Noname manuscript No.
(will be inserted by the editor)

Cross-Subkey Deep-Learning Side-Channel Analysis

Fanliang Hu[1,3] · **Huanyu Wang**[2] · **Junnian Wang**[1,3]

Received: date / Accepted: date

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 are restricted such as in this paper only $5K$ power traces, deep-learning models always suffer from underfitting 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. Experimental results show that the accuracy of the subkey combination training model is 28.20% higher than that of the individual subkey training model on trajectories captured in the microcontroller implementation of the STM32F3 with AES-128. At the same time, the number of traces that need to be captured when the model is trained is greatly reduced, demonstrating the effectiveness and practicality of the method.

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 [5–8] and hardware implementations [9–12] of AES have been presented. [5] 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 surpass the state-of-the-art result in ASCAD database [5]. In [6], 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

✉ jnwang@mail.hnust.edu.cn

[1] School of Physics and Electronic Science, Hunan University of Science and Technology, Xiangtan, China;
[2] School of EECS, KTH Royal Institute of Technology, Stockholm, Sweden;
[3] Hunan Provincial Key Laboratory of Intelligent Sensors and Advanced Sensor Materials, Xiangtan 411201, Hunan, China;

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, 8, 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 $k_i \in \mathcal{K} = \{0, 1, \ldots, 255\}$, called *subkeys*, for $i \in \{1, 2, \ldots, 16\}$. 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.

However, when the number of training traces are not sufficient, deep-learning models always suffer from underfitting. 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 is 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 that uses multiple subkeys rather than a single subkey to build models with better fitting 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. Our current results show that (1) the number of traces in the training set remains unchanged ($5K$), and the accuracy of the model is improved by an average of 6.52% by adding other non-target subkey related traces by changing the composition of the training set (i.e. changing the proportion of target subkey traces and non-target subkey traces). (2) Adding other non-target subkey related traces to expand the number of training sets improved the accuracy of the model by an average of 28.20%.

2 Background

This section first reviews AES-128. Afterwards, we briefly introduces 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 algorithm 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 points of attack are the first round of SBox output and the last round of SBox input of the AES algorithm. 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 $\mathcal{T} = \{ \mathrm{T}_1, \mathrm{T}_2, ..., \mathrm{T}_m \}$ 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 $l(\mathrm{T}_i) \in L$,

where $L = \{0, 1, \ldots, 255\}$, 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 $\mathcal{N} : \mathbb{R}^m \rightarrow \mathbb{I}^{|L|}$ and the output is a *score* vector $S = \mathcal{N}(\mathrm{T}) \in \mathbb{I}^{|L|}$. The element s_j with value j in S represents the probability that $l(\mathrm{T}) = j$.

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).

$$k_i = \underset{0 \leq j \leq 255}{\arg\max}\ \tilde{s}_j. \tag{1}$$

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].

$$\mathcal{P}_{total} = \mathcal{P}_{data} + \mathcal{P}_{op} + \mathcal{P}_{noise} + \mathcal{P}_{const} \tag{2}$$

3 Cross-Subkey Attack

Figure 1 shows an overview of how the cross-subkey model is trained, where the collaborative use of different subkeys provides more feature information to the model, providing a better fit to 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 underfitting with insufficient training measurements. Underfitting refers to the network being trained with a small number of samples, perhaps with fewer features being extracted from the training samples, resulting in a trained model that does not match the test set well and performs poorly, even if the samples themselves cannot be identified efficiently. 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 solution to the problem of limited training data is data augmentation [19], which aims to use the additional synthetically modified traces to act as a regularizer and helps enhance the fit when training models in the context of side-channel attacks.

In software implementations of AES, leakage is time-dependent since instructions are carried out one by one [10]. 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 are the same. Fig. 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 fitting capacity.

3.2 Cross-Subkey model training

As shown in Fig 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.

$$l_i = SBox(p_i \oplus k_i) \tag{3}$$

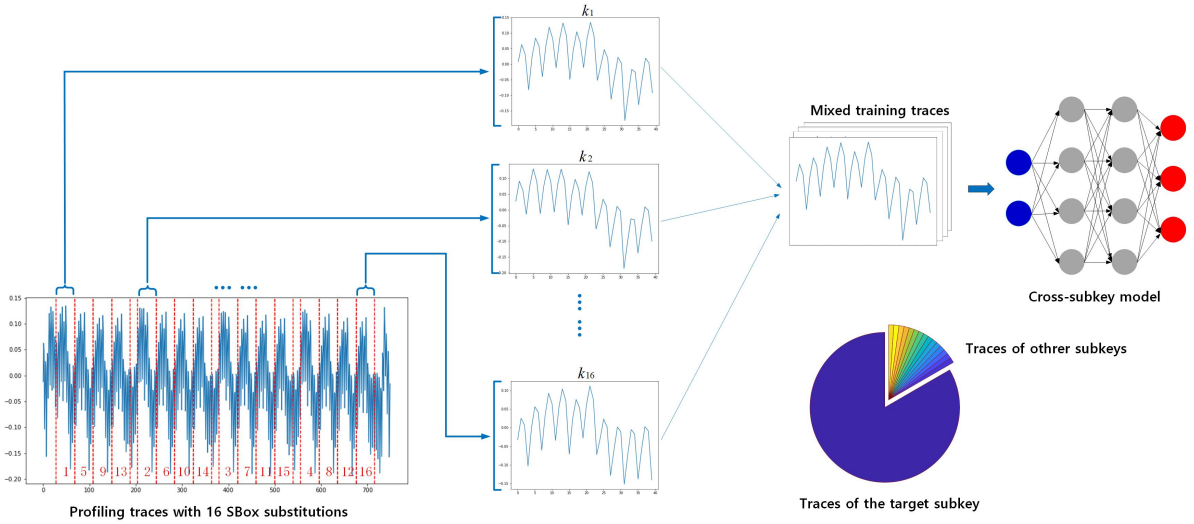
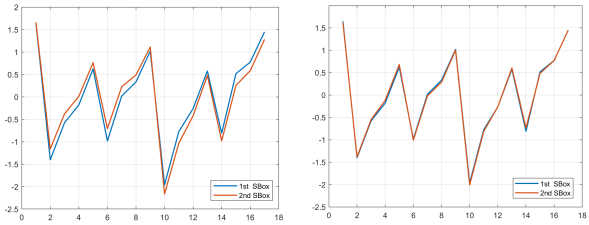


Fig. 1 An overview of how the cross-subkey model is trained on the mixed profiling set.



(a) Different data is processed

(b) The same data is processed

Fig. 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.

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 divided the experiment into two parts (notice: the number of training sets in this paper is $5K$):

- **Verifying the validity of cross-subkey training** (total training set $5K$ constant). We define the proportion of subtraces of the target subkey to the total training set as $x \in [1, 16]$. Thus the proportion of other subkeys in the training set is $16 - x$. The other 15 subkeys are average distributed in the training set.
- **Applying cross-subkey training** (total training set is increased by $5K$ at a time). We use all the power traces of the target subkey ($5K$ in this paper) for training, and add an equal number of power traces ($5K$) to the training set at a time as the number of target power traces, which are provided by the other 15 subkeys. The training set is thus $5K \times y(y \in [1, 16])$, where $5K \times (y - 1)$ is equally distributed in the training set by the other 15 subkeys.

4 Experimental Setup

In this section, we first present the dataset used for the experiments. We then show how we trained the deep learning model and how we evaluated the efficiency of the attack.

4.1 Data Sets

The dataset used in the paper is captured by a ChipWhisperer-Lite [20] device at a sampling frequency of 40MHz. The experimental target cryptographic board is the CW308T-STM32F3, and the target cryptographic chip is the Arm Cortex M4, which runs the cryptographic algorithm TinyAES. The encryption mode of operation is the Electric Code Book (ECB) mode. For the first round of the AES algorithm $7K$ power traces are captured as the data used for the experiments. Of these, $6K$ uses random plaintexts and random keys, $5K$ is used as the training set and $1K$ is used as the validation set. The remaining $1K$ are used as the test set for the experiments using fixed-key random plaintexts. Each power trace has 750 sampling points and contains all *SubBytes* from the first round. This is shown by Fig 3.

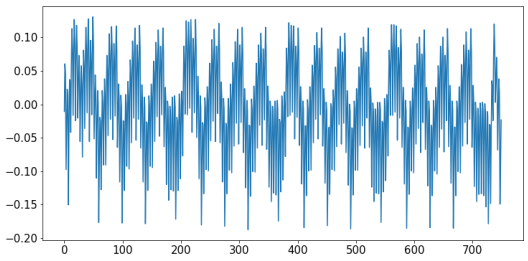


Fig. 3 An example trace representing the first round of AES *SubBytes*.

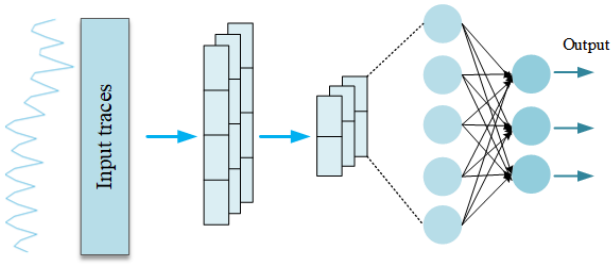


Fig. 4 An example trace representing the first round of AES *SubBytes*.

4.2 Model structure

The structure of the network model used in this work is shown by Fig 4. After passing through the Input Layer, the traces are connected to a Convolutional Layer with a step size of 5 and a neuron count of 16. After passing through an Average Pooling Layer with a pooling step of 3, there are expanded by Flatten and then connected to two Dense Layers with 256 neurons. The last of these Dense Layers is activated by Softmax and is used to generate 256 output predictions. The Activation Functions of the other layers in the network model are all with Rectified Linear Unit (Relu).

4.3 Training setup

We divide the experiment into two parts the first part in order to demonstrate that the inclusion of sub-traces of non-target subkeys positively influences the training of the model, and the second part for the application of the cross-subkey approach to the experiment.

Part I 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. Fig. 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 Fig. 1, which are related to 16 subkeys separately. So the total number of trace segments should be $16 \times x$. To train the model for the target subkey, the training set is composed of x target 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 $\frac{x}{x+y} \in \{\frac{1}{16}, \frac{2}{16}, ... \frac{16}{16}\}$, in which set_{16} denotes the set without trace augmentation. The corresponding trained models are denoted by $M_1, M_2, ..., M_{16}$.

Part II In image classification, data enhancement methods are often used such as cropping, rotating, flipping, deflating and shifting [21]. These methods are essentially a series of changes to the original data in order to expand the number of training sets on which the model is trained. In the Part I, we do not change the number of data sets on which the model is trained. The main work in this part is to use all the traces of the target subkey and expand the training set with other subtraces of non-target subkeys for the purpose of data augmentation. Assuming that the database contains x training traces, similar to the work in Part I, we will also train 16 models. The training set of 16 models is denoted by $\tilde{set}_1$ to $\tilde{set}_{16}$. The amount of data in $\tilde{set}_y (y \in [1, 16])$ is $x \times y$, where $(16-y) \times x$ is equally distributed by the sub-traces of non-target subkeys. $\tilde{set}_1$ denotes all traces of the target subkey x (no sub-traces of other subkeys), and $\tilde{set}_{16}$ denotes all traces of all subkeys $16 \times x$. The corresponding trained models are denoted by $\tilde{M}_1, \tilde{M}_2, ..., \tilde{M}_{16}$

4.4 Evaluation Metrics

Model accuracy is defined as the probability of a model achieving correct classification results on a test set. As one of the most commonly used model evaluation metrics in machine learning, model is used to characterise a model's ability to classify data. An increase in model accuracy accuracy indicates that the backpropagation algorithm's optimization of the weights and bias parameters gradually converges to the correct values, and the model gradually converges to the optimal model. The loss of a model characterises the degree of deviation between a model's predicted and actual values. The smaller the loss, the closer the model's prediction is to the actual value. The loss function used in this experiment is the *Categorical Crossentropy*. The formula for the accuracy of the model is:

$$acc(X_{attack}) = \frac{|\{x_i \in X_{attack}\}\tilde{k}|}{X_{attack}}. \quad (4)$$

Where X_{attack} denotes the test dataset, x_i denotes the ith power trace in that dataset, $\tilde{k}$ denotes the calculation result, and $x_i \in X_{attack}$ is the set when the guess keys are all equal to the correct key. The model's accuracy is the ratio of the number of power traces when the guessed key is equal to the correct key to the number of power traces in all the test sets.

5 Experimental Results

In this section we first present the experimental results of testing the crossover subkey model with a constant number of training sets. Afterwards, we show the results of the increased number of traces in the training set. We use the ρ-test as a leak detection method [22] to find the point of interest (POI) for each subkey. The power consumption model used in this paper is the identity model [6].

5.1 Results for constant number of training set traces

Since the traces we have captured from the STM32F3 microcontroller implementation of AES-128 are the first round of AES, the attack point is set to the output of the first round of SBox. Fig 5 shows the segments we located for each byte by using the attack point detection results described. In the experiments, each trace segment contained 40 sample points. Specifically, the trace segment for the first SBox operation is $[28:68]$ (we use the first subkey as the target subkey). Fig 6 shows how we allowed to synchronise segments for different bytes of the subkeys. In this experiment we generated 16 training sets, called set_1, set_2, ..., set_{16}, based on the training method in 4.3. Each training set contains $5K$ traces, with $1K$ of data for the target subkey as the validation set, which will be saved during model training when the model is at its highest accuracy in the validation set. The test set is the one containing $1K$ traces of the target subkey, and we also tested the other subkeys, which also contained $1K$ traces of the corresponding subkeys. Afterwards, model M_1, M_2, ..., M_{16} is trained on the corresponding training set respectively. The training batch_size are set to 256 and the maximum number of epochs is 500 and the learning rate is 0.0005. Since the optimiser RMSprop is random in updating parameters, we have trained each model 10 times and taken the mean value as the experimental result. Table 1 shows the accuracy of the 16 models on the

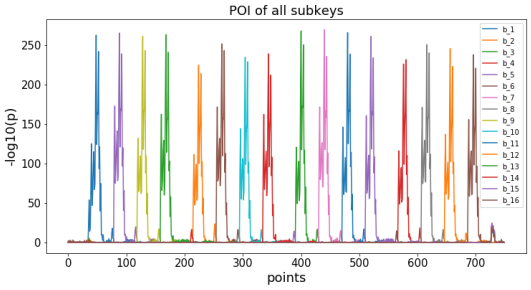


Fig. 5 POI for all subkeys in the first round.

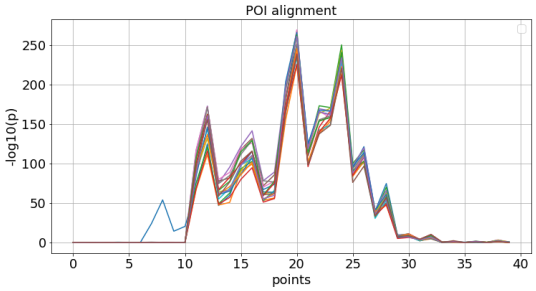


Fig. 6 POI alignment.

full test set of subkey. where $M_i(i \in [1,16])$ denote the model and $S_i(i \in [1,16])$ denote the test set of different subkeys, e.g. the first column in the first row shows the accuracy of M_1 on the test set of the first subkey (accuracy figures are in percentages, with the % omitted at the end).

We found that model M_{15} had the highest accuracy on the test set of the first subkey. Because the training set of model M_{16} is the full trace of the first subkey, a model trained by means of cross-subkey will be 6.52% more accurate than a model trained traditionally on a one-to-one approach. Next, we show the results of the trace number increase in the training set.

5.2 Results of increasing number of training set traces

Again in this subsection the first subkey is used as the target subkey. In contrast to 5.1 the number of training sets for each model is increasing when training the cross-subkey model, with the training set being increased by $5K$ traces at a time, and these $5K$ traces being equally distributed among the sub-traces of the other non-target subkeys. Where the training set for $\tilde{M}_1$ is all the traces of the first subkey and the training set for $\tilde{M}_{16}$ is all the traces of all subkeys. The model $\tilde{M}_i(i \in [1,16])$ is then trained on the corresponding dataset. The other hyperparameters are the same as 5.1. Finally each model is trained 10 times and the results on the test sets of different subkeys are taken

Table 1 Results for 16 models on a test set of 16 subkeys (No change in the number of training sets).

	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}	M_{11}	M_{12}	M_{13}	M_{14}	M_{15}	M_{16}
S_1	5.86	8.74	9.56	11.01	11.78	12.77	11.96	13.91	13.83	14.00	14.13	14.69	16.70	18.46	21.27	14.75
S_2	5.07	5.16	5.14	4.73	4.09	5.69	4.56	3.52	3.12	3.47	3.14	2.77	2.70	2.47	2.00	0.34
S_3	6.31	6.55	6.53	6.40	6.12	6.48	4.30	5.07	5.45	4.17	4.37	4.49	4.22	3.02	2.33	0.36
S_4	4.87	5.81	6.09	5.72	5.18	5.19	4.42	4.19	4.21	4.34	2.94	3.90	3.00	2.22	2.11	0.36
S_5	7.62	8.15	8.46	8.77	8.17	8.41	8.43	6.38	7.02	6.44	6.20	5.35	4.16	3.53	4.10	1.95
S_6	5.93	7.28	6.76	6.82	6.09	6.61	6.68	5.41	6.11	5.49	5.07	4.63	4.29	3.37	2.80	1.05
S_7	7.99	8.92	7.43	8.44	7.90	8.38	7.67	6.41	6.23	6.09	5.66	5.50	5.40	3.61	3.93	0.72
S_8	6.22	7.40	6.31	7.15	7.00	7.16	6.86	6.24	6.38	5.16	4.49	5.14	4.24	2.62	2.90	0.88
S_9	5.22	6.72	6.13	6.60	6.51	5.85	7.37	5.97	6.20	5.99	7.03	5.80	6.95	7.34	5.78	1.63
S_{10}	4.08	4.27	4.66	3.88	3.92	3.89	4.38	3.93	3.84	3.43	3.95	3.12	3.76	3.19	1.82	0.51
S_{11}	4.18	5.42	6.52	4.78	4.67	4.91	4.77	4.44	4.27	3.30	3.96	4.07	4.55	3.88	3.04	0.77
S_{12}	3.79	4.93	5.30	4.96	4.09	4.11	5.15	4.68	4.25	3.74	3.88	3.34	3.74	3.82	3.21	0.84
S_{13}	7.12	7.73	6.85	7.09	7.09	7.05	7.58	6.55	6.37	5.66	5.16	4.71	5.08	4.37	3.60	0.71
S_{14}	5.11	6.01	5.30	5.43	5.89	5.19	5.91	5.36	4.71	3.87	3.52	3.76	2.81	2.53	1.59	0.35
S_{15}	5.77	6.91	6.52	6.53	6.16	6.67	6.16	5.79	4.91	4.46	4.01	4.70	3.35	3.55	2.58	0.50
S_{16}	5.38	5.83	5.84	6.05	5.78	5.43	6.46	6.49	5.24	4.71	3.65	4.12	2.96	2.21	2.19	0.76

as the mean value for the experimental results. Table 2 shows the accuracy of the 16 models on the test set of all subkeys, where $\tilde{M}_i$ denotes the model and $S_i(i \in [1, 16])$ denotes the test set of different subkeys (accuracy figures are in percentages, with the % omitted at the end).

We found that model $\tilde{M}_{10}$ had the highest accuracy on the test set of the first subkey. It is 28.20% more accurate than the traditional one-to-one trained model $\tilde{M}_1$ on the test set of the first subkey.

5.3 Discussion

We set up two experiments. In Experiment I, the number of traces in the training set used when each model is trained is constant, and what is changed is the proportion of subtraces of the target subkey and subtraces of the non-target subkey in the training set. Because the model structure and hyperparameters are identical for the 16 models, only one independent variable, the training set, is used during the experiments. Our experimental results show that when the number of target subkey traces is $\frac{15}{16}$ of the training set and the number of non-target subkey traces is $\frac{1}{16}$ of the training set, the trained model M_{15} is 6.52% more accurate than the model trained using all target subkey traces. Because of the random nature of the iterative process of the parameters during the training of the neural network, we have repeated the training 10 times for each model and took the average accuracy of each model on the test set with different subkeys as the experimental results.

Experiment I is designed to validate the effectiveness of the cross-subkey training model. Model $\tilde{M}_1$ is trained using the full trace of the target subkeys. Model $\tilde{M}_i(i \in [2, 16])$ is trained using a training set that is expanded with sub-traces of non-target subkeys. Model $\tilde{M}_{10}$ had the highest accuracy of 42.95% on the test set of target subkeys, which is 28.20% more accurate than model $\tilde{M}_1$ on the test set of target subkeys. The results of Experiment II showed that by using the non-target subkeys traces to expand the training set obtained 2-fold better results than the model trained with the target subkeys.

Finally, when training the model, if a trace of a non-target subkey is added to the training set, the model is equally effective on the test set of non-target subkeys. This result suggests that the traditional approach of one model recovering one subkey can be replaced by one model recovering all subkeys.

6 Conclusion

In this 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 fitting capability. Our results show that the accuracy of the model on the test set can be improved by adding traces of other subkeys to the training set of the target subkeys when the traces of the capture are limited. This paper validates the effectiveness of the cross-subkey training model on the STM32F3 microcontroller implementation of AES-128 captured traces, but there are still many open rows for the links between different subkeys. As mentioned in the previous sections, there are many possible directions of research regarding the connections between different subkeys, which will ultimately bring more cohesion to the field and more confidence in the results obtained.

7 Acknowledgements

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

Table 2 Results for 16 models on a test set of 16 subkeys (Increasing number of training sets).

	$\tilde{M}_1$	$\tilde{M}_2$	$\tilde{M}_3$	$\tilde{M}_4$	$\tilde{M}_5$	$\tilde{M}_6$	$\tilde{M}_7$	$\tilde{M}_8$	$\tilde{M}_9$	$\tilde{M}_{10}$	$\tilde{M}_{11}$	$\tilde{M}_{12}$	$\tilde{M}_{13}$	$\tilde{M}_{14}$	$\tilde{M}_{15}$	$\tilde{M}_{16}$
S_1	14.75	21.56	27.16	28.79	31.52	31.47	34.78	32.55	38.53	42.95	37.37	37.39	37.63	36.78	39.08	39.85
S_2	0.35	4.76	9.09	11.87	14.53	15.94	18.26	18.81	19.12	21.81	23.29	23.54	24.80	25.40	27.66	25.93
S_3	0.36	8.51	12.49	14.15	14.90	15.53	19.07	18.97	20.01	23.08	23.42	23.92	25.67	26.14	28.50	26.81
S_4	0.36	9.05	12.79	13.21	17.42	16.69	19.76	18.97	21.78	23.88	26.54	23.98	26.13	27.75	28.63	23.72
S_5	1.94	11.87	14.30	17.13	18.29	20.73	25.59	23.23	27.39	30.20	27.02	32.56	30.54	31.55	33.88	35.64
S_6	1.01	9.29	12.59	11.50	14.14	16.27	19.16	19.22	19.98	19.70	23.54	23.85	25.02	28.15	25.87	29.85
S_7	0.72	9.73	13.04	12.75	13.22	13.74	16.97	15.79	18.17	18.47	22.88	23.15	22.60	24.07	23.09	26.41
S_8	0.88	9.99	11.78	12.76	12.62	14.01	15.95	16.12	15.96	17.21	20.17	21.25	19.54	22.54	20.39	23.23
S_9	1.63	7.82	11.05	10.15	14.07	14.26	17.01	17.29	21.22	21.43	21.84	23.19	25.18	24.42	27.91	31.51
S_{10}	0.51	5.90	8.01	8.57	11.38	9.62	13.19	11.88	13.20	15.25	14.42	13.48	15.96	14.69	18.18	19.11
S_{11}	0.75	6.73	8.59	9.87	13.02	12.30	15.92	14.51	17.44	17.78	17.38	17.77	20.81	16.94	22.61	22.76
S_{12}	0.84	6.68	8.76	10.96	14.54	14.39	18.81	17.48	20.81	21.16	20.77	22.13	24.22	21.71	27.06	26.44
S_{13}	0.71	8.06	13.25	17.11	20.66	20.61	27.53	25.78	29.17	31.85	31.66	33.89	33.30	33.62	34.25	37.10
S_{14}	0.28	7.22	11.56	14.28	16.60	16.24	21.85	19.31	21.68	23.34	24.86	23.41	26.35	25.83	26.51	28.52
S_{15}	0.50	7.38	12.51	13.42	15.02	15.23	20.98	19.04	21.68	22.94	23.98	24.71	26.01	24.45	26.94	28.14
S_{16}	0.76	8.47	12.28	14.98	16.28	18.00	21.66	21.69	24.06	25.03	25.49	28.08	28.01	27.31	28.60	28.07

References

1. J. Daemen and V. Rijmen, *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer, 2002.
2. P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Annual international cryptology conference*. Springer, 1999, pp. 388–397.
3. S. Mangard, E. Oswald, and T. Popp, *Power analysis attacks: Revealing the secrets of smart cards*. Springer Science & Business Media, 2008, vol. 31.
4. I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.
5. 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," *ANSSI, France & CEA, LETI, MINATEC Campus, France. Online verfügbar unter https://eprint. iacr. org/2018/053. pdf, zuletzt geprüft am*, vol. 22, p. 2018, 2018.
6. H. Wang, M. Brisfors, S. Forsmark, and E. Dubrova, "How diversity affects deep-learning side-channel attacks," in *2019 IEEE Nordic Circuits and Systems Conference (NORCAS): NORCHIP and International Symposium of System-on-Chip (SoC)*. IEEE, 2019, pp. 1–7.
7. D. Das, A. Golder, J. Danial, S. Ghosh, A. Raychowdhury, and S. Sen, "X-deepsca: Cross-device deep learning side channel attack," in *Proceedings of the 56th Annual Design Automation Conference 2019*, 2019, pp. 1–6.
8. H. Wang, S. Forsmark, M. Brisfors, and E. Dubrova, "Multi-source training deep learning side-channel attacks," *IEEE 50th International Symposium on Multiple-Valued Logic*, 2020.
9. T. Kubota, K. Yoshida, M. Shiozaki, and T. Fujino, "Deep learning side-channel attack against hardware implementations of aes," in *2019 22nd Euromicro Conference on Digital System Design (DSD)*. IEEE, 2019, pp. 261–268.
10. H. Wang and E. Dubrova, "Tandem deep learning side-channel attack against fpga implementation of aes." *IACR Cryptol. ePrint Arch.*, vol. 2020, p. 373, 2020.
11. J. Kim, S. Picek, A. Heuser, S. Bhasin, and A. Hanjalic, "Make some noise. unleashing the power of convolutional neural networks for profiled side-channel analysis," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 148–179, 2019.
12. L. Masure, C. Dumas, and E. Prouff, "A comprehensive study of deep learning for side-channel analysis," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 348–375, 2020.
13. L. Zhang, X. Xing, J. Fan, Z. Wang, and S. Wang, "Multi-label deep learning based side channel attack," in *2019 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*. IEEE, 2019, pp. 1–6.
14. A. Golder, D. Das, J. Danial, S. Ghosh, S. Sen, and A. Raychowdhury, "Practical approaches toward deep-learning-based cross-device power side-channel attack," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 12, pp. 2720–2733, 2019.
15. H. Wang and E. Dubrova, "Federated learning in side-channel analysis," Cryptology ePrint Archive, Report 2020/902, 2020, https://eprint.iacr.org/2020/902.
16. I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1.
17. Y. Wu, K. Shen, Z. Chen, and J. Wu, "Automatic measurement of fetal cavum septum pellucidum from ultrasound images using deep attention network," in *2020 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2020, pp. 2511–2515.
18. L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.
19. C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, p. 60, 2019.
20. C. O'Flynn and Z. D. Chen, "Chipwhisperer: An open-source platform for hardware embedded security research," in *International Workshop on Constructive Side-Channel Analysis and Secure Design*. Springer, 2014, pp. 243–260.
21. B. R. Frieden, "Image enhancement and restoration," in *Picture processing and digital filtering*. Springer, 1975, pp. 177–248.
22. F. Durvaux and F.-X. Standaert, "From improved leakage detection to the detection of points of interests in leakage traces," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2016, pp. 240–262.