysis with CNN (Sections III-E and V)

Fig. 1 shows a box-and-whisker plot of distributions of the measured amplitude of power consumption at a specific point in time from 30 different but identical microcontroller-based AES-128 crypto engines. Sample point #96 has been chosen using a feature selection method typically used for template attacks (Section III), namely, DOMs. Note how the medians and interquartile ranges vary from one device to another, and some outliers can be observed even for a particular device. It is clear from this figure that we need to validate our assumptions regarding identical leakage patterns from identical but different devices.

B. Contribution

Specific contributions of this paper are as follows.

1) This work analyzes the practical feasibility of using a 256-class classifier using MLP and CNN without any preprocessing to implement a cross-device attack on an 8-bit microcontroller-based SCA platform named ChipWhisperer [24] (Fig. 2) and demonstrates how multidevice training improves the test accuracy of cross-device attack (Section III) with a plausible explanation.
2) Using PCA [33]-based projection of raw traces to their principal subspaces along with multidevice training, we show that the accuracy of cross-device attack for all the test sets improves significantly (minimum test accuracy increases by $>10\times$), with $\geq$89.21% accuracy across all the 30 devices (Section IV).

TABLE I
LITERATURE REVIEW FOR PROFILED-ATTACK SCENARIO

Profiled-Attack Scenario	Method	Corresponding Articles
Same-device Attack	Template Attack	[8], [25], [26]
	Support Vector Machine	[16], [17], [27], [28]
	Random Forest	[17]
	Self-Organizing Map	[17]
	Time Series Approach	[29]
	Neural Networks	[10], [30], [19], [18], [20], [31]
Cross-device Attack	Template Attack	[14], [13], [32], [15]
	Neural Networks	[21], [22], This Work

3) Finally, to enhance the practicality of the proposed attack, we consider the misalignment between traces due to untimely triggering of the capture device (usually an oscilloscope). To resolve this issue, we demonstrate that DTW [34]-based preprocessing preceding PCA is a feasible solution, achieving $\geq$10.97% higher accuracy compared to a CNN-based approach (Section V) even in the presence of up to 50 time sample misalignments between traces for a cross-device attack scenario.

C. Paper Organization

In this paper, we use the following conventions: uppercase boldface italics for matrices, lowercase boldface italics for vectors, uppercase and lowercase italics for scalars, and uppercase italics within curly braces for sets.

The remainder of this paper is organized as follows. Section II presents the related works in the field of profiling-based attacks and summarizes the existing works on machine-learning-based side-channel attacks. In Section III, two deep neural networks (DNNs), MLP and CNN architectures, have been presented for a single-trace attack. Also, their limitations in cross-device attack scenario are presented. Section IV proposes a PCA-based preprocessing step to project raw traces to their principal subspace, and thus effectively increase the cross-device attack accuracy. Section V investigates a more practical scenario considering misalignment between traces, which can occur during trace capture, and we utilize a DTW-based preprocessing to realign traces, to allow subsequent PCA and MLP-classifier to work properly, and compare the performance of such an approach with that of a CNN. In Section VI, we present a relative timing performance comparison among different deep-learning techniques, and between deep-learning technique and CPA. Finally, in Section VII, we summarize the findings and conclude this paper.

II. BACKGROUND AND LITERATURE REVIEW

In the past couple of years, several ML techniques have been investigated, including but not limited to SVMs [16], [17] and RFs [17]. More recently, the signal processing community as well as the hardware security researchers have started exploring the field of DNNs [10], [30].

A. Related Works

Deep-learning-based profiling attacks have been successful even in the presence of masking [30] and jitter/misalignment [10]-based countermeasures. However, other than [21] and [22], none of the articles proposing ML-based attacks investigated a practical scenario where the attack is actually performed on a device other than the one used in training phase.

Table I summarizes the related works. As can be seen, most of the template attacks were evaluated on the same device, whereas only in a few cases [13]–[15], [32], attacks were performed on a different device. This paper significantly improves on [21] to present a generalized deep-learning-based cross-device SCA attack on 30 different devices, utilizing DTW and PCA along with the multidevice training.

There are two types of classification strategies for ML-based classifiers, one is based on the Hamming Weight (HW) model (nine-class classification) and the other is the identity (ID) model (256-class classification). When the ID model is used, the attack typically requires a single trace from the target device. Thus, these attacks are powerful in the event that the adversary has limited time and opportunity to gather such traces due to rekeying after each session. This paper utilizes the ID model to leverage the advantages of such an attack model.

Most of the previous ML-based attacks were evaluated using the DPAv2 [35] and DPAv4 [36] contest data sets, or recently published ASCAD [18] database. To the best of authors' knowledge, both data sets consist of traces captured from one single device, which are not suitable for current work. Hence, we collected new traces from 30 different 8-bit AVR microcontrollers running the AES-128 algorithm using the ChipWhisperer platform [24] (Fig. 2). Although 8-bit microcontrollers are becoming less preferred for encryption engines nowadays, a recent body of work [13], [18], [37]–[39] investigated the performance of profiled SCA attack using data sets gathered from 8-bit microcontrollers.

B. Background of Neural Networks

One of the most widely used neural networks is MLP that has an input layer, one or more hidden layers, and an output layer. Each layer has trainable weights and biases and produces an output based on its inputs coming from the preceding layer, weights connected to those inputs, and a nonlinear activation function, such as rectified linear units (ReLUs), sigmoid, or tanh. The choice of the number of layers, the number of neurons in each layer, and the activation function/s together creates the network architecture and defines the functions that can be approximated by this model. In a classification

problem, the last layer is a classification layer followed by a softmax layer. More complex architectures are possible such as CNN, which uses so-called convolutional layers that act as filters and slide over the preceding layer by 1 or more units (called stride), subsampling layers such as max-pooling or average-pooling to reduce the number of dimensions, and the fully connected (FC) layers. The gathered data are usually divided into three distinct sets: training set, as the name implies, to train the network, validation set to validate the performance of trained network on previously unseen data by usually keeping a part of training set separate from those actually used in training, and a test set, which is used to finally test the performance, i.e., prediction or classification accuracy. Typically, during the training phase, weights and biases of neurons are tuned iteratively over several epochs to minimize a categorical cross-entropy loss [40] function using a form of stochastic gradient descent (SGD) optimizer [41]. The training phase can be conducted in minibatches where the inputs for the network are continuously drawn from the complete set of training samples. On top of these, the use of batch normalization, dropout layer, and L_2 regularization are typical ways to address the problem of overfitting (when the model works well for the training set, but performs poorly on the unseen test set) to provide a better generalization on the test data. All the parameters that define the architecture of neural networks and dictate the training phase are called hyperparameters.

Note that, in contrast to template-based attack, where a template preparation method can be exported across platforms, the trained neural network architecture may be vastly different depending on the implementations. In other words, one particular neural network architecture for a specific attack scenario and a specific hardware implementation may not be the best-suited one for a different platform.

C. Overview of Principal Component Analysis

PCA [33] is a well-known dimensionality reduction technique and has been proven to be successful for time-series data. For example, we have a $M \times N$ matrix of traces, where each trace forms a row. In the following matrix representations, t_{ij} denotes sample j of trace i, and each column vector, $\boldsymbol{t_j}$ (each having a dimension of $M \times 1$) represents data for jth dimension

$$\boldsymbol{Traces} = \begin{bmatrix} \boldsymbol{trace_1} \\ \boldsymbol{trace_2} \\ \vdots \\ \boldsymbol{trace_M} \end{bmatrix} = \begin{bmatrix} t_{11} & t_{12} & \dots & t_{1N} \\ t_{21} & t_{22} & \dots & t_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ t_{M1} & t_{M2} & \dots & t_{MN} \end{bmatrix} = [\boldsymbol{t_1} \quad \boldsymbol{t_2} \quad \dots \quad \boldsymbol{t_N}].$$

Then, we subtract the mean from each of the dimensions to obtain a new matrix, $\boldsymbol{Traces_{adjust}}$

$$\boldsymbol{Traces_{adjust}} = [\boldsymbol{t_1} - \text{mean}(\boldsymbol{t_1}) \quad \boldsymbol{t_2} - \text{mean}(\boldsymbol{t_2}) \quad \dots \quad \boldsymbol{t_N} - \text{mean}(\boldsymbol{t_N})].$$

Next, the covariance (a measure of variation of each of the dimensions from their individual means with respect to each other; helpful for very high-dimensional data) matrix for the traces is computed as follows:

$$\begin{aligned} &\text{Covariance matrix}, \boldsymbol{C} \\ &= \text{cov}(\boldsymbol{Traces}) \\ &= \text{cov}([\boldsymbol{t_1} \quad \boldsymbol{t_2} \quad \dots \quad \boldsymbol{t_N}]) \\ &= \begin{bmatrix} \text{cov}(\boldsymbol{t_1}, \boldsymbol{t_1}) & \text{cov}(\boldsymbol{t_1}, \boldsymbol{t_2}) & \dots & \text{cov}(\boldsymbol{t_1}, \boldsymbol{t_N}) \\ \text{cov}(\boldsymbol{t_2}, \boldsymbol{t_1}) & \text{cov}(\boldsymbol{t_2}, \boldsymbol{t_2}) & \dots & \text{cov}(\boldsymbol{t_2}, \boldsymbol{t_N}) \\ \vdots & \vdots & \ddots & \vdots \\ \text{cov}(\boldsymbol{t_N}, \boldsymbol{t_1}) & \text{cov}(\boldsymbol{t_N}, \boldsymbol{t_2}) & \dots & \text{cov}(\boldsymbol{t_N}, \boldsymbol{t_N}) \end{bmatrix} \end{aligned}$$

Then, we calculate the unit eigenvector matrix of the covariance matrix, $\boldsymbol{V}$, and the diagonal eigenvalue matrix, $\boldsymbol{D}$. These unit eigenvectors are orthogonal to each other

$$\boldsymbol{V} = [\boldsymbol{v_1} \quad \boldsymbol{v_2} \quad \dots \quad \boldsymbol{v_N}].$$

Eigenvectors with the highest eigenvalues are the most significant principal components of the data set. Ordering eigenvalues from highest to lowest arranges the components according to the order of importance. If eigenvalues are very small, principal components corresponding to those of lesser importance can be discarded with negligible information loss. Here, the original data dimension was N, and we have correspondingly N eigenvalues and N eigenvectors. Choosing the first p eigenvalues after arranging them in descending order of eigenvalues, reduces the dimension for the data set to p. Then, we obtain a modified eigenvector matrix, $\boldsymbol{V_m}$ keeping the first p eigenvectors in that matrix. Finally a new data set, $\boldsymbol{Traces_m}$ is derived using

$$\boldsymbol{Traces_m} = (\boldsymbol{V_m}' \times \boldsymbol{Traces_{adjust}}')' \tag{1}$$

where $'$ denotes a matrix transpose operation.

PCA has been studied extensively [23], [42]–[44] in SCA context, but rarely so when it comes to deep-learning techniques. In contrast to template attacks, where dimension reduction is necessary, deep learning can handle large dimensions, and the benefit of PCA comes from the projection of trace samples to their principal subspace, and not particularly from the pruning (Section IV-A).

D. Dynamic Time Warping

Misalignment between captured power traces can occur due to inaccurate triggering [45] or countermeasures adopted by device manufacturers such as frequency scaling [46] or random insertion of dummy operations [47]. As pointed out in [45], time-series matching algorithms, such as DTW, originally adopted for alignment in speech recognition systems [48], can be beneficial to realign them in an attack scenario where only a limited number of traces from target device can be collected. However, in contrast to [45], where DTW has been used as a preprocessing step before performing a nonprofiled DPA attack or a profiled template attack [8], we propose to use it as a preprocessing step for a neural network classifier.

DTW resamples two traces such that matching parts of them are located at the same time index after the process, thus reducing distance (Euclidean/absolute) between them to a minimum.

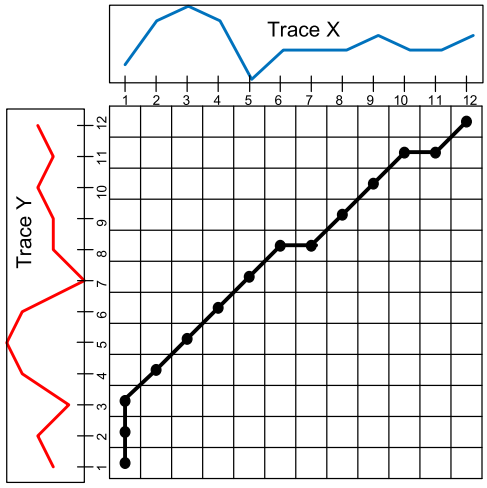

Fig. 3. Overview of DTW. Warp path for traces X and Y shows how both the traces are nonlinearly resampled to ensure matching. For example, samples 1 and 2 in trace Y are absent in trace X. Hence, DTW samples X at the sample 1 multiple times. Then, samples 3–8 in the trace Y match with the samples 1–6 in trace X, and so on. In this way, two traces are realigned such that the absolute or the Euclidean distance between them is minimized.

The resampled indices in both traces form a *warp path*. Note that as DTW can align only one trace with respect to another, we need a *reference trace* for realignment. Fig. 3 illustrates a warp path $W = \{z(k) : 1 \leq k \leq K\}$ that is set of $x(k)$ and $y(k)$ indices, $z(k) = (x(k), y(k))$ of two traces X and Y, under the following constraints:

$$z(k+1) = (x(k+1), y(k+1)) = \begin{cases} (x(k), y(k)+1), & \text{or} \\ (x(k)+1, y(k)), & \text{or} \\ (x(k)+1, y(k)+1)) \end{cases}$$

and

$$x(1) = y(1) = 1, \quad x(K) = y(K) = T, \; T \leq K < 2T$$

where

$$T = \text{number of samples in } X \text{ and } Y.$$

DTW algorithm tries to find a warp path W that results in the minimum cost L given as

$$L(X, Y) = \frac{1}{2T} \min_{W} \sum_{k=1}^{K} d(z(k))c(k)$$

where

$$d(z(k)) = |X(x(k)) - Y(y(k))|$$

and

$$c(k) = x(k) - x(k-1) + y(k) - y(k-1).$$

This process results in stretching both the traces (X and Y) by resampling them. The resulting traces become perfectly aligned.

III. Single-Trace Cross-Device Power SCA Using Neural Networks: Performance and Limitations

In this section, we present the architectures of our designed MLP and CNN, along with the empirical choice of hyper-parameters, and the performance and limitations of these single-trace attack without any preprocessing.

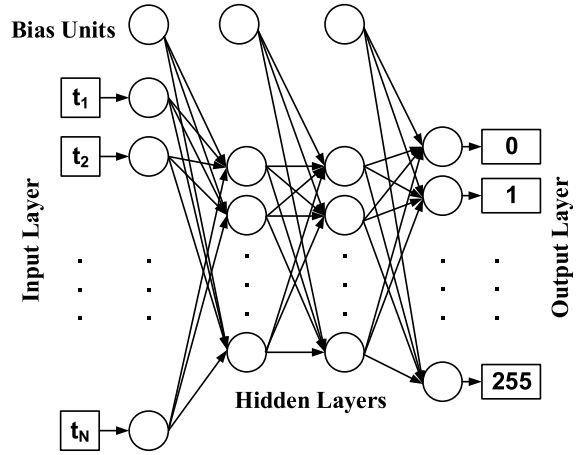

Fig. 4. Architecture of the proposed MLP for cross-device side-channel attack. The input layer consists of $N = 3000$ neurons. The first FC hidden layer consists of 100 hidden neurons, followed by batch normalization, ReLU activation, and a dropout layer. The second hidden layer is similar without the dropout layer. Finally, the output layer has 256 neurons for predicting the correct keybyte utilizing the softmax function.

A. Experimental Setup

We performed our experiments on CW308T-XMega, a microcontroller-based SCA platform from ChipWhisperer [24]. This microcontroller-based target board houses an 8-bit Atmel AVR XMega128 microcontroller running software AES-128. To capture traces from the target device, ChipWhisperer provides a platform CW308T UFO board, and a capture setup CW Lite Capture using an on-board ADC. This setup allows to send program, plaintext, and key to XMega Target board and record captured traces directly from a personal computer (Fig. 2). The on-board XMega microcontroller operates at the frequency of 7.37 MHz, and CW Lite Capture hardware captures traces at four times of that frequency, at 29.48 MHz. Power consumption is measured by inserting a small resistor (500 mΩ) in series with power supply and measuring the voltage drop across it. As this measured voltage corresponds directly to instantaneous current drawn from the dc power supply, it can be treated as the power trace. We utilize a *chosen plaintext* scenario for our attack, where we keep the plaintext fixed to a chosen value ($0\times00\ldots00$, i.e., all 16 bytes are 0s). We collect 10k traces (3k time samples per trace) from each of the 30 devices by varying the first keybyte in all 256 possible combinations (from 0 to 255) maintaining a uniform distribution [$(10k/256) \approx 40$ traces for each possible keybyte value], and rest of the keybytes randomly. We attack only the first keybyte in all our results, but the attack can be carried out for all other keybytes in the same manner. Neural network models were implemented using MATLAB and Python, using the keras [49] library with tensorflow [50] as backend.

B. Architecture of Multilayer Perceptron

Fig. 4 shows the architecture of our designed MLP for the 256-class classification, which is similar to the one presented in [21]. The size of the raw or processed traces determines

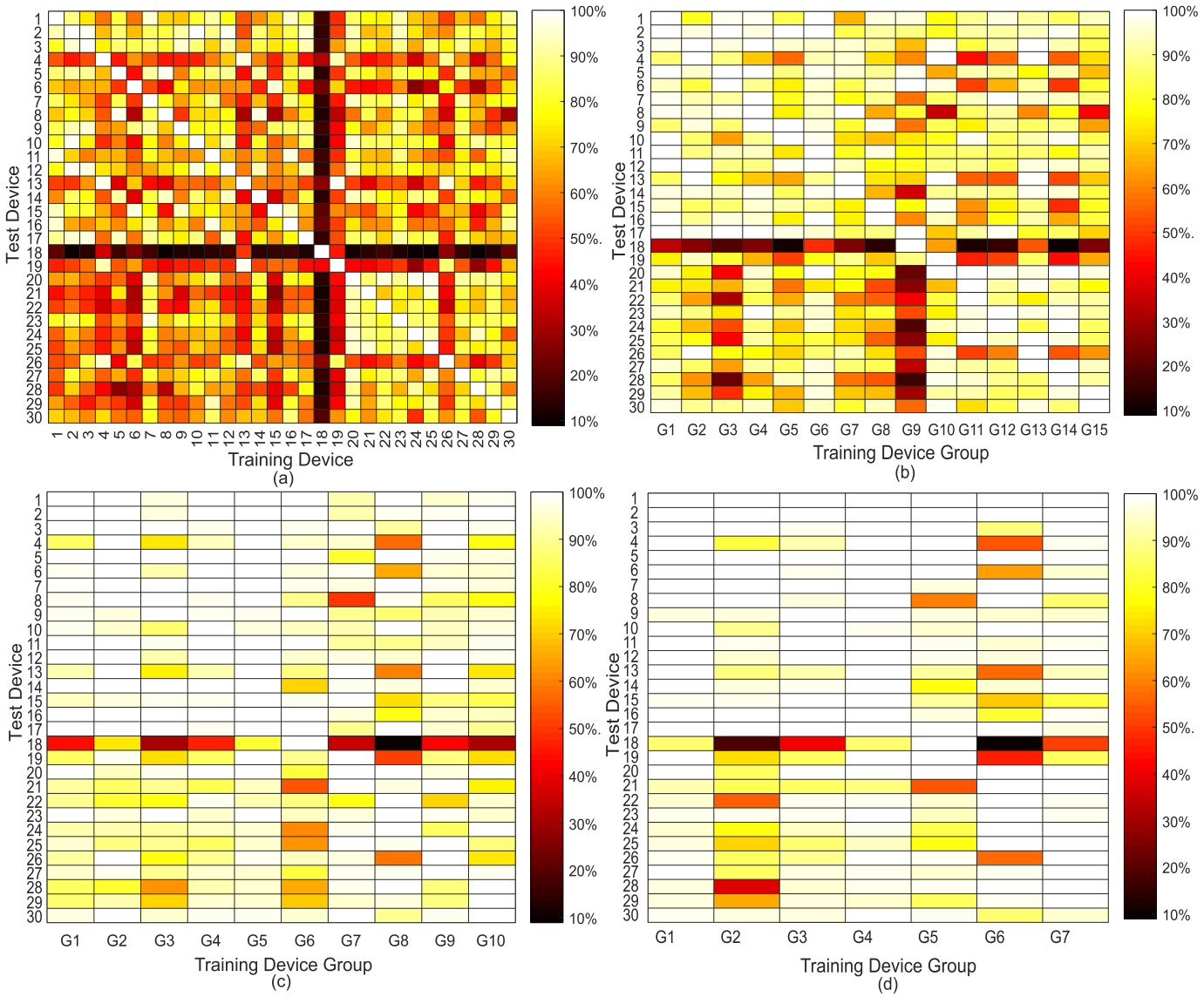

Fig. 5. Test accuracy of MLP classifier when the number of devices in training set is (a) one, (b) two, (c) three, and (d) four. As can be seen from (a), although the test accuracy for the *same device attack* is very high, it varies widely in case of a *cross device attack*. In particular, note how Device #18 has 100% test accuracy when the classifier is trained with that device, but a very low accuracy when the neural network is trained with other devices. (b)–(d) Illustrate the improvement in test accuracy with an increase in the number of training devices.

the size of the input layer of the MLP. Two fully connected layers each consisting of 100 neurons form the hidden layers. Activation function for the layers is chosen as ReLU. A dropout layer after the first hidden layer having a percentage dropout of 10% has been chosen to aid generalization. L_2 regularization seemed to have little effect. We keep it fixed at 10^{-4} for all the iterations. Also, the minibatch size is kept at 256, the network has been evaluated after being trained for 100 epochs. It was observed that higher number of hidden neurons and layers led to overfitting to the training data, resulting in relatively poor performance in the test set. Also, the choice of the batch size is a critical issue. It was observed that small batch sizes led to high test accuracy. To optimize the network, we train the MLP with 8k training traces for single-device training and optimize the performance by validating against a test set of the remaining 2k traces from the same device.

C. Performance of MLP

In this section, we evaluate the performance of MLP. The problem with one device training is that the neural network overfits to that device-specific leakage and cannot generalize to new data set from a different device. As shown in Fig. 5(a), although the same-device attack performance of MLP is very high ($\geq$99.99%), cross-device attack performance is relatively poor, averaging at 61.98% across the 30 devices after training for 100 epochs (Table II). Also, we note from Fig. 5(a) that Device 18 is clearly an outlier, although we cannot see much deviation for Device 18 in Fig. 1. In Fig. 1, we only observed the distribution for one specific dimension of a 3000-D trace. That is why, it does not give us the whole picture. To find out why Device 18 is an outlier in this experiment, the average for each time sample of all 8k traces (with 3000 samples each) has been calculated from the training set for all 30 devices in our data set. Then, the mean μ and standard deviation σ have

TABLE II

CROSS-DEVICE ATTACK PERFORMANCE OF DEEP-LEARNING-BASED METHODS FOR DIFFERENT TRAINING SCENARIOS*

Number of Training Devices	*Test Accuracy (%)*								
	MLP			*PCA − MLP*			*CNN*		
	Average	Maximum	Minimum	Average	Maximum	Minimum	Average	Maximum	Minimum
1	61.98	98.70	2.95	90.09	99.94	53.18	29.97	44.86	10.09
2	79.14	99.92	4.47	96.65	**99.99**	71.28	47.75	74.42	21.27
3	90.76	99.93	8.93	99.37	**99.99**	**90.82**	78.69	98.93	51.15
4	91.72	99.95	8.02	**99.43**	**99.99**	89.21	80.39	94.63	60.08

**Does not include Test Accuracy for Devices used in Training Set*

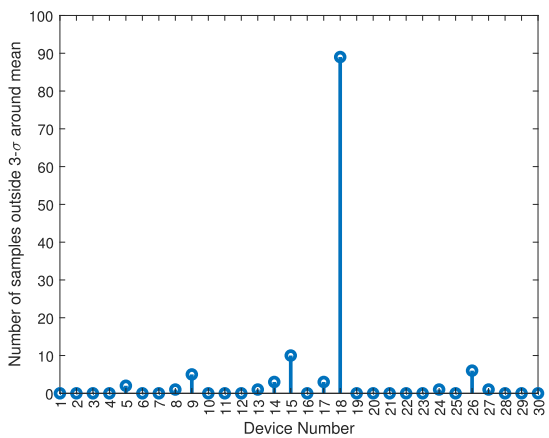

Fig. 6. Rationale behind Device 18 being an outlier in Fig. 5(a). Approximately 90 samples of averaged trace for Device 18 fall outside the 3σ range around the mean for averaged traces across all devices.

been calculated for each time sample of the averaged traces from all 30 devices. Assuming an approximately Gaussian distribution, 99.7% of the time samples of the averaged traces for each device should fall within three standard deviation around the mean ($\mu + 3\sigma$ and $\mu - 3\sigma$). Then, the number of samples (out of 3000) of averaged traces for each device, which fall outside this range, has been counted. In Fig. 6, the result has been illustrated, where Device 18 is certainly an outlier, which explains why we obtained poor test accuracy for Device 18 when the neural network was trained with traces from other devices, and vice versa.

To eliminate the problem of low test accuracy of MLP with single-device training, we perform an empirical evaluation of the multidevice training method [21], so that the neural network learns from the device manufacturing and packaging variations and generalizes to test traces from a new device. It should be noted that although this would strengthen the adversary, the attack becomes more difficult to implement, as it assumes more control on the part of the adversary. Nevertheless, such a multidevice training improves the average test accuracy to as high as 91.72% as illustrated in Fig. 5(b)–(d) and Table II. Note that, we merge the data sets for different devices before multidevice training and use 20k, 30k, and 40k traces in training and validation set for the two-device, three-device, and four-device training, respectively. To incorporate cross validation, but at the same time bearing in mind the training time required for an exhaustive one, we construct several training device groups and use them for all subsequent analysis. Training groups for multidevice training are formed using the formulation: $\{G_j(i)\} = \{D(k), D(k+1), ..., D(k+j-1) \mid k = (i-1)j+1 \text{ and } k+j-1 \leq 30\}$, where $\{G_j(i)\}$ is the ith group used in j device training, and $D(k)$ is the kth device in our data set. Also, note that further increasing the number of traces (or more devices) in training set did not improve test accuracy.

Although the improvements with multidevice training seem intuitive, in this paper, we further investigate the factor that could have led to this. In this regard, we seek inspiration from a POI selection method, namely, DOMs [8], [13], typically used in template attacks. Using the sum of the absolute value of pairwise differences between mean of traces for different keys (keeping plaintext fixed), we observed that the samples #96 and #148 are two such POIs. To achieve a more accurate result, template attacks typically identify 10–20 such POIs and calculate a probability density function (PDF) using a multivariate Gaussian distribution approximation. Although such a high-dimensional template would give us more accurate results, we concentrated on a simple bivariate analysis (as it can be visualized in 2-D) to observe if a pattern could be identified from the distribution. The PDF for multivariate normal distribution of k-variables is given as

$$f_{\boldsymbol{x}} = \frac{exp(-1/2(\boldsymbol{x}-\boldsymbol{\mu})^T \ \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}))}{\sqrt{(2\pi)^k|\boldsymbol{\Sigma}|}}$$

where $\boldsymbol{\Sigma}$ = Covaraince Matrix and $\boldsymbol{\mu}$ = Mean Vector.

Covariance matrix and mean vector for the bivariate normal joint density function have been calculated from experimentally measured data. Fig. 7 shows the boundary regions of PDFs. It can be seen that data from different keys lead to different distributions for the same device. Also, traces from a single device do not span the whole distribution of 30 devices, but with four devices, most of it can be spanned. This is plausibly why our trained MLP model achieved a high average test accuracy with four-device training.

D. Limitation of Only MLP-Based Attack

Note from Table II that the minimum test accuracy does not improve much with the increasing number of training devices, but the average test accuracy improves significantly. This motivates us to investigate further into preprocessing techniques such as PCA to improve the minimum test accuracy

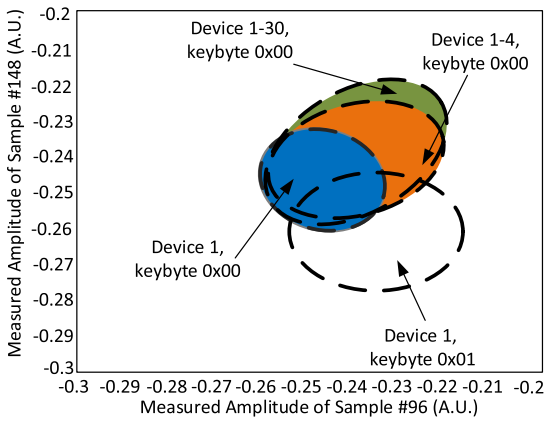

Fig. 7. Boundary of bivariate PDF for different scenarios. Two of the most prominent leakage samples from the raw power traces are chosen as the variables for constructing the bivariate distribution. Note that the PDFs are different for different keybyte values. Also, as the number of devices increases from 1 to 4, the sample PDF for a specific keybyte value (0×00) approximates the total PDF of the 30 devices more accurately.

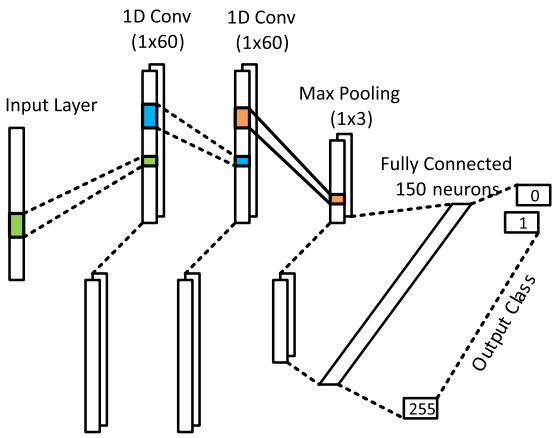

Fig. 8. Architecture of 1-D CNN. Samples of raw traces are directly fed to the input layer. Two 1-D convolutional layers act as filters and extract high-level features from raw traces. Next, a max-pooling layer reduces the number of dimensions by subsampling. The outputs of max-pooling layer are flattened and provided as inputs to an FC layer, which connects the final classification Layer.

(Section IV) and utilize the multidevice training to prevent overfitting and aid generalization.

E. Architecture and Performance of Convolutional Neural Network

We designed a 1-D CNN architecture (Fig. 8), as shown in the recent works [10], [18] to have very high accuracy in the presence of countermeasures. CNN architecture presented here is different from those works, as it is designed for our new data set. We gathered inspiration from VGG-Net [51], but used a much shallower network. Such a choice again depends on the data set.

We kept the same input layer as in the MLP, i.e., 3000 neurons. In this architecture, the first convolutional layer has 70 filters, each having a kernel size of 60, with a default stride of 1, and with ReLU activation function. The second convolutional layer is exactly the same. Then, a max-pooling layer with a pool size of 3 was used. A subsequent layer flattens the output of max-pooling layer, followed by an FC layer with 150 neurons, a batch normalization layer, and finally the classification layer. A dropout factor of 20% after the flatten layer, and 10% after the batch normalization layer reduced overfitting. The parameters have been optimized by validating on a test set from the same-device, similar to the approach in the case of the MLP architecture. Note that CNNs require a larger data set to generalize well to new data, and we augmented the available data set by introducing normally distributed noise (with 10^{-10} standard deviation) to trace samples, and thus increasing a number of training traces to 60k in all training scenarios. We evaluate the performance of CNN for single-device training and multidevice training after 20 epochs. Fig. 9(a) illustrates the result of single-device training for CNN, and we can see that this CNN network achieves very high test accuracy for traces from the same-device, but cross-device attack accuracy is low. Fig. 9(b) shows the improvement in cross-device attack accuracy with multi-device training. However, comparing Fig. 5 to Fig. 9, we can observe that MLP outperforms the CNN in the cross-device attack performance (although comparable for the same device attack) for both single-device and four-device training.

Table II summarizes the comparison of cross-device performance between the MLP and the CNN models with multidevice training. Compared to the MLP, CNN provides better minimum accuracy, but lower average accuracy across the 30 devices taken from two different batches. Hence, we choose MLP as our desired classifier for the next sections and develop strategies to achieve high accuracies for all the 30 devices.

IV. Performance of MLP With PCA-Based Preprocessing (PCA-MLP)

In this section, we evaluate the effect of PCA [33] as a preprocessing step to enhance the performance of cross-device attacks. Fig. 10(a) shows the amplitude of the features extracted from raw traces using (1). Note that, the eigenvectors with higher eigenvalues point to the direction of higher variance in data. As a result, in the transformed trace [Fig. 10(a)], the samples on the left have higher amplitudes, and as we proceed to the right, the amplitudes decrease. From Fig. 10(b), we can see that the first 60 time samples contribute the most to the total variance. Also, it has been observed that, for this data set, 99% of total variance is contributed by the first 370 principal components.

A. Performance of PCA-MLP

As seen earlier, MLP-based classifier without preprocessing achieves good average accuracy (>90%) with multidevice training, but the minimum accuracy is as low as ∼8% (Fig. 5 and Table II). In this section, we show that with PCA as a preprocessing step, the performance of the neural network classifier is substantially improved due to the projection of raw trace samples to their principal subspace. For a principal subspace, coordinate axes are ordered in a way that they point to the direction of the maximal variance in data [23].

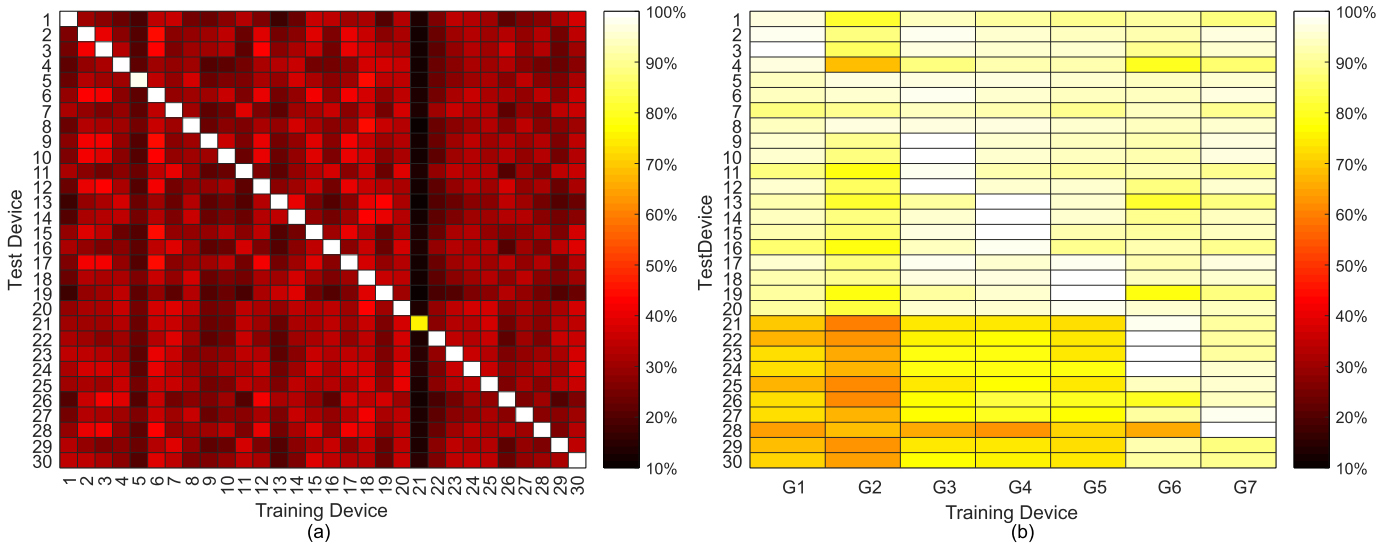

Fig. 9. Test accuracy of CNN classifier when the number of devices in training set is (a) one and (b) four. As can be seen from (a), although the test accuracy for the *same device attack* is very high, it is extremely low in the case of a *cross device attack*. In (b), improvement in test accuracy with an increase in number of training devices is seen, but compared to MLP, the average accuracy still remains lower.

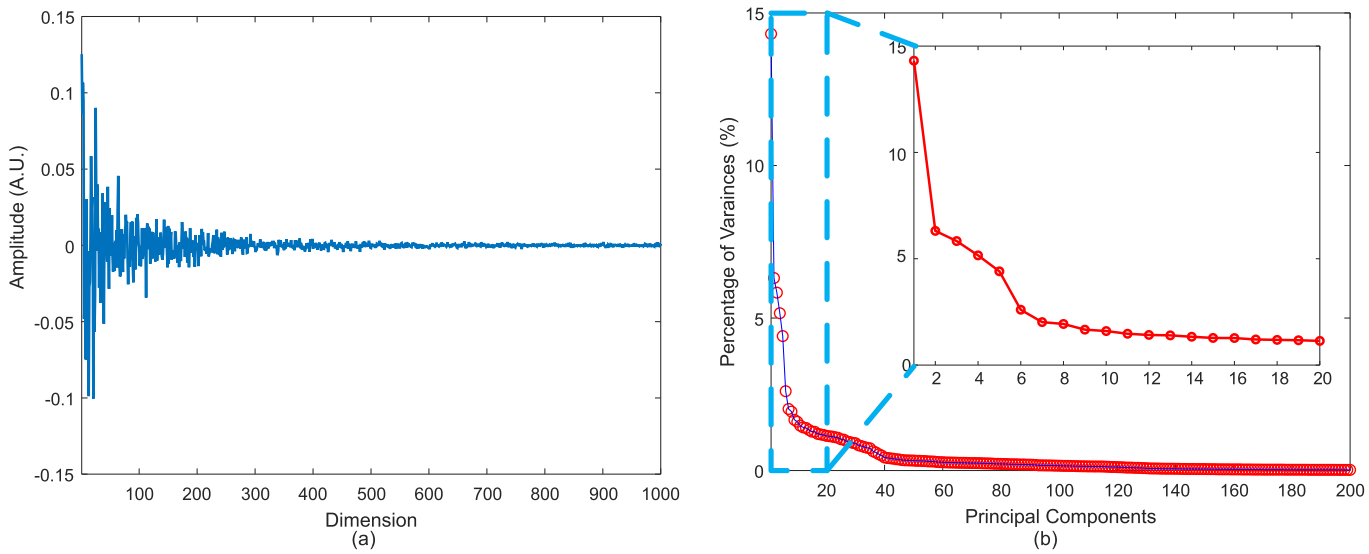

Fig. 10. PCA of raw traces. (a) Transformed trace after PCA (first 1000 time samples). Note that amplitudes are lower with higher dimensions. (b) Contribution of each principal component as percentage of total variances (first 200 principal components). The zoomed-in region corresponds to the first 20 samples. Note that the first 60 samples contribute the most to the total variance.

Fig. 11(b) shows that both the minimum and the average test accuracy of PCA-MLP improve significantly with four-device training compared to only MLP [Fig. 5(d)]. Also, as summarized in Table II, it can be seen that the average as well as the minimum test accuracy improve going from single-device to multidevice training. Since with four devices, the PCA-MLP model achieves the best average accuracy, we chose 4 as the number of devices for multidevice training. It should be noted that the same number of training traces was used to train this model as reported in Section III. Fig. 11(a) illustrates that reducing a number of dimensions did not improve the average test accuracy in our case. Hence, we used all the 3000 principal components in the analysis presented in this section, although higher dimensions have less informative features.

B. Limitation of PCA-MLP

One inherent assumption in Sections III and IV-A was that the traces were all perfectly aligned (as they were collected using the ChipWhisperer capture setup), which may not always be the case in a practical scenario due to faulty triggering. Fig. 12(a) shows the traces when they are misaligned. The limitation of PCA and MLP is that the traces need to be perfectly aligned. This motivates us to investigate ways to realign traces so that the benefits of the high classification accuracy for PCA-MLP can be utilized. Cagli *et al.* [10] proposed using CNN in case of misaligned traces, but as PCA-MLP showed better cross-device attack performance, we chose to use PCA-MLP, and to account for the misalignment, we adopted DTW as a preprocessing step (Section V).

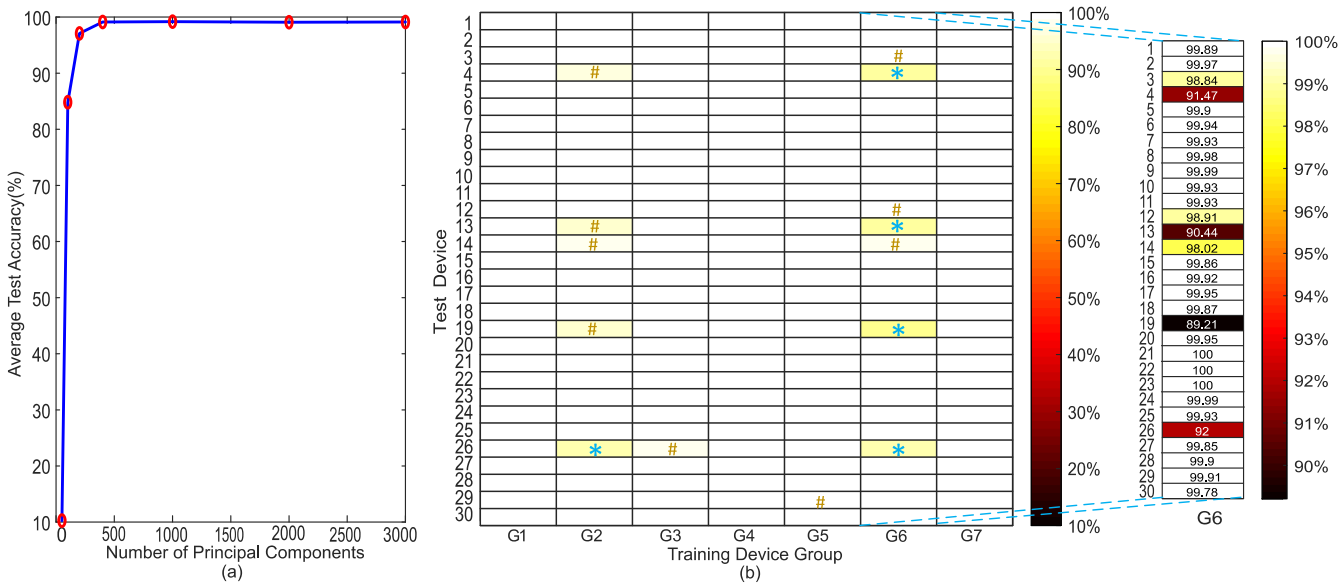

Fig. 11. Performance of PCA-MLP. (a) Average test accuracy (%) versus the number of principal components used (for Training Device Group G1). Test accuracy does not improve on removing the principal components with a lower contribution in the percentage variance. Hence, we chose to include all the 3000 principal components in the PCA-MLP model. (b) Test accuracy after training PCA-MLP with four devices shows drastic improvement in the minimum cross-device accuracy. Note that "*" symbol has been used to highlight the cases with 89%–94% test accuracy, and "#" symbol for 95%–98% test accuracy.

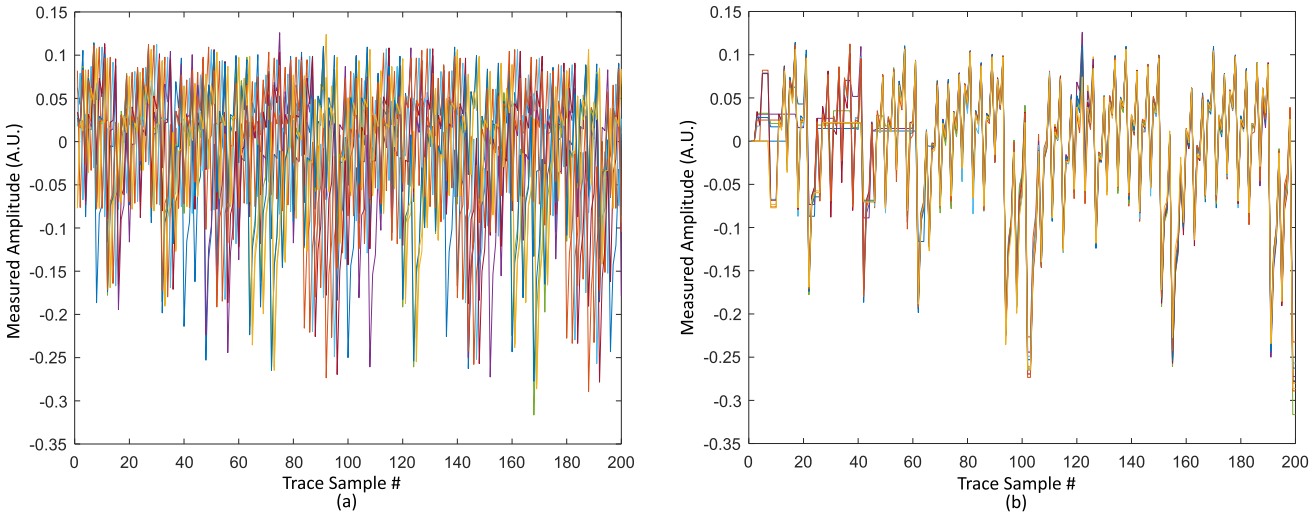

Fig. 12. Ten traces superimposed on each other (first 200 samples shown) when they are (a) misaligned randomly up to 50 time samples and (b) realigned using DTW.

V. Dynamic Time Warping as a Preprocessing for Misaligned Traces

In this section, we show how DTW can be used to realign misaligned traces due to a fault in triggering. As ChipWhisperer platform perfectly synchronizes each capture event, the traces obtained from CW308T-XMega target board are perfectly aligned with each other. To simulate the event of trace misalignment, we artificially create up to 50 time-sample misalignments to evaluate the performance of the proposed DTW-PCA-MLP architecture. Misalignment has been created by shifting traces by a random number of samples, in the same manner presented in [18]. Fig. 12(a) shows such a collection of ten traces superimposed on each other. Note that the traces are randomly misaligned up to 50 samples. We chose five devices from our set of 30 devices and created misaligned data sets $M1$–$M5$. Then, we evaluated the performance of our proposed approach on a $cross-device\ attack$ to show its effectiveness by training on four devices from the misaligned data set and testing on the other one and performing cross validation for all the possible combinations.

A. Implementation of DTW-PCA-MLP

As mentioned in Section II, DTW method requires a reference trace to realign another trace. Obtaining such a reference trace is possible for an adversary from the device/s he has in his possession. In our experiments, we use a reference trace that has a 3000 time-sample window, containing all relevant samples. The network has been trained on four devices and

TABLE III
PERFORMANCE COMPARISON OF DTW-PCA-MLP WITH CNN FOR MISALIGNED TRACES

Training Set	*Test Set*	*TestAccuracy*(%)			
		DTW-PCA-MLP	CNN	DTW-CNN	DTW-PCA-CNN
M1-M4	M5	99.80	87.05	88.91	89.63
M1-M3,M5	M4	99.71	88.37	95.53	93.22
M1-M3,M4-M5	M3	99.69	88.72	92.64	90.16
M1,M3-M5	M2	99.94	78.98	92.41	95.66
M2-M5	M1	98.86	80.61	92.44	95.40

TABLE IV
QUALITATIVE COMPARISON BETWEEN DIFFERENT DEEP-LEARNING-BASED ATTACK METHODS

		Average Test Accuracy of Different Methods*			
		MLP	**PCA-MLP**	**DTW-PCA-MLP**	**CNN**
Same-Device Attack	**Aligned Traces**	Very High	Very High	Very High	Very High
	Misaligned Traces	Very Low	Very Low	Very High	Very High
Cross-Device Attack with multi-device training	**Aligned Traces**	High	Very High	Very High	High
	Misaligned Traces	Very Low	Very Low	Very High	High

**Evaluated on the same data set comprising of traces from 30 identical microcontroller devices*

Algorithm 1 Algorithm for DTW-Based Trace Realignment

Input: Misaligned Trace Matrix ($\boldsymbol{Tr}_m^{M\times N}$), Reference Trace ($\boldsymbol{tr}_{ref}^{1\times N}$)
Output: Aligned Trace Matrix ($\boldsymbol{Tr}_a^{M\times W_M}$), Modified Reference trace ($\boldsymbol{tr}_{refm}^{1\times W_M}$)

$W_0 \leftarrow N$
for $i \leftarrow 1$ to M **do**
  $[\boldsymbol{x}_i^{1\times W_i}, \boldsymbol{y}_i^{1\times W_i}] \leftarrow DTW(\boldsymbol{Tr}_m^{M\times N}(i,:), \boldsymbol{tr}_{ref}^{1\times W_{i-1}})$
  if $i > 1$ **then**
    $\boldsymbol{Tr}_a^{M\times W_i} \leftarrow \boldsymbol{Tr}_a^{M\times W_{i-1}}(1:i-1, \boldsymbol{y}_i^{1\times W_i})$
  end if
  $\boldsymbol{Tr}_a^{M\times W_i}(i,:) \leftarrow \boldsymbol{Tr}_m^{M\times N}(i, \boldsymbol{x}_i^{1\times W_i})$
  $\boldsymbol{tr}_{refm}^{1\times W_i} \leftarrow \boldsymbol{tr}_{refm}^{1\times W_{i-1}}(1, \boldsymbol{y}_i^{1\times W_i})$
end for
return $\boldsymbol{Tr}_a^{M\times W_M}$, $\boldsymbol{tr}_{refm}^{1\times W_M}$

tested on one, as shown in Table III. We keep the neural network architecture for MLP the same as in previous sections, apart from a change in the size of the input layer. With experiments, it has been observed that for realigned data set, dimensionality reduction after PCA improved test accuracy. This can be partly due to the fact that DTW resamples the traces to find the best match, and in doing that, some samples are copied multiple times. We empirically found that 600 features led to the best test accuracy. Therefore, we modified the number of input neurons to 600, but the rest of the architecture remains the same. Algorithm for DTW-based trace realignment is shown in Algorithm 1. Using this algorithm, traces have been realigned [Fig. 12(b)].

B. Performance Comparison Among Different Methods for Misaligned Traces

After trace realignment using DTW, subsequent PCA-based preprocessing and MLP-based classifier resulted in ≥98.86% test accuracy, compared to the CNN (which has been reoptimized to deal with the misalignment) with ≥78.98% test accuracy. The minimum difference between the test accuracy of DTW-PCA-MLP and CNN is 10.97% for the test set M3. We expect this trend to continue if extended to all 30 devices of our data set, based on results presented in Table III. Consequently, based on the results, we can clearly see that DTW-PCA-MLP is a convenient method for cross-device profiled-attack for misaligned traces, not only because of high average test accuracy but also due to its simpler architecture and less effort on the choice of hyperparameters and lower training time compared to CNN. For the sake of completeness, we also evaluate the performance of CNN with only DTW and both DTW and PCA-based preprocessing. CNN is expected to benefit from realignment through DTW, although CNN is believed to be able to handle misalignments intrinsically. To see if CNN benefits from DTW (and PCA) when traces are misaligned, we conducted the experiment and Table III summarizes the results. From Table III, we see that the inclusion of DTW certainly helps to improve test accuracy in all cases for CNN, but inclusion of PCA after DTW may improve test accuracy in some cases compared to DTW-CNN, but that does not generalize well across all sets.

VI. RELATIVE TIMING PERFORMANCE COMPARISON BETWEEN DIFFERENT SCA APPROACHES

In comparison to profiled attacks, nonprofiled attacks, such as CPA/DPA, would require at least tens to thousands of traces to correctly identify the key. As shown in [21], even in low signal-to-noise (SNR) scenarios, the benefit of using deep-learning techniques persists, as CPA requires $\sim 10\times$ more traces for the same level of accuracy.

Training time depends on network complexity, chosen hyperparameters, size of the data set, software platform (e.g., tensorflow, pytorch, etc.), and last but not least, the hardware used to train. We have conducted the training and testing phase for MLP and CNN on the same hardware platform, and we observed that for the same training set (four training devices used), the batch size (256), and the number of epochs (100), CNN requires 6× more time to achieve the same level of accuracy (≥99%) compared to MLP (453.4 s compared to

75.6 s), which can be attributed to larger number of parameters to train. Also, during the testing phase, CNN takes 2.5× longer time to generate the output class (0.442 s compared to 0.177 s).

We would like to mention that as the training and testing are done off-line, the training time for the network is not crucial to launch a successful attack, whereas the number of traces is required to recover a key. An adversary may record all the ciphertexts and corresponding power traces, and later break them off-line. As a result, the high probability of success using deep-learning techniques for a single-trace attack is of a major concern.

VII. Conclusion

This paper presents a practical cross-device attack using deep-learning methods even in the presence of misalignment in the captured traces and with significant interdevice variations. Both of these practical issues are challenging to deal with while implementing a cross-device attack. This paper demonstrates how multidevice training improves average test accuracy (e.g., from 61.98% to 91.72% in case of MLP). Moreover, it presents how such attacks can be further improved using preprocessing methods, such as PCA and DTW, and identifies the best deep-learning-based approach. PCA-based preprocessing improves the minimum test accuracy by 10×, and DTW-based preprocessing allows subsequent PCA-MLP to maintain its high test accuracy (up to 99.94%). Although deep-learning techniques are more interesting when leakage is harder to model, we note that success of such techniques lies in data, and preprocessing can help to clean the data for more efficient learning.

Table IV summarizes the findings by presenting a qualitative comparison of average test accuracy of different methods for both same-device attack and a cross-device attack with multidevice training. In our experiments, CNN maintained a high accuracy in all scenarios, but the best cross-device performance was obtained using the proposed DTW-PCA-MLP. Moreover, compared to the CNN, the proposed DTW-PCA-MLP has a much simpler architecture (less number of tunable parameters) and also requires shorter training time.

Going forward, we would like to study the feasibility of the proposed deep-learning attack on 32-bit ARM microcontrollers and field-programmable gate array (FPGA)-based platforms, which have more widespread use.

References

[1] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Proc. Annu. Int. Cryptol. Conf.* Berlin, Germany: Springer, 1999, pp. 388–397.

[2] D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi, "The EM side—Channel(s)," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst.* Berlin, Germany: Springer, 2002, pp. 29–45.

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

[4] K. Gandolfi, C. Mourtel, and F. Olivier, "Electromagnetic analysis: Concrete results," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst.* Berlin, Germany: Springer, 2001, pp. 251–261.

[5] M. G. Kuhn, "Optical time-domain eavesdropping risks of CRT displays," in *Proc. IEEE Symp. Secur. Privacy*, May 2002, pp. 3–18.

[6] J. Loughry and D. A. Umphress, "Information leakage from optical emanations," *ACM Trans. Inf. Syst. Secur.*, vol. 5, no. 3, pp. 262–289, 2002.

[7] D. Asonov and R. Agrawal, "Keyboard acoustic emanations," in *Proc. IEEE Symp. Secur. Privacy*, May 2004, pp. 3–11.

[8] S. Chari, J. R. Rao, and P. Rohatgi, "Template attacks," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst.* Berlin, Germany: Springer, 2002, pp. 13–28.

[9] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst.* Berlin, Germany: Springer, 2004, pp. 16–29.

[10] E. Cagli, C. Dumas, and E. Prouff, "Convolutional neural networks with data augmentation against jitter-based countermeasures," in *Proc. CHES*, 2017, pp. 45–68.

[11] L. Lerman, R. Poussier, O. Markowitch, and F.-X. Standaert, "Template attacks versus machine learning revisited and the curse of dimensionality in side-channel analysis: Extended version," *J. Cryptograph. Eng.*, vol. 8, no. 4, pp. 301–313, 2018.

[12] D. Oswald and C. Paar, "Breaking mifare DESFire MF3ICD40: Power analysis and templates in the real world," in *Proc. CHES*, 2011, pp. 207–222.

[13] M. O. Choudary and M. G. Kuhn, "Efficient, portable template attacks," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 2, pp. 490–501, Feb. 2018.

[14] D. P. Montminy, R. O. Baldwin, M. A. Temple, and E. D. Laspe, "Improving cross-device attacks using zero-mean unit-variance normalization," *J. Cryptogr. Eng.*, vol. 3, no. 2, pp. 99–110, 2013.

[15] N. Hanley, M. O'Neill, M. Tunstall, and W. P. Marnane, "Empirical evaluation of multi-device profiling side-channel attacks," in *Proc. IEEE Workshop Signal Process. Syst. (SiPS)*, Oct. 2014, pp. 1–6.

[16] T. Bartkewitz and K. Lemke-Rust, "Efficient template attacks based on probabilistic multi-class support vector machines," in *Proc. Int. Conf. Smart Card Res. Adv. Appl.* Berlin, Germany: Springer, 2012, pp. 263–276.

[17] L. Lerman, G. Bontempi, and O. Markowitch, "Power analysis attack: An approach based on machine learning," *Int. J. Appl. Cryptogr.*, vol. 3, no. 2, pp. 97–115, 2014.

[18] R. Benadjila, E. Prouff, R. Strullu, E. Cagli, and C. Dumas, "Study of deep learning techniques for side-channel analysis and introduction to ASCAD database," Cryptol. ePrint Arch., San Diego, CA, USA, Tech. Rep. 2018/053, 2018. [Online]. Available: https://eprint.iacr.org/2018/053.pdf

[19] H. Maghrebi, T. Portigliatti, and E. Prouff, "Breaking cryptographic implementations using deep learning techniques," in *Proc. Int. Conf. Secur. Privacy Appl. Cryptogr. Eng.* Cham, Switzerland: Springer, 2016, pp. 3–26.

[20] Z. Martinasek, P. Dzurenda, and L. Malina, "Profiling power analysis attack based on MLP in DPA contest V4.2," in *Proc. 39th Int. Conf. Telecommun. Signal Process. (TSP)*, Jun. 2016, pp. 223–226.

[21] D. Das, A. Golder, J. Danial, S. Ghosh, A. Raychowdhury, and S. Sen, "X-DeepSCA: Cross-device deep learning side channel attack," in *Proc. 56th Annu. Design Automat. Conf.*, 2019, p. 134.

[22] M. Carbone *et al.*, "Deep learning to evaluate secure RSA implementations," *Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2019, no. 2, pp. 132–161, Feb. 2019.

[23] C. Archambeau, E. Peeters, F.-X. Standaert, and J.-J. Quisquater, "Template attacks in principal subspaces," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst.* Berlin, Germany: Springer, 2006, pp. 1–14.

[24] C. O'Flynn and Z. D. Chen, "Chipwhisperer: An open-source platform for hardware embedded security research," in *Proc. Int. Workshop Constructive Side-Channel Anal. Secure Design*. Cham, Switzerland: Springer, 2014, pp. 243–260.

[25] C. Rechberger and E. Oswald, "Practical template attacks," in *Proc. Int. Workshop Inf. Secur. Appl.* Berlin, Germany: Springer, 2004, pp. 440–456.

[26] E. Oswald and S. Mangard, "Template attacks on masking—Resistance is futile," in *Proc. Cryptographers' Track RSA Conf.* Berlin, Germany: Springer, 2007, pp. 243–256.

[27] L. Lerman, G. Bontempi, and O. Markowitch, "A machine learning approach against a masked AES," *J. Cryptograph. Eng.*, vol. 5, no. 2, pp. 123–139, Jun. 2015.

[28] A. Heuser and M. Zohner, "Intelligent machine homicide," in *Proc. Int. Workshop Constructive Side-Channel Anal. Secure Design*. Berlin, Germany: Springer, 2012, pp. 249–264.

[29] L. Lerman, G. Bontempi, S. B. Taieb, and O. Markowitch, "A time series approach for profiling attack," in *Proc. Int. Conf. Secur. Privacy Appl. Cryptogr. Eng.* Berlin, Germany: Springer, 2013, pp. 75–94.

[30] R. Gilmore, N. Hanley, and M. O'Neill, "Neural network based attack on a masked implementation of AES," in *Proc. HOST*, May 2015, pp. 106–111.

[31] Z. Martinasek, J. Hajny, and L. Malina, "Optimization of power analysis using neural network," in *Proc. Int. Conf. Smart Card Res. Adv. Appl.* Cham, Switzerland: Springer, 2013, pp. 94–107.

[32] M. Renauld, F.-X. Standaert, N. Veyrat-Charvillon, D. Kamel, and D. Flandre, "A formal study of power variability issues and side-channel attacks for nanoscale devices," in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn.* Springer, 2011, pp. 109–128.

[33] I. Jolliffe, "Principal component analysis," in *International Encyclopedia Statistical Science*. Cham, Switzerland: Springer, 2011, pp. 1094–1096.

[34] M. Müller, "Dynamic time warping," in *Information Retrieval for Music and Motion*. 2007, pp. 69–84.

[35] *DPA Contest, 2008–2009*, TELECOM ParisTech SEN Res. Group, 2009.

[36] *DPA Contest, 2013–2014*, TELECOM ParisTech SEN Res. Group, 2014.

[37] 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," *Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2019, no. 3, pp. 148–179, May 2019.

[38] G. Yang, H. Li, J. Ming, and Y. Zhou, "Convolutional neural network based side-channel attacks in time-frequency representations," in *Proc. Int. Conf. Smart Card Res. Adv. Appl.* Cham, Switzerland: Springer, 2018, pp. 1–17.

[39] S. Picek, A. Heuser, A. Jovic, K. Knezevic, and T. Richmond, "Improving side-channel analysis through semi-supervised learning," in *Proc. Int. Conf. Smart Card Res. Adv. Appl.* Springer, 2018, pp. 35–50.

[40] Y. LeCun and F. J. Huang, "Loss functions for discriminative training of energy-based models," in *Proc. AIStats*, vol. 6, 2005, p. 34.

[41] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: https://arxiv.org/abs/1412.6980

[42] L. Batina, J. Hogenboom, and G. J. van Woudenberg, "Getting more from PCA: First results of using principal component analysis for extensive power analysis," in *Proc. RSA*, 2012, pp. 383–397.

[43] M. O. Choudary and M. G. Kuhn, "Efficient stochastic methods: Profiled attacks beyond 8 bits," in *Proc. Int. Conf. Smart Card Res. Adv. Appl.* Cham, Switzerland: Springer, 2014, pp. 85–103.

[44] E. Cagli, C. Dumas, and E. Prouff, "Enhancing dimensionality reduction methods for side-channel attacks," in *Proc. Int. Conf. Smart Card Res. Adv. Appl.* Springer, 2015, pp. 15–33.

[45] J. G. van Woudenberg, M. F. Witteman, and B. Bakker, "Improving differential power analysis by elastic alignment," in *Proc. Cryptographers' Track RSA Conf.* Springer, 2011, pp. 104–119.

[46] K. Baddam and M. Zwolinski, "Evaluation of dynamic voltage and frequency scaling as a differential power analysis countermeasure," in *Proc. 20th Int. Conf. VLSI Design Held Jointly 6th Int. Conf. Embedded Syst.*, Jan. 2007, pp. 854–862.

[47] J. A. Ambrose, R. G. Ragel, and S. Parameswaran, "RIJID: Random code injection to mask power analysis based side channel attacks," in *Proc. DAC*, 2007, pp. 489–492.

[48] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-26, no. 1, pp. 43–49, Feb. 1978.

[49] F. Chollet *et al.*, "Keras: The python deep learning library," Astrophys. Source Code Library, Tech. Rep., 2018.

[50] M. Abadi *et al.*, "TensorFlow: A system for large-scale machine learning," in *Proc. OSDI*, vol. 16, 2016, pp. 265–283.

[51] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: https://arxiv.org/abs/1409.1556

Anupam Golder (S'19) received the B.Sc. degree in electrical and electronic engineering from the Bangladesh University of Engineering and Technology (BUET), Dhaka, Bangladesh, in 2015. He is currently working toward the Ph.D. degree at the Georgia Institute of Technology, Atlanta, GA, USA.

He is currently a Graduate Research Assistant at the Integrated Circuits and Systems Research Laboratory (ICSRL), School of Electrical and Computer Engineering, Georgia Institute of Technology. His current research interests include analog and digital VLSI circuit and system design, machine learning, and hardware security.

Debayan Das (S'17) received the B.E. degree in electronics and telecommunication engineering from Jadavpur University, Kolkata, India, in 2015. He is currently working toward the Ph.D. degree at the SPARC Lab, Purdue University, West Lafayette, IN, USA.

From 2015 to 2016, he was an Analog Design Engineer with xSi Semiconductors (startup), Bengaluru, India. His current research interests include hardware security and mixed-signal IC design.

Mr. Das was a recipient of the IEEE HOST Best Student Paper Award in 2017 and 2019 and the 3rd Best Poster Award in IEEE HOST 2018.

Josef Danial (S'14) received the B.Sc. degree in computer engineering from Purdue University, West Lafayette, IN, USA, in 2018, where he is currently working toward the master's degree at the SPARC Lab.

He has two years of industry experience, in automotive (Fiat Chrysler Automobiles, Auburn Hills, MI, USA) and IOT (Cisco Jasper, Santa Clara, CA, USA) companies. He is currently a Graduate Research Assistant with the SPARC Lab, Purdue University. His current research interests include machine learning, hardware security, and computer vision.

Santosh Ghosh received the Ph.D. degree from the Department of Computer Science and Engineering, IIT Kharagpur, Kharagpur, India, in 2011.

He was a Post-Doctoral Researcher at COSIC, KU Leuven, Leuven, Belgium. He is currently at Intel Labs, Intel Corporation, Hillsboro, OR, USA. He has authored or coauthored more than 35 research publications and 35 filed patents in USA. His current research interests include cryptography, hardware security, security for IoT, and autonomous driving.

Shreyas Sen (S'06–M'11–SM'17) received the Ph.D. degree in electrical and computer engineering from the Georgia Institute of Technology (Georgia Tech), Atlanta, GA, USA, in 2011.

He has more than 5 years of industry research experience at Intel Labs, Hillsboro, OR, USA, Qualcomm, Austin, TX, USA, and Rambus, Los Altos, CA, USA. He is currently an Assistant Professor at the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA. He has authored or coauthored 2 book chapters, more than 120 conferences and journal papers. He holds 13 patents granted/pending. His current research interests include mixed-signal circuits/systems for Internet of Things (IoT), biomedical, and security.

Dr. Sen serves/has served as an ETS and Technical Program Committee Member for DAC, CICC, DATE, ISLPED, ICCAD, ITC, VLSI Design, IMSTW, and VDAT and an Executive Committee Member for the IEEE Central Indiana Section. He was chosen by MIT Technology Review as one of the top 10 Indian Inventors Worldwide under 35 (MIT TR35 India Award), in 2018, for the invention of using the Human Body as a Wire, which has the potential to transform healthcare, neuroscience, and human–computer interaction. He was a recipient of the AFOSR Young Investigator Award 2017, the NSF CISE Research Initiation Initiative (CRII) Award 2017, the Google Faculty Research Award 2017, the HKN Outstanding Professor Award, the Intel Labs Divisional Recognition Award 2014 for industry-wide impact on USB-C type, the Intel Ph.D. Fellowship 2010, the IEEE Microwave Fellowship 2008, the GSRC Margarida Jacome Best Research Award 2007, the Best Paper Awards at CICC 2019, HOST 2017, 2018, and 2019, the ICCAD Best-in-Track Award 2014, the VTS Honorable Mention Award 2014, the RWS Best Paper Award 2008, the Intel Labs Quality Award 2012, the SRC Inventor Recognition Award 2008, and the Young Engineering Fellowship 2005. He serves/has served as an Associate Editor for the *IEEE Design & Test*.

Arijit Raychowdhury (SM'13) received the B.E. degree in electrical and telecommunication engineering from Jadavpur University, Kolkata, India, in 2001, and the Ph.D. degree in electrical and computer engineering from Purdue University, West Lafayette, IN, USA, in 2007.

From 2013 to July 2019, he was an Associate Professor and held the ON Semiconductor Junior Professorship with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA. His industry experience includes 5 years as a Staff Scientist at the Circuits Research Lab, Intel Corporation, Hillsboro, OR, USA, and 1 year as an Analog Circuit Researcher with Texas Instruments Inc. In January 2013, he joined the School of Electrical and Computer Engineering, Georgia Institute of Technology, where he is currently a Professor. He is currently the Co-Director of the Georgia Tech Quantum Alliance, Atlanta, GA, USA. He has authored or coauthored more than 170 articles in journals and refereed conferences. He holds more than 25 U.S. and international patents. His significant contributions to the semiconductor industry include the design of the world's first adaptive echo-cancellation network for integrated DSLs (TI) and embedded world-line boosting for SRAM arrays (Intel). His current research interests include low-power digital and mixed-signal circuit design, design of power converters, sensors and exploring interactions of circuits with device technologies.

Dr. Raychowdhury has served on the Technical Program Committees for VLSI Symposium, CICC, DAC, ICCAD, ISLPED, and DATE. He has also been a Guest Editor for multiple IEEE and ACM journals. He has also taught many short courses and invited tutorials at multiple conferences, workshops, industries, and universities. He was a recipient of the IEEE/ACM Innovator under 40 Award, the NSF CISE Research Initiation Initiative Award (CRII) in 2015, the Intel Labs Technical Contribution Award in 2011, the Dimitris N. Chorafas Award for outstanding doctoral research in 2007, the Best Thesis Award, College of Engineering, Purdue University, in 2007, the SRC Technical Excellence Award in 2005, the Intel Foundation Fellowship in 2006, the NASA INAC Fellowship in 2004, the Meissner Fellowship 2002. He and his students have received 11 best paper awards over the years. He was the Associate Editor of the IEEE TRANSACTIONS ON COMPUTER AIDED DESIGN from 2013 to 2018 and the Editor of the *Microelectronics Journal* (Elsevier Press) from 2013 to 2017.