The ISC Int'l Journal of Information Security

November 2023, Volume 15, Number 3 (pp. 149–158)

http://www.isecure-journal.org

Presented at the ISCISC'2023 in Tehran, Iran.

Cross-Device Deep Learning Side-Channel Attacks using Filter and Autoencoder **

Maryam Tabaeifard [1,*] and Ali Jahanian [1]

[1] *Shahid Beheshti University Faculty of Computer Science and Engineering Tehran, Iran*

ARTICLE INFO.

Keywords:
Cross-Device, Deep Learning, Hardware Security, Side-Channel Attack

Type:
Research Article

doi:
10.22042/isecure.2023.187517

ABSTRACT

Side-channel Analysis (SCA) attacks are effective methods for extracting encryption keys, and with deep learning (DL) techniques, much stronger attacks have been carried out on victim devices. However, carrying out this kind of attack is much more challenging in cross-device attacks when the profiling device and target device are similar but not the same, which can cause the attack to fail. We also reached this conclusion when using only DL-SCA attack on our cross-devise (Atmega microcontroller devices). Due to different processes that lead to significant device-to-device variations, the accuracy of the attack was, on average, only 23%. In this paper, we proposed a method for a real attack on cross-devices using pre-processing methods based on a combination of DL-based Autoencoder and Gaussian low-pass filter (GLPF). According to our analysis results, the accuracy of the attack using only deep learning-based Autoencoder increased to 70% on average, and it improved up to 82% by adding the GLPF technique. The results also showed that combining DL-based autoencoder and GLPF can lead to a successful attack with a maximum of 300 power traces from the victim device.

1 Introduction

Side Channel Analysis (SCA) attacks are one of the most common types of physical attacks on electronic cryptography devices. During the execution of cryptographic algorithms on these devices, much physical information like power consumption [1], execution time [2], and electromagnetic radiation leaks that adversaries can utilize this information to recover the secret keys of the system [3].

Template Attacks (TA) are one of the strongest types of SCAs, which are difficult to deal with [4], [5], especially when deep learning (DL) techniques are used to reach a more powerful TA [6]. Although the steps and methods of performing TA are similar to using DL technique, it is categorized as a separate attack called DL-based SCA because of the great power of DL in key extraction.

Generally, DL-based SCA attacks are carried out in two phases: training and attacking. In the training phase, the adversary stores a complete set of mea-

* Corresponding author.

**The ISCISC'2023 program committee effort is highly acknowledged for reviewing this paper.

Email addresses: m.tabaeifard@mail.sbu.ac.ir, jahanian@sbu.ac.ir

ISSN: 2008-2045

sured inputs, outputs, and leaked data of the profiled device. Then, this information is processed using DL techniques to create models and node weights. The first phase is very time-consuming and requires a high volume of data processing [7], while in the attack phase, the adversary uses a limited number of power traces during the execution of the encryption algorithm on the victim's device to retrieve the secret of the design. Then, the attacker can guess the secret key by matching the victim's power traces with the models that have been previously created.

As mentioned, DL-based SCA is one of the strongest attacks when the trained correctly and the victim device are the same. However, some concerns arise when using cross-device attacks in which the adversary has to use a similar device to the victim (i.e., not the same device) as the trained device. This is because he/she may have limited access or insufficient time to use the victim's device for training. Due to differences in the processing of these devices, transferring the models learned by the deep neural network (DNN) from the trained device may result in less accuracy when attacking the victim device, and the attack may fail [8]. The reason for the differences in processing may be related to the manufacturer, the manufacturing process, or any other factors. Recently, considerable contributions have been published attempting to mitigate or reduce the impact of these differences on attack efficiency using various methods.

The analysis of electromagnetic signal traces reveals their high sensitivity to the probe's position [9]. Therefore, any changes in the probe position during the data capture process in the profiling and attack phases will result in different traces. Similarly, differences in device manufacturers, the presence of environmental noise, various countermeasures [9] in the attack device, or any minor changes between the profiling and attack devices will result in capturing sets of tracing data for training and attack. In cryptography, a cross-device attack is a type of attack where a machine learning model is trained on data collected from one device, and then the same model is used to attack a different device. This attack aims to use the knowledge learned from the original device to infer sensitive information about the target device.

Researchers have recently tried to improve cross-device attacks by introducing different DNNs and applying different pre-processing methods. Thapar *et al.* [10] proposed a new method (TranSCA) that used a transfer-learning strategy to reduce training data and fewer iterations for fine-tuning an already trained model with a new dataset. The authors of [3] introduced MTL-SCA attack to improve the attack and make the layers changeable by combining transfer learning and meta-learning. They successfully performed the attack for both power consumption and electromagnetic leakage models in cross-device/domain modes for protected and unprotected AES on different processors. According to [11], devices can be classified into four categories based on their relationship. In this classification, homogeneous and heterogeneous devices are the most challenging to attack, respectively, due to process variations. They introduced FL-PA methodology to solve the process variation problem that was performed in the time domain and overcame the heterogeneous devices.

Pei Cao *et al.* [12] introduced an attack strategy using transfer learning for profiled side-channel analysis (CD-PA) that fine-tunes a pre-trained model with MMD loss, improving attack performance, even across devices and countermeasures to address cross-device attack problems. The authors in [13] used adversarial learning (AL-PA) to achieve device-invariant features without needing target-specific pr-eprocessing. Danial *et al.* [14] presented a cross-device deep-learning-based method (EM-SCA) in a low SNR scenario. They analyzed and compared the effectiveness of PCA, LDA, FFT, and spectrogram pre-processing techniques and proposed an algorithm for the optimal selection of training devices. Their results showed that LDA is the most efficient approach to achieve maximum accuracy.Picek *et al.* [9] also, investigate how using the DL-based denoising techniques can improve profiled SCA on targets protected with countermeasures. They analyze six types of noise and countermeasures, showing that denoising autoencoder significantly improves attack performance, but complex countermeasures can still pose challenges. Yu *et al.* [15] also proposed a novel denoising approach utilizing the U-Net model for SCA traces, harnessing the benefits of U-Net's deep architecture and inductive transfer learning to address cross-device issues.

Another approach to improve the cross-device attack is using multi-device training. Golder *et al.* [16] have carried out a practical attack with multi-device training. They improved the accuracy of the attack and reduced the impact of misaligned traces using the PCA method. Furthermore, Das *et al.* [17] presented a successful cross-device SCA using a deep learning method, X-DeepSCA. They achieved an average of 99.9% accuracy for all the test devices under attack with 200K traces for training.
We presented a new method as cross-device SCA based on a deep learning approach with pre-processing by autoencoder and Gaussian low-pass filter to improve attack success in cross-device scenarios by reducing the effects of process variations between two similar devices. We performed the attack by adding

an effective pre-processing method to achieve this goal. The steps of the performed attack in this paper are as follows:

- Training a deep neural network with profiling device traces and storing the network specifications.
- Pre-processing of attack device traces using a low-pass filter and then a convolutional autoencoder.
- Attacking the victim device by transferring the trained network.

The rest of the paper is organized as follows: Section 2 describes the background of profiled SCAs and DNNs. Section 3 introduces our new pre-processing on traces to apply to DNNs-based SCAs. Section 4 describes the experiments to examine the performance of the proposed model. Finally, Section 5 concludes the paper with the suggested future research scopes.

2 Background

This section introduces power-based SCAs, general information about deep neural networks and pre-processing methods, gaussian low pass filter, and convolutional autoencoder architecture used in the proposed method.

2.1 Profiled Side-Channel Analysis

Profiled side-channel attacks are considered a powerful type of SCAs. In these attacks, it is assumed that an adversary has access to an open clone device whose keys can either be chosen by the adversary or are already known. Template attack (TA) is one commonly used type of profiled side-channel attack. Unlike non-profiled attacks, TA can use the system's noise with an accurate model using the Gaussian noise model to analyze the system. The attack proceeds in two phases; the first phase involves creating a profile (profiling phase), which is used to calculate the parameters of the multivariate normal distribution using a device similar to the victim's device. The second phase is the attack phase, which uses the parameters obtained in the previous step to encrypt the target device's secret keys. This type of attack assumes that the profile device and the attack device are equal. The phases of this attack in detail are as follows [18]:

- **Profiling Phase:** In this phase, the attacker collects many power traces from a device. It is assumed that he has full control over the device and creates templates based on the power traces. Usually, the S-Box output of the first round of the AES algorithm is selected as the intermediate leakage value [4]. The S-box output byte is calculated according to Equation 1.

$$Subbyte = SBOX(P \oplus K) \tag{1}$$

where P is a byte of input data and K is a byte of the key. Each sample point in the captured power trace is composed of two parts: signal and noise. The mean (μ) and variance (σ) are two characteristics of the signal. If the signal has a Gaussian probability density function, these two characteristics describe the signal well [5].

If we use the model of 256 identifiers for the keys, then each trace is mapped to one of the 256 templates depending on the output of the S-box function. Each template has a mean vector (μ) and a covariance matrix (C). The variance and covariance components determine the distribution of noise for each template. The covariance matrix of the i^{th} template is calculated as Equation 3.

$$C_i = \frac{\sum_{j=1}^{n_i}(t_{i,j} - \mu_i)^{-1} \times (t_{i,j} - \mu_i)}{n_i - 1}, \forall i \in 0...8 \tag{2}$$

where μ_i is the mean vector of the traces, $t_{i.j}$ is the j^{th} power trace vector, and n_i is the number of traces in the i^{th} template.
Finally, the attacker will have several templates that can construct a multivariate Gaussian distribution function using the mean vector and covariance matrix of each template, as follows [19]:

$$f_{i,j}\Big(t_{i,j};(\mu_i.C_i)\Big) = \frac{exp\Big(t_{i,j} - \mu_i)^T.(C_i)^{-1}.(t_{i,j} - \mu_i)\Big)}{\sqrt{(2\pi)^n.|C_i|}} \tag{3}$$

- **Attacking Phase:** In this phase, the attacker focuses on recovering the secret key using the collected power traces from the target device. As mentioned earlier, the attack requires a small number of power traces at this phase to recover the secret key.The attacker calculates the Probability Density Function (PDG) for each guessed key and input and stores it in a matrix [20]. Finally, he has a vector with 256 elements by summing the columns of the matrix, where each element corresponds to a possible key. The key with the highest probability sum will be the best guess for the secret key.

In profiled attacks, point of interest (POI) selection is so important, which are the points that show the maximum correlation between the modeled power and the actual power consumption. Using POIs may reduce the number of power trace vectors, which leads to lower computations in the profiling phase. Sum Of Squared Differences (SOSD) and PCA are common methods for finding POIs [21].

2.2 Deep Neural Networks

An artificial neural network is a mathematical model that simulates the structure and functions of biological neural networks. Each neural network consists of at least three layers, including the input layer, hidden layer(s), and output layer. Each layer includes a set of nodes that vary based on the input and output dimensions and the nature of the problem.

The operation of a deep neural network (DNN) involves assigning weights to each layer that represents the features of that layer. During the training phase, the network adjusts these weights to find values that enable it to correctly map input data to its corresponding output [22]. Also, a loss function is measured to determine the distance between the output and the expected output. The feedback from the loss function is then used to update the weights to minimize the loss function.

Convolutional neural networks (CNNs) are a widely used type of deep neural network that is commonly used in SCA and have shown promising results in this field [23]. CNNs use convolutional layers that extract features from the input data using a small set of learned filters. This technique makes CNNs particularly well-suited for analyzing data with spatial or temporal relationships, such as images or time-series data.

2.3 Autoencoder-based Deep Neural Networks

Commonly, we face the challenge of reconciling the differences between the device profile and the attack when working with power consumption traces captured from a real device. An autoencoder based on DL is used to reduce this variation in device traces. This method is particularly useful when the exact shape and behavior of the unknown noise cannot be determined.

To train the autoencoder, the attacker's device traces should be provided as input and the victim's device traces as output to the network. Then, the network is trained to transform the input traces into the desired output traces. In other words, the network learns to detect and reduce the differences between the input and output traces to achieve its desired output, allowing the automatic encoder to reduce the data's dimensions and act as a noise reduction method.

Overall, the autoencoder based on DL is a powerful tool for reducing the variation between device profiles and attacks, particularly when dealing with unknown noise. The method's ability to learn and reduce differences between input and output traces makes it a valuable feature and noise reduction technique [9].

2.4 Low Pass Filter

A low-pass filter (LPF) operates based on the PDF of the signal samples. The LPF is a well-established and widely used method for noise reduction in images and signals. This method works by convolving the input signal with a function that reduces the effect of high-frequency components [9].

Gaussian filters are linear filters defined based on the Gaussian probability distribution function as:

$$f(x) = e^{\frac{-x^2}{2\sigma^2}} \tag{4}$$

where σ is the standard deviation. Gaussian Low Pass Filters (GLPF) work similarly to the neighborhood averaging method. Intuitively, if we consider an image as input to this filter, decreasing the standard deviation of the image makes it more apparent, while increasing it makes the image more blurred. Using a GLPF with a convolution mask can reduce the high-frequency components of the power consumption traces captured from the devices. So, GLPF is known as a powerful noise reduction technique in various applications, including images and signals.

As the presence of noise in the captured traces is one of the reasons for the variation between the traces of similar devices [9], using the GLPF for deducting the noise, in particular, is effective in reducing the variation between the devices and improving the accuracy of cross-device side-channel attacks.

2.5 Transfer Learning

In recent years, transfer learning has led to significant advances in machine learning. Transfer learning is generally used when labeled training data is insufficient for a given task. In this case, a pre-trained model can be utilized that has been trained on a large volume of data for a similar task. Transfer learning can also be used when some tasks share similar input data.

One of the advantages of transfer learning is that it allows us to use the pre-trained model in its entirety or only a few specific layers, depending on our needs and the task at hand. This flexibility can significantly reduce the time and resources required for training a model from scratch.

Moreover, transfer learning can also improve the performance of a model by leveraging the knowledge learned from the pre-trained model. The pre-trained model has already learned valuable features from a large volume of data, which can be fine-tuned for the specific task. Doing so can improve performance with

less labeled data and training time [10].

3 The Proposed Methodology

In this section, we describe our proposed approach for enhancing the DL-based SCA to tackle the problem of cross-device variations. Firstly, we demonstrate the source of dissimilarities between two similar devices and quantify these differences. For this purpose, the Pearson product-moment correlation coefficient (PPMCC) was proposed in [3] as a statistical measure that quantifies the degree of linear relationship between two variables. In the context of SCA, these variables are typically the power consumption traces of two similar devices. According to equation (5), the PPMCC is calculated separately for each device using n pairs of data (X_i, Y_i), where X_i and Y_i are the power consumption trace of profiled and attack devices, respectively, for the same operation.

We used the dataset in [17] that includes traces captured from 6 separate devices, D1 to D6, with an Atmega microprocessor. All the traces were obtained under the same conditions and during the execution of the AES 128-bit encryption algorithm using the *Chipwhisperer* platform. The result of the PPMCC test between our devices for the cross-device attack using the proposed method is shown in Figure 1.

$$PPMCC = \frac{\sum_{j=1}^{n_i}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{j=1}^{n_i}(X_i - \bar{X})}.\sqrt{\sum_{j=1}^{n_i}(Y_i - \bar{Y})}} \tag{5}$$

Figure 1 shows that the coefficient is higher for certain cross-device cases, specifically between devices (1 with 5), (2 with 4), and (3 with 6), indicating that these devices are more similar and are more vulnerable to attack. Conversely, the coefficient is lower for the other cross-device cases, making them more challenging to attack.

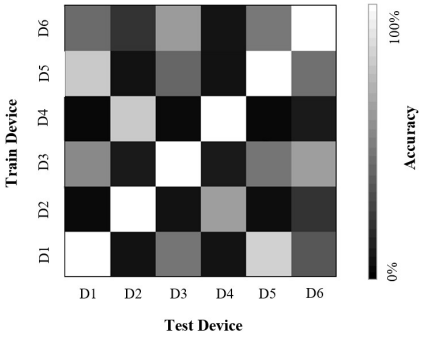

Figure 1. Similarity between devices based on the PPMCC criterion

To improve the portability of the attack, it is necessary to eliminate the differences between the devices. Since the differences between similar devices are entirely random, it can be considered as a Gaussian noise [9], which naturally exists in the devices and causes variations between them. Therefore, we propose a noise reduction technique to reduce these differences. As it is not feasible to eliminate the differences in practice, we aim to reduce them as much as possible using available techniques.

3.1 Deep Neural Network Architecture

We utilize the DNN architecture [21] to enhance cross-device attacks using DL technique. While the method proposed in [17] employed multi-device training in the profiling phase, we aim to use only single-device training to minimize cost and time. Our DNN network contains an input layer with 150 neurons, equal to the number of input points. The first layer is fully-connected (FC) with 100 neurons. A batch normalization layer is used to regulate and normalize the data, and a random dropout layer with a rate of 0.1 is used to prevent the network from overfitting the data. Moreover, the Rectified Linear Unit (ReLU) activation function is used in this method to learn non-linear mappings from input to output. The second layer is FC, with a dropout rate of 0.05, followed by another batch normalization layer. Finally, the output layer has 256 neurons and the function is used to predict the probability of the correctness of each 256 key bytes with a predicted power trace.

The loss function used was categorical cross-entropy with a learning rate 0.01 and optimized with the Adam algorithm. The experimental results show that increasing the random dropout rate for the first and second layers, each beyond 0.1 and 0.05, respectively, can result in underfitting. Conversely, selecting smaller dropout values can preserve the features learned by the network but may not generalize well for unseen samples during evaluation. However, we observe that these values can be changed at a very low rate of about 4%, which is attributed to the noise present in the power traces. This noise allows the network to be generalized to some extent. The network is trained for 50 epochs and 15% of the training data is used for evaluation. The batch size is selected between 16 and 512 with a step size of 32 to find the optimal batch size. On average, a batch size of 80 leads to the best accuracy during the training process.

3.2 Pre-processing with Autoencoder

In the proposed method, firstly, the DNN is trained using power traces captured from the profiled device (i.e., the trained device); after that, the convolutional autoencoder is trained by the power traces and the

average of every five power traces as the training sets.

Once the training process is finished, the trained model can pre-process the power traces obtained from the victim device. These pre-processed power traces are fed as the input to the DNN. The transfer learning technique is utilized to optimize the time and use a lower number of traces for the attack. In this approach, the weights and models of the DNN created during the training phase are transferred to the DNN of the attack phase. However, for fine-tuning the network, the weights of the last two layers are updated according to the traces of the victim device. This technique allows the network to fine-tune with the transferred weights, saving time to reach the optimal model.

Figure 2 shows the autoencoder network architecture used in this paper. It consists of 11 layers comprising 4 convolutional layers, each with 128 and 64 filters, respectively, to detect the differences between two devices and remove the input differences to obtain the output. Then, 2 FC layers with 32 filters are used to transfer the obtained features. Furthermore, 4 deconvolution layers, each with 64 and 128 filters, are used to reconstruct the traces using the transferred features and create a general representation. Finally, a convolutional layer with one output neuron is added as an output layer. Moreover, Selu activation function is used for all 10 autoencoder layers and Sigmoid activation function for the output layer. The RMSprop optimizer and binary cross-entropy loss function are utilized for this network. This network's optimal number of epochs and batch size are estimated as 50 and 100, respectively. Among all the training power traces, 23% is used for evaluation and the remaining is used for training.

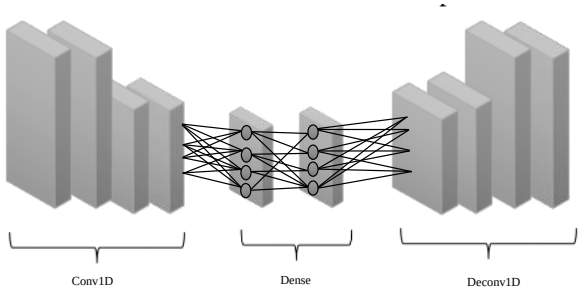

Figure 2. Convolutional autoencoder network architecture

We apply a low-pass Gaussian filter technique on raw power traces of the attack device before the autoencoder process to improve the accuracy in the attack phase. For this purpose, a one-dimensional filter is constructed using the "fspecial" function with three parameters (i.e., a Gaussian filter with filter size $= 5 \times 1$ and standard deviation $= 5.0$) shown in Figure 3. By passing the traces through the filter, each trace point is convolved with the constructed Gaussian filter to reduce the noise [24]. After all, the output traces are fed into the autoencoder network.

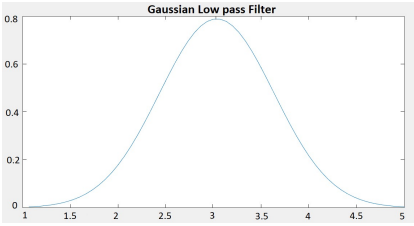

Figure 3. Gaussian low pass filter

4 Experimental Results

In this section, we present the experimental results for evaluating the efficiency of our proposed method. The traces used in this research are captured from 6 Atmega 8-bit microcontrollers (NAE 308T XMEGA) devices while executing the 128-bit AES encryption algorithm using the *Chipwhisperer* platform under the same conditions [17]. An essential feature of these traces is their synchronous timing.

The Deep Neural Network model is implemented using Python and the *Keras* library with *Tensorflow* as the backend [19]. As mentioned before, we use the leakage model in SCA to train the DNN. Therefore, the network output has 256 classes, and all experiments are performed with 10,000 traces from each device, approximately 40 traces for each class with the same inputs. The number of points that *Chipwhisperer* stores is 3000, and by finding the most important points, the complexity of the network is reduced. Therefore, the first 150 points are given as input to the network, in which the CPA attack [20] is successfully performed on this range.

We trained the DNN using traces from one of the six devices and selected the others as the attack devices. The number of traces is reduced from 1000 to 300 to decrease the minimum number of traces to disclosure (MTD). By selecting appropriate hyperparameters and achieving similar accuracy for testing and training, overfitting or underfitting is avoided. The average attack accuracy at this stage is, on average, 20% due to process variations, which leads to an unsuccessful attack. We used our defined autoencoder to reduce the variations between profile and attack devices and improve the attack accuracy. This technique increases the attack accuracy on average to 73%, which is a significant improvement. We applied a Gaussian low-pass filter to the attack traces before executing the autoencoder for further improvement.

This technique increases the accuracy to 82%, resulting in a successful attack. For example, Figure 4 shows the results of using Device 1 as a training device and Devices 2 to 6 as attack devices. As shown in Figure 4 to Figure 9 similar results can be derived for all permutations of the devices used as training and attack devices.

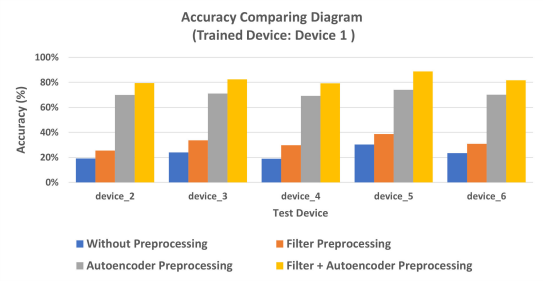

Figure 4. Accuracy comparing diagram with training on Device 1

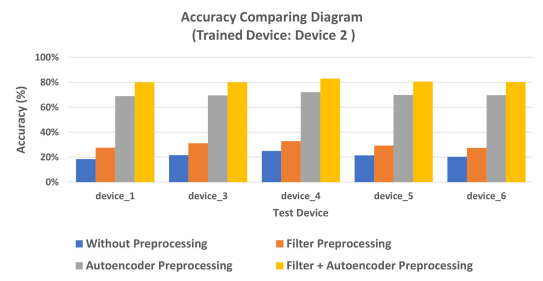

Figure 5. Accuracy comparing diagram with training on Device 2

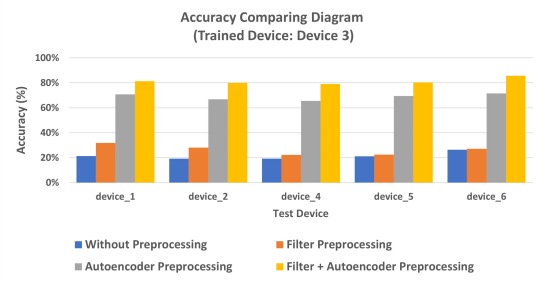

Figure 6. Accuracy comparing diagram with training on Device 3

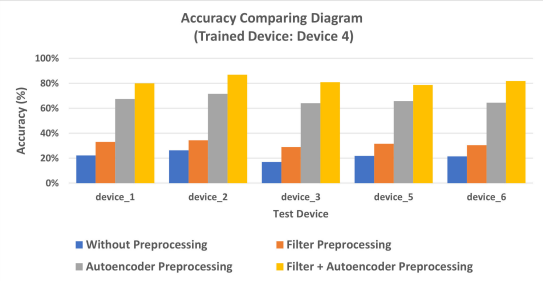

Figure 7. Accuracy comparing diagram with training on Device 4

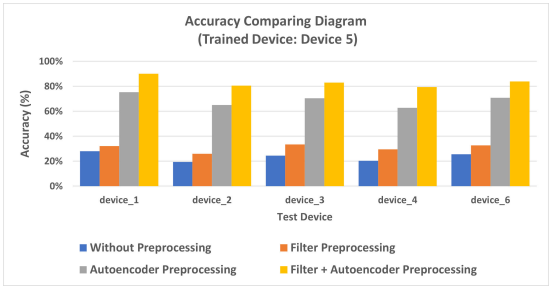

Figure 8. Accuracy comparing diagram with training on Device 5

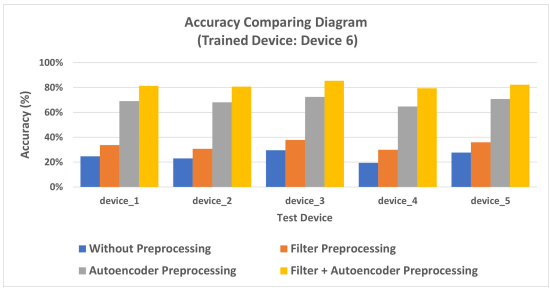

Figure 9. Accuracy comparing diagram with training on Device 6

The comparison to related works is presented in Table 1. All six studies are based on deep learning techniques differing in their methods for improvement. Consequently, the required power traces from the trained device differ for each technique. Using a combination of CAE and GLPF methods in our study significantly reduces noise and variation between the training and victim devices. Considering the time-consuming nature of the training phase, the results of our method demonstrate that a successful attack can be achieved with fewer traces. We reach successful attacks with the desired accuracy in all our tests by using only 10,000 traces from a single device for training and 300 traces from the victim device to fine-tune the network. This outcome highlights the positive impact of this method in cross-device attacks.

Table 1. A comparison to related works with their techniques and results

Method	Improving Technique	Train traces	Fine tune traces	Result
CD-PA [12]	Transfer Learning + Regularization	25000	100	GE = 0
AL-PA [13]	Adversarial Learning	25000	x	GE = 0
U-Net-SCA [15]	Supervised U-shaped Network	20000	×	GE = 0
MTL-SCA [3]	Transfer Learning + Meta Learning	20000	800	GE = 0
X-DeepSCA [17]	Multi device training	$4 * 10000$	×	Accuracy = 90%
This Work	Transfer Learning + GLPF + CAE	10000	300	Accuracy = 82%

4.1 Adding Cross-Device Variations

To demonstrate the effectiveness of the proposed method in the presence of more significant process variations between devices, we artificially added random Gaussian noise at intervals of (0 to 0.04), (0 to 0.08), and (0 to 0.12) voltage to the traces of the test devices respectively. We performed the attacks on the test devices with and without pre-processing methods.

Figures 10, 11, and 12 show the attack accuracy on Device 1 to 6 for the DNN trained on Device 1, 2, and 3, respectively, while increasing the variation between the devices. The graph trend indicates that with an increase in the variation between the devices from 0.01 to 0.12 volts, the accuracy decreases from about 80% to about 60%. Furthermore, we observe the impact of correlation (Figure 1) between the training and victim devices up to 0.05 volts of variation between the devices. As indicated in Figure 1, the correlation between Device 1 and 5 is higher than the others. Furthermore, in Figure 10, we observe that Device 5 exhibits more accuracy than the other devices. Similarly, in Figure 11, due to the elevated correlation between Device 2 and Device 4, it becomes evident that Device 4 possesses higher accuracy and in Figure 12, Device 6 demonstrates a more significant correlation with Device 3, leading to a higher accuracy outcome for Device 6. The overall results for the permutations of the six devices show that this correlation effect diminishes in the voltage range of 0.5 to 0.8 volts and beyond this range, the effect of the initial correlation between the devices disappears.

The results in Figure 13 show that by increasing variation from 0 to 0.12 voltage, the average accuracy without pre-processing methods decreased from 23% to 6%. It is obvious that by using only the autoencoder method, the accuracy will be decreased from 70% to 59%. However, after executing our proposed attack method, the accuracy changes from 82% to 70%.

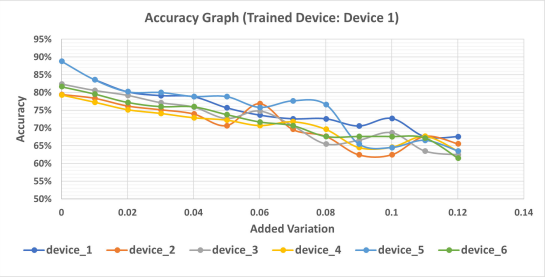

Figure 10. Accuracy trending for the proposed attack to Device 1 to 6, in considering different variations between devices with the trained DNN on Device 1

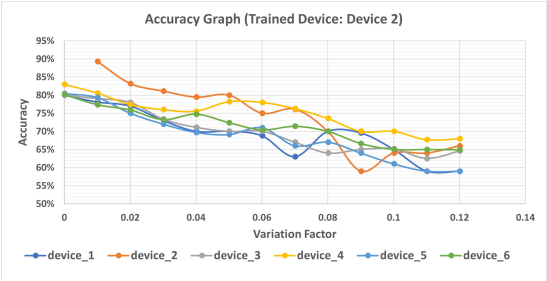

Figure 11. Accuracy trending for the proposed attack to Device 1 to 6, in considering different variations between devices with the trained DNN on Device 2

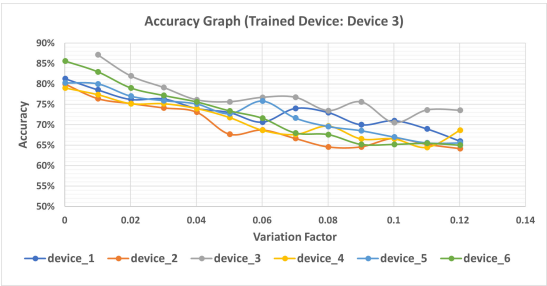

Figure 12. Accuracy trending for the proposed attack to Device 1 to 6, in considering different variations between devices with the trained DNN on Device 3

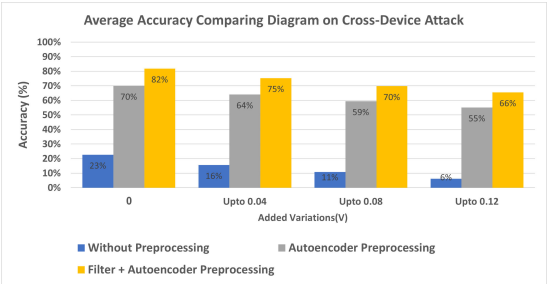

Figure 13. Average accuracy comparing diagram on cross-device attack

5 Conclusion

In this article, we presented a cross-device SCA based on deep learning algorithm with pre-processing by autoencoder and Gaussian low-pass filter. As the first step, raw attack traces are used and get an average of 23% accuracy; then using the proposed autoencoder, we achieved an average accuracy of 70% for the attack. Finally, we pre-processed the attack device's power traces with a combination of Gaussian low-pass filter and our autoencoder and achieved an average accuracy of 82% with a maximum of 300 power traces. This is a desirable result, and for every six sets of power traces provided to the network, the correct key is calculated accurately by eight tries.

Regarding future work, one of the goals is to achieve high accuracy with multi-device training of a network and increase the likelihood of a successful attack, which would also increase the success rate for

cross-device attacks with more significant variations between devices.

For the future work, we can do the proposed method for multi-device training to reach better accuracy and also apply pre-processing methods on the training device to get more accurate models in the training phase.

References

[1] Debayan Das, Shovan Maity, Saad Bin Nasir, Santosh Ghosh, Arijit Raychowdhury, and Shreyas Sen. High efficiency power side-channel attack immunity using noise injection in attenuated signature domain. In *2017 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 62–67. IEEE, 2017.

[2] Yangdi Lyu and Prabhat Mishra. A survey of side-channel attacks on caches and countermeasures. *Journal of Hardware and Systems Security*, 2:33–50, 2018.

[3] Honggang Yu, Haoqi Shan, Maximillian Panoff, and Yier Jin. Cross-device profiled side-channel attacks using meta-transfer learning. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pages 703–708. IEEE, 2021.

[4] Suresh Chari, Josyula R Rao, and Pankaj Rohatgi. Template attacks. In *Cryptographic Hardware and Embedded Systems-CHES 2002: 4th International Workshop Redwood Shores, CA, USA, August 13–15, 2002 Revised Papers 4*, pages 13–28. Springer, 2003.

[5] Marios O Choudary and Markus G Kuhn. Efficient, portable template attacks. *IEEE Transactions on Information Forensics and Security*, 13(2):490–501, 2017.

[6] Dongxin Guo, Kaiyan Chen, Xiaoyang Hu, Yanhai Wei, and Jianlong Li. A survey of prototype side-channel attacks based on machine learning algorithms for cryptographic chips. In *Journal of Physics: Conference Series*, volume 1176, page 032005. IOP Publishing, 2019.

[7] Soroor Ghandali, Samaneh Ghandali, and Sara Tehranipoor. Profiled power-analysis attacks by an efficient architectural extension of a cnn implementation. In *2021 22nd International Symposium on Quality Electronic Design (ISQED)*, pages 395–400. IEEE, 2021.

[8] Shivam Bhasin, Anupam Chattopadhyay, Annelie Heuser, Dirmanto Jap, Stjepan Picek, and Ritu Ranjan. Mind the portability: A warriors guide through realistic profiled side-channel analysis. In *NDSS 2020-Network and Distributed System Security Symposium*, pages 1–14, 2020.

[9] Lichao Wu and Stjepan Picek. Remove some noise: On pre-processing of side-channel measurements with autoencoders. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 389–415, 2020.

[10] Dhruv Thapar, Manaar Alam, and Debdeep Mukhopadhyay. Deep learning assisted cross-family profiled side-channel attacks using transfer learning. In *2021 22nd International Symposium on Quality Electronic Design (ISQED)*, pages 178–185. IEEE, 2021.

[11] Fan Zhang, Bin Shao, Guorui Xu, Bolin Yang, Ziqi Yang, Zhan Qin, and Kui Ren. From homogeneous to heterogeneous: Leveraging deep learning based power analysis across devices. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2020.

[12] Pei Cao, Chi Zhang, Xiangjun Lu, and Dawu Gu. Cross-device profiled side-channel attack with unsupervised domain adaptation. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 27–56, 2021.

[13] Pei Cao, Hongyi Zhang, Dawu Gu, Yan Lu, and Yidong Yuan. Al-pa: Cross-device profiled side-channel attack using adversarial learning. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*, pages 691–696, 2022.

[14] Josef Danial, Debayan Das, Anupam Golder, Santosh Ghosh, Arijit Raychowdhury, and Shreyas Sen. Em-x-dl: Efficient cross-device deep learning side-channel attack with noisy em signatures. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 18(1):1–17, 2021.

[15] Honggang Yu, Mei Wang, Xiyu Song, Haoqi Shan, Hongbing Qiu, Junyi Wang, and Kaichen Yang. Noise2clean: Cross-device side-channel traces denoising with unsupervised deep learning. *Electronics*, 12(4):1054, 2023.

[16] Anupam Golder, Debayan Das, Josef Danial, Santosh Ghosh, Shreyas Sen, and Arijit Raychowdhury. Practical approaches toward deep-learning-based cross-device power side-channel attack. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 27(12):2720–2733, 2019.

[17] Debayan Das, Anupam Golder, Josef Danial, Santosh Ghosh, Arijit Raychowdhury, and Shreyas Sen. X-deepsca: Cross-device deep learning side channel attack. In *Proceedings of the 56th Annual Design Automation Conference 2019*, pages 1–6, 2019.

[18] Farshideh Kordi, Hamed Hosseintalaee, and Ali Jahanian. A time randomization-based countermeasure against the template side-channel attack. *ISeCure*, 14(1), 2022.

[19] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean,

Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. {TensorFlow}: a system for {Large-Scale} machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pages 265–283, 2016.

[20] Owen Lo, William J Buchanan, and Douglas Carson. Power analysis attacks on the aes-128 s-box using differential power analysis (dpa) and correlation power analysis (cpa). *Journal of Cyber Security Technology*, 1(2):88–107, 2017.

[21] Farshideh Kordi, Hamed Hosseintalaee, and Ali Jahanian. Cost-effective and practical countermeasure against the template side channel attack. In *2020 17th International ISC Conference on Information Security and Cryptology (ISCISC)*, pages 22–27. IEEE, 2020.

[22] Stjepan Picek, Annelie Heuser, Alan Jovic, Simone A Ludwig, Sylvain Guilley, Domagoj Jakobovic, and Nele Mentens. Side-channel analysis and machine learning: A practical perspective. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 4095–4102. IEEE, 2017.

[23] Eleonora Cagli, Cécile Dumas, and Emmanuel Prouff. Convolutional neural networks with data augmentation against jitter-based countermeasures. In *Cryptographic Hardware and Embedded Systems-CHES 2017-19th International Conference*, 2017.

[24] MathWorks. https://uk.mathworks.com/help/\images/ref/fspecial.html. MathWorksR2023a.

Maryam Tabaeifard received her B.S. degree in Computer Hardrware Engineering from Shahid Beheshti University, Tehran, Iran in 2018 and the M.S. degrees in Computer Engineering from Shahid Beheshti University, Tehran, Iran in 2023. Her current research interests are Hardware Security and Deep Learning Techniques in Security.

Ali Jahanian received his B.S. degree in Computer Engineering from University of Tehran, Tehran, Iran 1996 and the M.S. and Ph.D. degrees in Computer Engineering from Amirkabir University of Technology, Tehran, Iran in 1998 and 2008, respectively. He joined Shahid Beheshti University, Tehran, Iran in 2008. His research interests are Hardware Security and Biochips Design.