Open Access Library Journal
2022, Volume 9, e8307
ISSN Online: 2333-9721
ISSN Print: 2333-9705

Software Implementation of AES-128: Cross-Subkey Side Channel Attack

Fanliang Hu, Junnian Wang, Wan Wang, Feng Ni

School of Physics and Electronics, Hunan University of Science and Technology, Xiangtan, China
Email: fanliang@mail.hnust.edu.cn

How to cite this paper: Hu, F.L., Wang, J.N., Wang, W. and Ni, F. (2022) Software Implementation of AES-128: Cross-Subkey Side Channel Attack. *Open Access Library Journal*, **9**: e8307.
https://doi.org/10.4236/oalib.1108307

Received: December 19, 2021
Accepted: January 24, 2022
Published: January 27, 2022

Abstract

The majority of recently demonstrated Deep-Learning Side-Channel Attacks (DLSCAs) use neural networks trained on a segment of traces containing operations only related to the target subkey. However, when the number of training traces is restricted such as in the ASCAD database, deep-learning models always suffer from overfitting since the insufficient training data. One data-level solution is called data augmentation, which is to use the additional synthetically modified traces to act as a regularizer to provide a better generalization capacity for deep-learning models. In this paper, we propose a cross-subkey training approach which acts as a trace augmentation. We train deep-learning models not only on a segment of traces containing the SBox operation of the target subkey of AES-128, but also on segments for other 15 subkeys. We show that training a network model by combining different subkeys outperforms a traditional network model trained with a single subkey, and prove the conclusion on two well-known datasets.

Subject Areas

Information and Communication Security, Privacy, and Trust

Keywords

Side-Channel Attack, Deep Learning, AES, Cross-Subkey Training

1. Introduction

Side-Channel Attacks (SCAs) have become a realistic threat to implementations of cryptographic algorithms, such as Advanced Encryption Standard (AES) [1]. Even theoretically secure cryptography may be broken since the encryption has to run in hardware or software at some point to actually do things. There might be some unintentional physical leakage during the execution of a cryptographic algorithm, such as the power consumed [2] [3] by the victim device. By utilizing the unintentional physical leakage, it is possible for SCAs to bypass the theoretical strength of cryptographic algorithms and to recover the key. This is particularly threatening since once the secret key is leaked, the ciphertext can be decrypted and the signature can be forged.

Recently, with advances in deep learning [4], SCAs are able to be more effective than the conventional cryptanalysis and are more practical to mount. Since well-trained deep-learning models are good at extracting features from the raw data, which helps the attacker to find the correlation between the physical measurements and the internal state of the processing device. Many deep-learning based side-channel attacks against both software [5] [6] [7] and hardware implementations [8] [9] [10] [11] of AES have been presented. [12] first investigates hyper parameters of deep-learning models in SCAs and builds the ASCAD benchmark database. In [13], a multi-label approach is proposed and it surpasses the state-of-the-art result in ASCAD database. In [5], the effect caused by the board diversity has been demonstrated and it shows that it is easy to overestimate the attack efficiency if deep-learning models are trained on traces captured from the victim device. Afterwards, several different aggregation methods are proposed to mitigate this accuracy gap caused by the board diversity. The data-level aggregation attack (also called cross-device attack) [7] [6] [14] trains deep-learning models on traces captured from multiple devices. The model-level aggregation [15] utilizes the newly introduced federated learning framework to build the global model by averaging multiple local models' weights.

Most of these existing deep-learning based attacks use a divide-and-conquer strategy to recover a 128-bit secret key of AES-128, in which the 128-bit key K is divided into 8-bit parts $k_i \in \mathcal{K} = \{0,1,\cdots,255\}$, called *subkeys*, for $i \in \{1,2,\cdots,16\}$. We use $\mathcal{K}$ to denote the set of all possible subkey candidates. Afterwards, each subkey k_i is recovered independently by using the deep-learning models trained on traces only related to a specific subkey k_i. In our experiments, we focus on the 1_{st} subkey, and others will be the same.

However, when the number of training traces is not sufficient, deep-learning models always suffer from overfitting. A common solution for this is data augmentation, which is to use modified version of existing data to expand the training set. In SCAs, a trace segment leaked by an operation related to the i_{th} subkey k_i could be used as an augmenting trace for another subkey k_j, with the same operation and the same input. In some implementations of AES-128, instructions are computed sequentially and procedures are executed byte-by-byte. This means if two identical operations have the same input data, for example, two SBox substitutions in the first round of AES, the resulting power consumption or electromagnetic emission could be similar. Probably this was noticed before but the potential benefit of training models on traces for multiple subkeys has not been fully explored.

In this paper, we propose a cross-subkey training approach, which utilizes multiple subkeys instead of one to build deep-learning models with a better generalization capacity. By adding a certain amount of traces which are related to the non-targets subkeys, the profiling data set can be considered as a data augmentation for the traces of the target subkey. We conducted experiments on two well-known publicly available datasets (AES_GPU and ASCAD) in order to explore the impact of the different occupancy profiles of non-target and target subkey traces in the training set on the classification accuracy of the network.

2. Background

This section first reviews AES-128. Afterwards, we briefly introduce deep learning and how to apply deep learning to side-channel attacks. For a broader introduction for deep learning, see [4].

2.1. AES-128

AES [1] is one of the most widely used symmetric cryptographic algorithms standardized by NIST in FIPS 197 and included in ISO/IEC 18033-3. AES-128 is a subset of AES which takes a 128-bit key K to encrypt a 128-bit block of plaintext P, and the output is a 128-bit block of ciphertext C. AES-128 contains 10 encryption rounds in total and except the last round, each round repeats 4 steps sequentially: *SubBytes*, *ShiftRows*, *MixColumns* and *AddRoundKey*. The final round does not contain *MixColumns*. In our experiment, the mode of operation is set to Electronic Codebook (ECB) mode, which first divides the message into blocks and each block is encrypted separately. The *SubBytes* procedure is a non-linear substitution which maps an 8-bit input to an 8-bit output by using the Substitution Box (SBox).

An attack point for side-channel attacks is a selected intermediate state which can be used to describe the power consumed by the victim device during the execution of AES. The selection of attack point is affected by known input data (e.g. plaintext, ciphertext) and physical measurements (e.g. power consumption, EM emissions, timing). Two common attack points are SBox output in the first and the last round of AES. An appropriate attack point will lead to a more efficient attack.

2.2. Deep-Learning Side-Channel Attack

Deep learning is a subset of machine learning [16] that uses deep neural networks to learn from experience and understand the input data in terms of a hierarchy of concepts. Since deep-learning techniques are good at extracting features in raw data [4] [17] [18], deep-learning based SCAs become several orders of magnitude more effective than the traditional cryptanalysis. A typical deep-learning side-channel attack can be divided into two stages:

At the profiling stage, the attacker aims to use the deep-learning model to learn a leakage profile by using a large set of power traces $\mathcal{T} = \{T_1, T_2, \cdots, T_m\}$ captured from the profiling device, where m is the number of traces in the training set. Each trace T_i is labeled by the data processed at the attack point

$l(T_i) \in L$, where $L = \{0,1,\cdots,255\}$, which can be used to derive the subkey by using some known input (e.g. the plaintext, ciphertext). The process of building a neural network can be viewed as a mapping $\mathcal{N} : \mathbb{R}^m \to \mathbb{I}^{|L|}$ and the output is a *score* vector $S = \mathcal{N}(T) \in \mathbb{I}^{|L|}$. The element s_j with value *j* in *S* represents the probability that $l(T) = j$.

At the attack stage, the attacker uses the trained deep-learning model to classify traces captured from the victim device and obtain the score vector. The attacker can find the i_{th} subkey $k_i = j$ which has the largest probability in *S*. We use k_i^* to denote the real subkey. Once $k_i = k_i^*$, the subkey is recovered successfully. To quantify the classification error of the neural network, we use the cross-entropy [16] as the loss function and the optimizer is set to RMSprop (Root Mean Square prop).

$$k_i = \underset{0 \le j \le 255}{\arg\max}\, \tilde{s}_j .$$

2.3. Composition of Power Traces

Power based side-channel attacks utilize the fact that the power consumed during the execution of the encryption process by the victim device might be different according to the different input data and different operations. Therefore, the most interesting parts of a power consumption trace can be defined as a data-dependent component $\mathcal{P}_{data}$ and an operation-dependent component $\mathcal{P}_{op}$. Besides, using the same device to repeat the same operation with the same input data will also consume different amount of power for every repetition because of the electronic noise component $\mathcal{P}_{noise}$. Meanwhile, the switching activities of the transistors which are independent from the input data can generate a constant amount of power consumption, which is called the constant component $\mathcal{P}_{const}$. Thus, each point of a power trace can be modeled as the sum of these components [3].

$$\mathcal{P}_{total} = \mathcal{P}_{data} + \mathcal{P}_{op} + \mathcal{P}_{noise} + \mathcal{P}_{const}$$

3. Cross-Subkey Attack

Figure 1 shows an overview of how a cross-subkey model is trained, in which different subkeys are used collaboratively to provide a better generalization capacity for the target subkey.

3.1. Trace Augmentation

Deep-learning techniques have performed remarkably well on many side-channel attack scenarios. However, deep-learning models always suffer from overfitting with insufficient training measurements. Overftting refers to the phenomenon when a network learns a function with very high variance such as to perfectly model the training data [19]. Unfortunately, many attackers may not have access to big profiling data, for instance, attackers may not have a full control to the profiling device and can only capture a limited amount of traces. One data-level

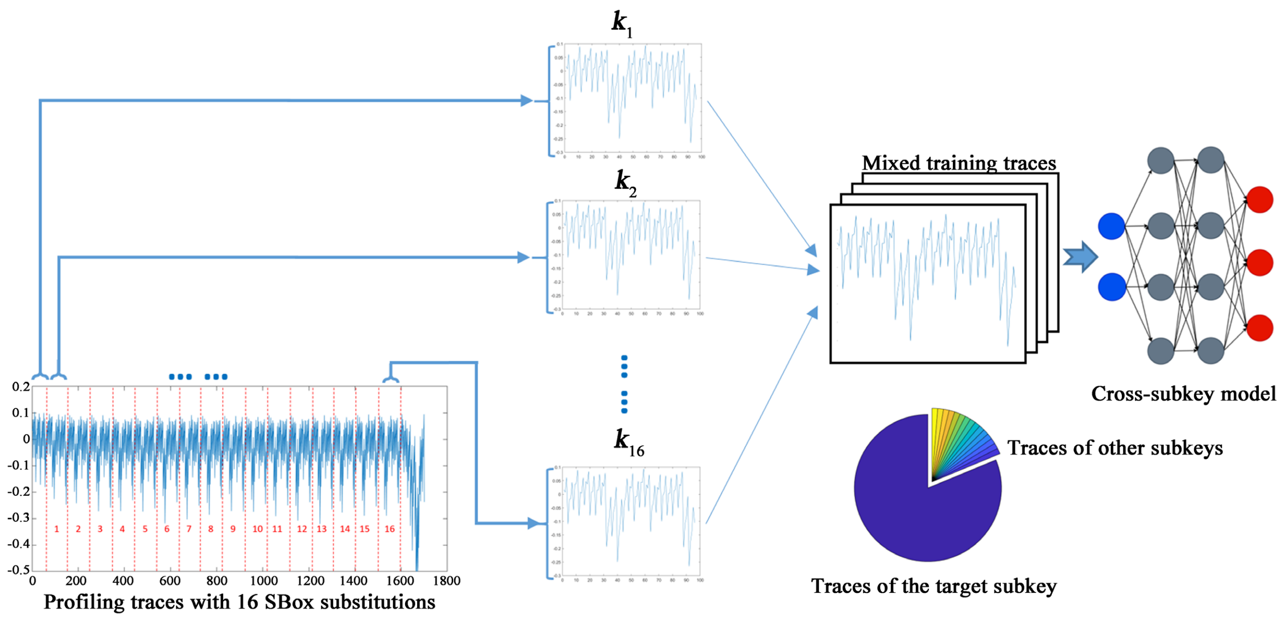

Figure 1. An overview of how the cross-subkey model is trained on the mixed profiling set.

solution to the problem of limited training data is data augmentation, which aims to use the additional synthetically modified traces to act as a regularizer and helps reduce overfitting when training a model in side-channel attacks' context.

In software implementations of AES, leakage is time-dependent since instructions are carried out one by one [9]. This leads to a generally accepted approach for the attack to against software implementation of AES, which is to build a leakage profile between traces and the target subkey. Typically for the 8-bit microcontrollers and microprocessors, the encryption is implemented byte by byte. If the same data is processed by two SBox substitutions, power traces of these two operations could be similar since the the data-dependent components and operation-dependent components in formula 2.3 are the same. Figure 2 shows power traces captured from an 8-bit microcontroller implementation of AES, which represent the first SBox and the second SBox operations in the first round. One can see that power traces look very similar if the same data is processed by two SBox substitutions. So we could use a small amount of traces related to the non-target subkeys as a regularizer for the training set which contains traces only for the target subkey. It is a data augmentation for a specific subkey to build the model with a better generalization capacity.

3.2. Cross-Subkey Model Training

As shown in Figure 1, a trace which contains 16 SBox computations of the first round is first divided into 16 sub-traces. The i_{th} sub-trace is labeled by l_i which represents the output of the i_{th} SBox procedure, with p_i denotes the i_{th} byte of the plaintext.

$$l_i = \text{SBox}\left(p_i \oplus k_i\right)$$

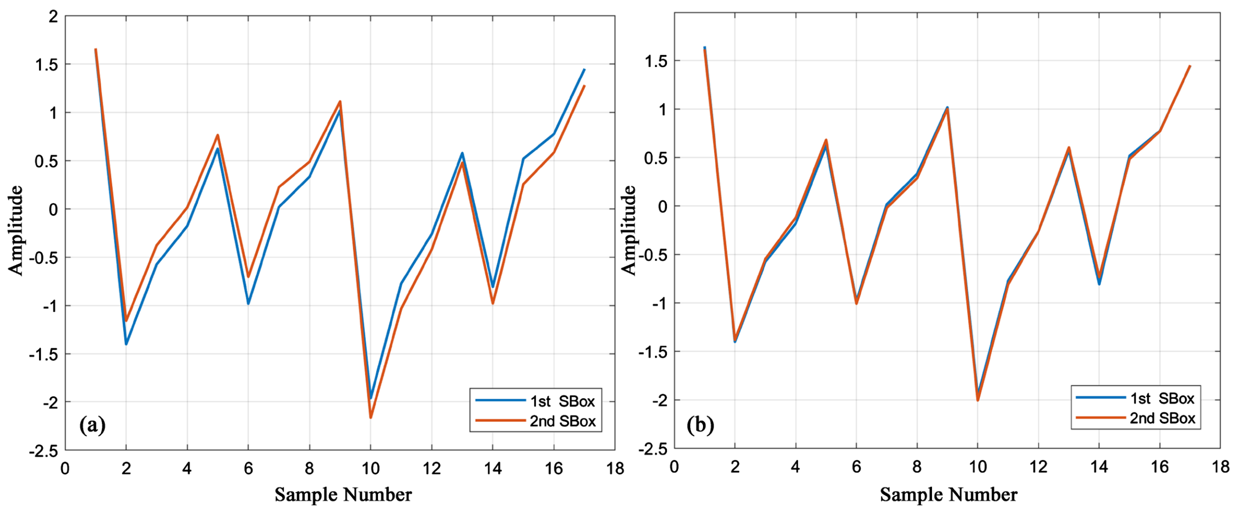

Figure 2. Power traces captured from an 8-bit microcontroller implementation of AES, which represent the first SBox and the second SBox operations in the first round. Traces look very similar if the same data is processed. (a) Different data is processed; (b) The same data is processed.

At the profiling stage, traces are divided into 16 sub-traces by analyzing the Point of Interest (POI), and each sub-trace is labeled by the corresponding SBox output. Generally, to recover the i_{th} subkey, attackers train dep-learning models on sub-traces which are labeled by the i_{th} Sbox output. In the cross-subkey training, we go to one step further by adding a small amount sub-traces which represent the other 15 SBox operations into the training set. We define the proportion of sub-traces of the target subkey to the total training set as $x \in [0,1]$. Thus, the proportion of other subkeys in the training set is $1-x$. The other 15 subkeys are equally distributed in the training set for average. Afterwards, the deep-learning model is trained on this mixed training set, as shown in Figure 1. The attack point is the input of the last round of SBox, some information in the label l_i will be changed, where the plaintext p_i is changed to the ciphertext c_i, the key is the k_i^{last} of the last round of *AddRoundKey*, and the SBox is changed to SBox^{-1}.

$$l_i = \text{SBox}^{-1}\left(c_i \oplus k_i^{last}\right)$$

At the attack stage, the trained cross-subkey model is used to classify traces and extract the target subkey.

4. Experimental Setup

In this section, we first describe two well-known datasets for side-channel attacks. Afterwards, we show how we train the deep-learning model and how we evaluate the attack efficiency.

4.1. Data Sets

Table 1 describes a summary of two databases used in our experiments.

The first database is called *AES_GPU* [20], in which electromagnetic traces are captured from an NVIDIA GeForce GT620 Graphics Processing Unit (GPU) implementation of an AES. In [20], up to 11,000 traces by a non-profiling attack to recover the key. Afterwards, Guang *et al.* [21] proposes a convolutional denoising autoencoder to further improve the attack efficiency and uses only 50 traces to recover a subkey. In summary, AES_GPU database contains 39,511 traces in total and each trace contains 15,001 samples. In our experiments, we use 35 K traces for training with with 3.5 K traces randomly set aside for validation. Also, 4511 traces are used for testing.

The second database is called *ASCAD*, in which traces are captured from an 8-bit AVR implementation of masked AES-128 by using an electromagnetic probe. This dataset is composed of 60 K traces with 100 K samples of each trace. The training set contains 50 K variable-key traces with 5 K traces randomly set aside for validation in our experiments, and 10 K fixed-key traces are used for testing, see [12] for further details.

4.2. Model Structure

Table 2 shows the structure of two deep-learning models used in our experiments, for AES_GPU and ASCAD databases respectively. Existing works use different types of deep neural networks for different attack scenarios. Multiple Layer Perceptrons (MLPs) seem to be a suitable architecture when traces are less noisy and well synchronized [6] [5] [15]. Thus, we use MLPs for the AES_GPU database as shown in **Table 1**, in which four dense layers are fully connected. For both cases, we use the Rectifed Linear Unit (ReLU) as the activation function and the optimizer is set to RMSprop.

The excellent feature extraction capabilities of Convolutional Neural Networks (CNNs) have been applied to DLSCAs. For example, in side-channel attacks' context, CNNs have been successfully applied to bypass the trace misalignment and to overcome jitter-based countermeasures [22]. CNNs were also

Table 1. A summary of two databases used in our experiments.

Database	AES_GPU	ASCAD
#Training traces	31,500	45,000
#Validation traces	3500	5000
#Testing traces	4511	10,000

Table 2. The MLP and CNN network structure used in the paper.

Target	Input layer	Convolutional layer	Dense layer	Output layer
AES_GPU	100	-	64|128|128|256	256
ASCAD	300	Size × filter = 11 × 64 Size × filter = 11 × 128 Size × filter = 11 × 256	512	256

used to break protected AES [23] [24] [25] [26]. For this reason, we use CNNs to classify traces of ASCAD database since the implementation is a masked AES. We experimentally find the optimal CNN model for ASCAD database, which contains three convolutional layers and one dense layer.

In our experiments, we use the identity power model, which assumes that the power consumption is proportional to the data processed at the attack point. Power model defines the size of a neural network's output. If the data processed at the attack point is a byte, the output size of the model is set to 256.

4.3. Training Setup

We know that data augmentation increases the amount of training data by adding minor alterations to the existing training traces. However, too many alterations in the training set may confuse the neural network. So to find the optimal amount of augmenting traces in the training set becomes a realistic problem. Thus, for each database, we build 16 different training sets, which contains different amount of augmenting traces to train 16 deep-learning models. **Figure 1** shows an example of how these training sets are built. We call these training sets from set_1 to set_{16}. Suppose the database contains x traces for training and we divide each trace to 16 segments as shown in **Figure 1**, which are related to 16 subkeys separately. So the total number of trace segments should be $16x$. To train the model for the 1_s -subkey, the training set is composed of x 1_{st}-subkey segments and y other-subkey segments. Segments of 15 non-target subkeys are equally distributed in all training sets. From set_1 to set_{16}, the ratio of the target-subkey segments to all segments is defined by $\frac{x}{x+y} \in \left\{\frac{1}{16}, \frac{2}{16}, \cdots, \frac{16}{16}\right\}$, in which set_{16} denotes the set without trace augmentation. The corresponding trained models are denoted by $M_1, M_2, \cdots, M_{16}$.

4.4. Estimation Metric

In SCAs, the common metrics to assess the performance of models are the *Partial Guessing Entropy* (PGE) and the *Success Rate* (SR).

The *rank* of the a subkey *k*, $\text{Rank}(k)$ is the number of subkey values with a higher probability than the one of *k* where *k'* denotes the a subkey candidate and *p* represents the corresponding sub-plaintext.

$$\text{Rank}(k) = \left|k' \in \mathcal{K} : \Pr[k \mid T, p] < \Pr\left[k'[T, p]\right]\right|$$

The PGE is the expected rank among all possible subkeys, which estimates the number of subkey candidates required to be evaluated for a successful attack.

$$\text{PGE}(k) = \mathop{\mathbb{E}}_{k \in \mathcal{K}}\left(\text{Rank}(k)\right)$$

However, in some cases, the attacker has a very limited access to the victim device, for example, the attacker can capture only one trace during the attack stage. In this scenario, success rate becomes a more suitable estimation metric. It evaluates the average probability that $\text{PGE}(k) = 0$ of using the deep-learning model to classify a single trace, which represents the expected probability of a successful attack with a single try [12].

5. Experimental Results

In this section, we first present the experimental results of our cross-subkey model tested on AES_GPU database. Afterwards, we show the results in ASCAD database. We use the ρ-test as the leakage detection method [27] to find the point of interest (POI) of each subkey.

5.1. Results on AES_GPU

Since traces in AES_GPU are synchronized for the last round of AES, so the attack point for AES_GPU is set to the SBox input of the last round. Figure 3(a) shows the leakage detection results by using the mentioned attack point and we locate segments for each subkey. In our experiments, each trace segment contains 100 samples. Specifically, the trace interval for the first SBox operation is $[13081:13181]$. Figure 3(b) shows how we synchronize segments for different subkeys. In this experiment, we generate 16 training sets called $set_1, set_2, \cdots, set_{16}$ according to the training method in 2. Each training set contains 31.5 K traces in total with 3.5 K first-subkey traces selected for validation. The testing set contains 4511 traces for the first subkey. Afterwards, models $M_1, M_2, \cdots, M_{16}$ are trained on the corresponding training set separately. The training batch is set to 256, and the maximum epoch number is 500. Table 3 shows the single-trace

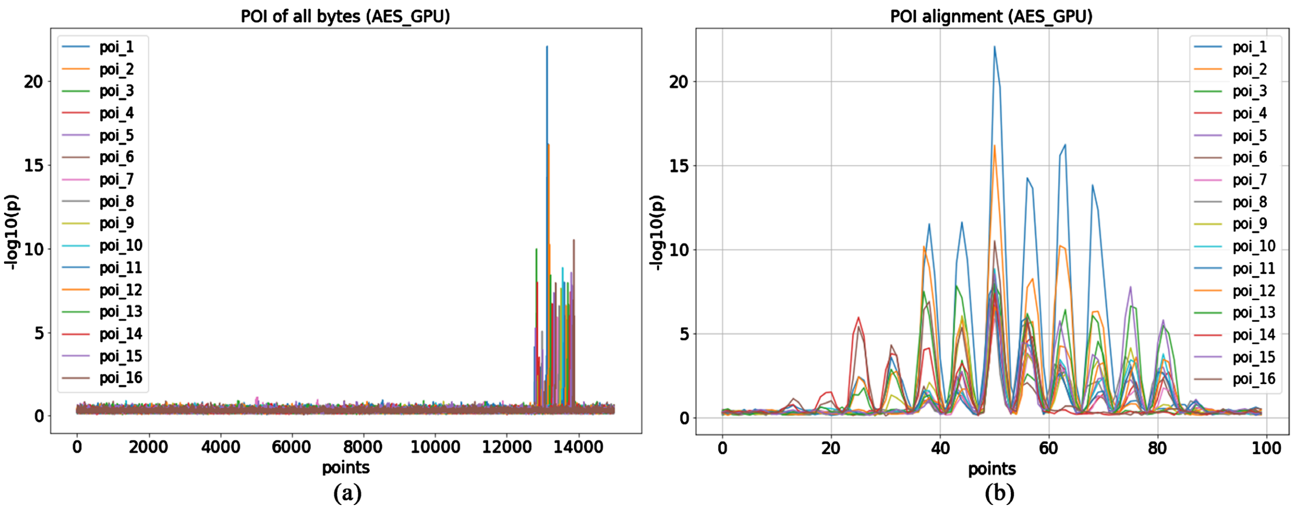

Figure 3. (a) POI of all subkeys in the last round of AES for AES_GPU (by ρ-test); (b) POI alignment.

Table 3. The accuracy of the 16 models on the test set (AES_GPU).

M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8
1.0%	0.8%	1.2%	1.7%	1.7%	1.8%	1.9%	2.0%
M_9	M_{10}	M_{11}	M_{12}	M_{13}	M_{14}	M_{15}	M_{16}
2.1%	2.0%	2.0%	2.0%	2.0%	2.0%	2.3%	2.0%

attack accuracy of these 16 models. We find that model M_{15} achieves the highest single-trace attack accuracy to recover the first subkey.

Figure 4 shows the PGE results of using model M_{15} to recover the first subkey of AES_GPU. We can see that the model requires only 17 traces to recover the subkey. Table 4 compares our results to other existing works, in which the cross-subkey model achieves a 66% improvement.

We also investigate if the addition of noise to training traces can have the same impact as the synthetically modified traces. We build a training set which contains x traces which are related to the first subkey. Afterwards, we add y white Gaussian noise traces with mean $\mu = 0$ and standard deviation $\sigma = 0.01$, with random labels range from 0 to 255. We construct a new dataset $\frac{x}{x+y} = \frac{15}{16}$ using noisy traces instead of other byte traces, and the trained model has an accuracy of only 0.6% on the test set, demonstrating the effectiveness of the cross-subkey model.

5.2. Results on ASCAD

Traces in ASCAD are captured from an 8-bit AVR implementation of masked AES-128. The attack point used to break ASCAD is the SBox output in the fisrt round of AES. Figure 5(a) shows the leakage detection of all subkeys by using ρ-test. In this experiment, the target subkey is the 3_{rd}-subkey by considering the impact caused by the mask and the corresponding trace segment interval is $[71820:72120]$, which is selected according to the leakage detection. Trace

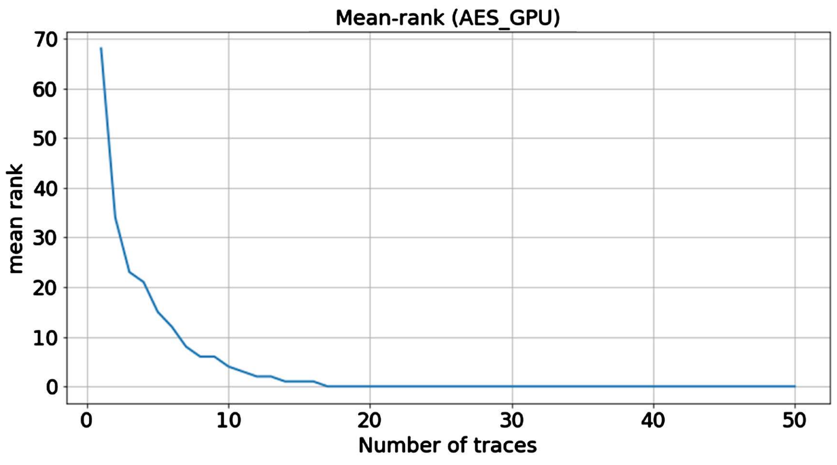

Figure 4. PGE result of using model M_{15} to recover the first subkey (AES_GPU).

Table 4. Existing results' summary (AES_GPU).

Previous attacks	#Traces for a successful attack
[20]	650
[21]	50
This work	17

segment of different subkeys contain the same number of sampling points and are synchronized based on the leakage peak.

This experiment is divided into two parts. We first assume that the mask is a known variable. Afterwards, we assume that the attacker has no access to the mask.

5.2.1. Test with a Known Mask

First we assume that the mask is known. Figure 5(b) shows how we synchronize segments for different subkeys. In this experiment, we generate 16 training sets called $set_1, set_2, \cdots, set_{16}$ according to the training method in 2. Each training set contains 45 K traces in total with 5 K 3_{rd}-subkey traces selected for validation. The testing set contains 10 k traces for the 3_{rd}-subkey. Afterwards, models $M_1, M_2, \cdots, M_{16}$ are trained on the corresponding training set separately. The training batch is set to 256, and the maximum epoch number is 200. Table 5 shows the single-trace attack accuracy of these 16 models. We find that model M_{13} achieves the highest single-trace attack accuracy to recover the 3_{rd}-subkey. Figure 6 shows that M_{13} recovery target subkey requires 2 traces. We also used the method in 2 to add noise to the training set. A new dataset was constructed for M_{13}. The model trained using this dataset had an accuracy of 57.2% on the test set, which was lower than the accuracy of M_{13} on the test set. The effectiveness of cross-subkey training was also demonstrated. Next, the mask is treated as an unknown condition in the attack phase.

5.2.2. Test with an Unknown Mask

According to the specific encryption algorithm of the ASCAD data set given by

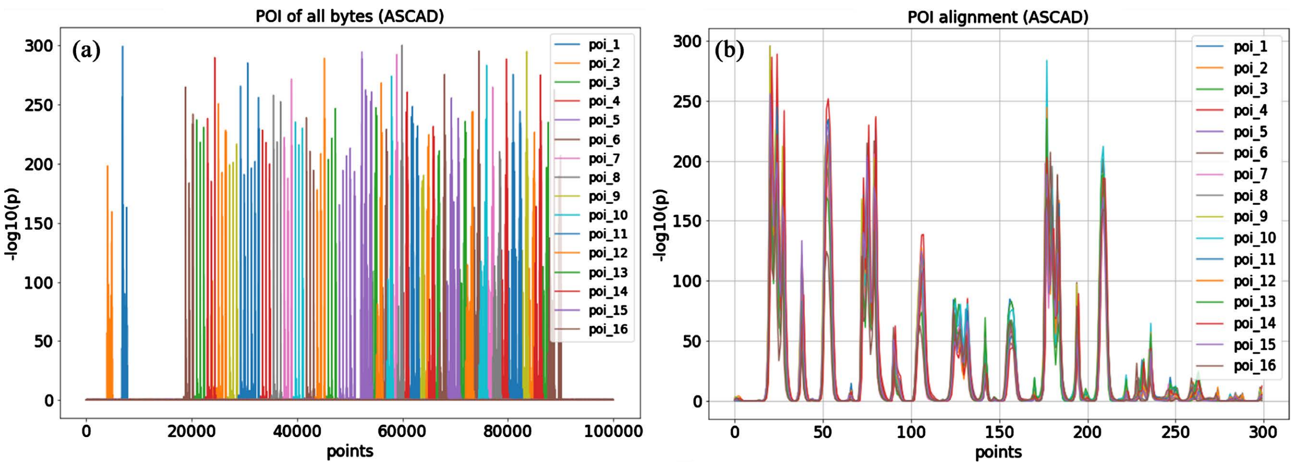

Figure 5. (a) POI of all subkeys of AES (ASCAD); (b) POI alignment.

Table 5. The accuracy of the 16 models on the test set (ASCAD).

M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8
45.6%	52.3%	56.0%	57.8%	59.8%	61.8%	63.1%	63.4%
M_9	M_{10}	M_{11}	M_{12}	M_{13}	M_{14}	M_{15}	M_{16}
63.6%	62.7%	63.6%	65.6%	67.4%	65.6%	65.3%	64.5%

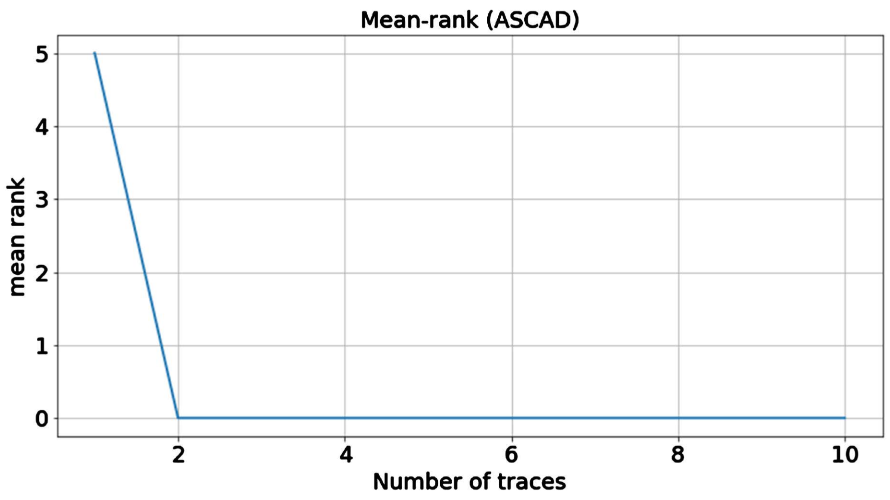

Figure 6. PGE result of using model M_{13} to recover the 3_{rd}-subkey (ASCAD).

[12], there is an obvious mask power leak in *Last AddRoundKey*. Use ρ-test to find the POI of all byte masks. According to the POI of the mask, we can build a neural network model to recover the mask information used in the attack stage. Finally, the average correct rate of each trace recovery mask on the test set is 55.7%. Therefore, the average accuracy of each trace to recover the 3_{rd}-subkey is $67.4\% \times 55.7\% = 37.5\%$.

6. Discussion

In fact, our work is just the beginning of cross-subkey training. It was demonstrated experimentally that when training the target subkey, adding data from other subkeys to the training set can enhance the generalisation of the network model. Note that the power traces are given in a combined form, where each part can be exploited by an attacker. One reason why neural networks are so successful in image classification is that the multi-dimensional features of an image create a multiplicity of image enhancements, and the ordered feature transformation of an image enhances the generalisation ability of the network. This is what we do: we use data from other subkeys as sub-samples of the target subkey, because when the other subkeys are labelled the same as the target subkey, their features have a high similarity, and by adjusting the proportion of other subkey in the training set, an optimal target subkey model can be obtained. This has been demonstrated in our experiments.

We do not need to restrict the fact that other subkeys have an impact on the target subkey to the deep learning part. As Cagli *et al.* results show that using intelligent preprocessing (e.g. data augmentation) can lead to more significant performance increases than by changing the network architecture [28]. Various data augmentation techniques have been widely used in the field of machine learning for many years, and there is no reason why these more general approaches should not be used in SCAs. Furthermore, we must mention that data augmentation is not limited to deep learning and what would happen if SCAs-specific data augmentation would be used with other, simpler machine learning techniques.

Finally, the thesis gives its own solution to the question of whether or not ASCAD knows the mask during the attack phase. Of course some papers argue that the mask should not be known during the analysis phase either, in fact, during the analysis phase we have full access to the device and this thesis argues that during the analysis phase, it is possible to have the mask information. Nevertheless, all our work effectively demonstrates that other subkeys have a positive effect on the key subkey in the near-field dataset AES_GPU and in the near-field ASCAD with the mask (again, we have done experiments on the dataset of electrical signals, which will not be repeated as they are similar to those in the paper).

When discussing the results at a more general level, we can observe a number of trends.

- Capturing the case where the amount of data does not increase, and increasing the size of the entire dataset by adding other subkeys to the dataset is a common method of data augmentation.
- A network model is trained using data from all subkeys, and this network model can recover all subkeys, recovering all subkeys much more efficiently than traditional DLSCAs (models trained for a single subkey). Models trained for specific subkeys cannot be used on other subkeys, which raise concerns about the computational cost and potential performance gains.

7. Conclusion

In the paper, we propose a cross-subkey deep-learning side-channel attack, which utilizes the additional synthetically modified power traces as a data augmentation to build models with a better generalization capacity. We show that the accuracy of the network model on the test set can be enhanced by adding other subkeys of data to the training set of target subkey. This paper uses two well-known datasets to demonstrate the effectiveness of cross-subkey training, but there are still many interesting open problems with the study of the connections between different subkeys. As mentioned in the previous sections, there are many possible directions of research to follow regarding the connection between different subkeys, which will ultimately bring more cohesion to the field and more confidence in the results obtained.

Acknowledgements

This research was supported by National Natural Science Foundation of China (No. 61973109).

Conflicts of Interest

The authors declare no conflicts of interest.

References

[1] Daemen, J. and Rijmen, V. (2002) The Advanced Encryption Standard. In: *The De-*

sign of Rijndael, Springer, Berlin, 1-8. https://doi.org/10.1007/978-3-662-04722-4_1

[2] Kocher, P., Jaffe, J. and Jun, B. (1999) Differential Power Analysis. In: Wiener, M., Ed., *Annual International Cryptology Conference*, Springer, Berlin, 388-397. https://doi.org/10.1007/3-540-48405-1_25

[3] Mangard, S., Oswald, E. and Popp, T. (2008) Power Analysis Attacks: Revealing the Secrets of Smart Cards. Vol. 31, Springer Science & Business Media, Berlin.

[4] Goodfellow, I., Bengio, Y. and Courville, A. (2016) Deep Learning. MIT Press, Cambridge, MA. http://www.deeplearningbook.org

[5] Wang, H., Brisfors, M., Forsmark, S. and Dubrova, E. (2019) How Diversity Affects Deep-Learning Side-Channel Attacks. 2019 *IEEE Nordic Circuits and Systems Conference* (*NORCAS*): *NORCHIP and International Symposium of System-on-Chip* (*SoC*), Helsinki, 29-30 October 2019, 1-7. https://doi.org/10.1109/NORCHIP.2019.8906945

[6] Das, D., Golder, A., Danial, J., Ghosh, S., Raychowdhury, A. and Sen, S. (2019) X-DeepSCA: Cross-Device Deep Learning Side Channel Attack. *Proceedings of the* 56*th Annual Design Automation Conference* 2019, Las Vegas, NV, 2-6 June 2019, Article No. 134. https://doi.org/10.1145/3316781.3317934

[7] Wang, H., Forsmark, S., Brisfors, M. and Dubrova, E. (2020) Multi-Source Training Deep Learning Side-Channel Attacks. *IEEE* 50*th International Symposium on Multiple-Valued Logic*, Miyazaki, 9-11 November 2020, 58-63. https://doi.org/10.1109/ISMVL49045.2020.00-29

[8] Kubota, T., Yoshida, K., Shiozaki, M. and Fujino, T. (2019) Deep Learning Side-Channel Attack against Hardware Implementations of AES. 2019 22*nd Euromicro Conference on Digital System Design* (*DSD*), Kallithea, 28-30 August 2019, 261-268. https://doi.org/10.1109/DSD.2019.00046

[9] Wang, H. and Dubrova, E. (2020) Tandem Deep Learning Side-Channel Attack against FPGA Implementation of AES. 2020 *IEEE International Symposium on Smart Electronic Systems* (*iSES*) (*Formerly iNiS*), Chennai, 14-16 December 2020, 147-150. https://doi.org/10.1109/iSES50453.2020.00041

[10] Kim, J., Picek, S., Heuser, A., Bhasin, S. and Hanjalic, A. (2019) Make Some Noise. Unleashing the Power of Convolutional Neural Networks for Profiled Side-Channel Analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, **2019**, 148-179. https://doi.org/10.46586/tches.v2019.i3.148-179

[11] Masure, L., Dumas, C. and Prouff, E. (2020) A Comprehensive Study of Deep Learning for Side-Channel Analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, **2020**, 348-375. https://doi.org/10.46586/tches.v2020.i1.348-375

[12] Prouff, E., Strullu, R., Benadjila, R., *et al.* (2018) Study of Deep Learning Techniques for Side-Channel Analysis and Introduction to ASCAD Database. *Cryptology ePrint Archive.*

[13] Zhang, L., Xing, X., Fan, J., Wang, Z. and Wang, S. (2019) Multi-Label Deep Learning Based Side Channel Attack. 2019 *Asian Hardware Oriented Security and Trust Symposium* (*AsianHOST*), Xi'an, 16-17 December 2019, 1-6. https://doi.org/10.1109/AsianHOST47458.2019.9006657

[14] Golder, A., Das, D., Danial, J., Ghosh, S., Sen, S. and Raychowdhury, A. (2019) Practical Approaches toward Deep-Learning-Based Cross-Device Power Side-Channel Attack. *IEEE Transactions on Very Large Scale Integration* (*VLSI*) *Systems*, **27**, 2720-2733. https://doi.org/10.1109/TVLSI.2019.2926324

[15] Wang, H. and Dubrova, E. (2020) Federated Learning in Side-Channel Analysis.

Cryptology ePrint Archive, Report 2020/902. https://eprint.iacr.org/2020/902

[16] Goodfellow, I., Bengio, Y., Courville, A. and Bengio, Y. (2016) Deep Learning. Vol. 1, MIT Press, Cambridge, MA.

[17] Wu, Y., Shen, K., Chen, Z. and Wu, J. (2020) Automatic Measurement of Fetal Cavum Septum Pellucidum from Ultrasound Images Using Deep Attention Network. 2020 *IEEE International Conference on Image Processing* (*ICIP*), Abu Dhabi, 25-28 October 2020, 2511-2515. https://doi.org/10.1109/ICIP40778.2020.9191002

[18] Breiman, L. (1996) Bagging Predictors. *Machine Learning*, **24**, 123-140. https://doi.org/10.1007/BF00058655

[19] Shorten, C. and Khoshgoftaar, T.M. (2019) A Survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, **6**, Article No. 60. https://doi.org/10.1186/s40537-019-0197-0

[20] Gao, Y., Zhang, H., Cheng, W., Zhou, Y. and Cao, Y. (2018) Electro-Magnetic Analysis of GPU-Based AES Implementation. *Proceedings of the* 55*th Annual Design Automation Conference*, San Francisco, 24-29 June 2018, Article No. 121. https://doi.org/10.1145/3195970.3196042

[21] Yang, G., Li, H., Ming, J. and Zhou, Y. (2019) CDAE: Towards Empowering Denoising in Side-Channel Analysis. In: Zhou, J., Luo, X., Shen, Q. and Xu, Z., Eds., *International Conference on Information and Communications Security*, Springer, Cham, 269-286. https://doi.org/10.1007/978-3-030-41579-2_16

[22] Cagli, E., Dumas, C. and Prouff, E. (2017) Convolutional Neural Networks with Data Augmentation against Jitter-Based Countermeasures—Profiling Attacks without Pre-Processing. Cryptology ePrint Archive, Report 2017/740. https://eprint.iacr.org/2017/740

[23] Perin, G., Ege, B. and van Woudenberg, J. (2018) Lowering the Bar: Deep Learning for Side-Channel Analysis (White-Paper). Proc. BlackHat, 1-15.

[24] Gilmore, R., Hanley, N. and O'Neill, M. (2015) Neural Network Based Attack on a Masked Implementation of AES. 2015 *IEEE International Symposium on Hardware Oriented Security and Trust* (*HOST*), Washington DC, 5-7 May 2015, 106-111. https://doi.org/10.1109/HST.2015.7140247

[25] Martinasek, Z., Dzurenda, P. and Malina, L. (2016) Profiling Power Analysis Attack Based on MLP in DPA Contest V4.2. 2016 39*th International Conference on Telecommunications and Signal Processing* (*TSP*), Vienna, 27-29 June 2016, 223-226. https://doi.org/10.1109/TSP.2016.7760865

[26] Jin, M., Zheng, M., Hu, H. and Yu, N. (2020) An Enhanced Convolutional Neural Network in Side-Channel Attacks and Its Visualization. arXiv: 2009.08898

[27] Durvaux, F. and Standaert, F.-X. (2016) From Improved Leakage Detection to the Detection of Points of Interests in Leakage Traces. In: Fischlin, M. and Coron, J.S., Eds., *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Springer, Berlin, 240-262. https://doi.org/10.1007/978-3-662-49890-3_10

[28] Cagli, E., Dumas, C. and Prouff, E. (2017) Convolutional Neural Networks with Data Augmentation against Jitter-Based Countermeasures. In: Fischer, W. and Homma, N., Eds., *International Conference on Cryptographic Hardware and Embedded Systems*, Springer, Cham, 45-68. https://doi.org/10.1007/978-3-319-66787-4_3