Multi-Leak Deep-Learning Side-Channel Analysis

Fanliang Hu[1], Huanyu Wang[2], Junnian Wang[1]*
[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;
Email: {fanliang, jnwang}@mail.hnust.edu.cn; huanyu@kth.se

***Abstract*—Deep Learning Side-Channel Attacks (DLSCAs) have become a realistic threat to implementations of cryptographic algorithms, such as Advanced Encryption Standard (AES). By utilizing deep-learning models to analyze side-channel measurements, the attacker is able to derive the secret key of the cryptographic alrgorithm. However, when traces have multiple leakage intervals for a specific attack point, the majority of existing works train neural networks on these traces directly, without a appropriate preprocess step for each leakage interval. This degenerates the quality of profiling traces due to the noise and non-primary components. In this paper, we first divide the multi-leaky traces into leakage intervals and train models on different intervals separately. Afterwards, we concatenate these neural networks to build the final network, which is called multi-input model. We test the proposed multi-input model on traces captured from STM32F3 microcontroller implementations of AES-128 and show a 2-fold improvement over the previous single-input attacks.**

***Index Terms*—AES, Deep learning, Multiple leakage, Multi-input model, Side-channel attacks**

I. Introduction

Side Channel Attacks (SCAs) [1] were proposed 20 years ago, and have become a realistic concern recently with the help of deep-learning techniques. By analysing the unintentional physical leakage during the execution of the cryptographic algorithms, SCAs are able to break ciphers that are assumed to be mathematically secure. Once the secret key is extracted, the ciphertext can be decrypted and the signature can be forged, which is particularly threatening. Since Kocher introduced the first attack which is based on time consumption traces in 1996 [1], many other types of leakages have been used. For example, leakages via acoustic channels [2], power consumption [3], electromagnetic (EM) emissions [4], [5], and photon emissions [6] are now widely studied.

In most cases, a well-trained deep-learning [7] classifier is able to use fewer side-channel measurements (traces) to recover the secret key from an implementation of AES than the traditional signal processing approaches. Since deep learning models are good at extracting features from raw data, they can help attackers to find correlations between physical measurements and the internal state of the processed algorithm. Deep-learning techniques start helping power analysis in 2013 [8], in which a three-layer MLP network is trained to break a Smart Card implementation of AES-128 which contains an 8-bit microcontroller PIC16F84 [9]. Subsequently, many softwares [5], [10]–[12] and hardwares [13]–[16] implementations of AES have been broken by DLSCAs. In [17], Cagli et al. evaluated the CNN network's performance in datasets with jitter based countermeasure. In [10], Huanyu et al. studied the impact of how diversity of target chips affects side-channel attacks. In [18], the influence of the depth of the neural networks on DLSCAs were studied by visualising the heatmap. These papers provide a strong evidence for the effectiveness of deep learning techniques in the context of side channel attacks.

In most existing DLSCAs, neural networks are trained by using the value at the attack point. Once the value at attack point is recovered, the key can be derived. At the profiling stage of DLSCAs, models are trained to learn a leakage profile between side channel traces and the value at the attack point (an attack point is an intermediate value which can be used to describe the power consumed by the victim device during the execution of a cryptographic algorithm). In software implementations of AES, the attack point is usually set to the output of the *SubBytes* operation of the first and last round of AES, in which a lookup table called SBox is used. Most existing deep learning side channel attacks train models on traces which contain all leakage points directly or mainly focus on the main leakage point. They ignored the fact that leakage points appear in small trace segments grouply in some cases. A dedicated model which makes use of all these leakage intervals may potentially further increase the attack efficiency. Therefore, we propose a multi-input model to explore the benefit of using multiple leakage intervals collaboratively.

A. Our Contributions

In this paper, our main contributions are summarized below:

- We find that in software implementations of AES-128, a chosen attack point can lead to multiple leakage intervals in the traces.
- We propose a multi-input deep-learning model in which multiple leakage intervals could be used collaboratively to perform the attack. For the proposed model, we investigated the effect of different fusion techniques on the multi-input model in terms of classification accuracy.
- We expenrimetally show that the proposed multi-input model is capable of outperforming the coventional single-input approach. The results on traces captured from STM32F3 microcontroller implementations of AES-128 show a 2-fold improvement over the previous attack. Three datasets are used for validation.

B. Paper Organization

The rest of the paper is organised as follows. Section II discusses how a single attack point causes multiple leakage

intervals in traces for software implementations of AES and introduces the multi-input model. Section III describes three datasets used in our experiment and shows the results. Section V concludes this paper.

II. Multi-leakage and Multi-input Model

In this section, we first explains why a specific attack point can have multiple leakage intervals and uses different attacks as examples. Afterwards, we show the network structure of the proposed multi-input model.

A. Leakage Analysis

Power consumption of software implementations of AES is mainly derived from the bit transitions in the CMOS cells. Thus data processed in the device dominate its power dissipation. As previous researchs of SCAs [19], [20], attackers have commonly chosen the output of the SBox as the attack point on software implementations of AES, based on the fact that the non-linear output of the SBox has a higher level of confusion.

However, in AES not only the output of the SBox that can be used as an attack point, other phases of key-related intermediate values can also be used. Fig.1 shows the flow of AES-128 algorithm and some potential attack points which can be used for the key recovery. AES-128 requires a total of 10 rounds of encryption, each round consists of four basic steps, which are *SubBytes*, *ShiftRows*, *MixColumns* and *AddRoundKey*. The last round encryption doesn't have *MixColumns* procedure. 1, F_leak represents the intermediate value which is related to the plaintext and the initial key. L_leak denotes the point which is related to the ciphertext and $10th$ round key.

To locate the leakage intervals for a specific attack point in traces, we utilize Correlation Power Analysis (CPA). In general, CPA calculates the Pearson Correlation Coefficient [21] between real traces and modeled power consumption. Afterwards, the attacker find the key value which correlates best to the measured traces. Currently, there are three commonly used power models: Identity (ID), Hamming weight (HW) and Hamming distance (HD). For example, the ID model assumes the power consumption is proportional to the value at the attack point.

In Fig.1, we denote seven potential attack points for breaking software implementations of AES. $F\_leak\ 1$ represents the *AddRoundKey*'s output before the first round of AES-128.

Next, we introduce the leakage function. A leakage function is used to obtain the value related to an attack point based on a specific power model to describe the leakage. In our case, the power model is set to ID model. When $F\_leak\ 1$ is used as the attack point, the leakage function $V_{F\_leak\ 1}$ is denoted as:

$$V_{F\_leak\ 1} = Pt \oplus Key\ 0 \tag{1}$$

where Pt represents the plaintext and $Key\ 0$ represents the original key. We use $Key\ i (i \in [1, 10])$ to denote the ith round key which is derived from $Key\ 0$ by using a Key Expansion algorithm [22].

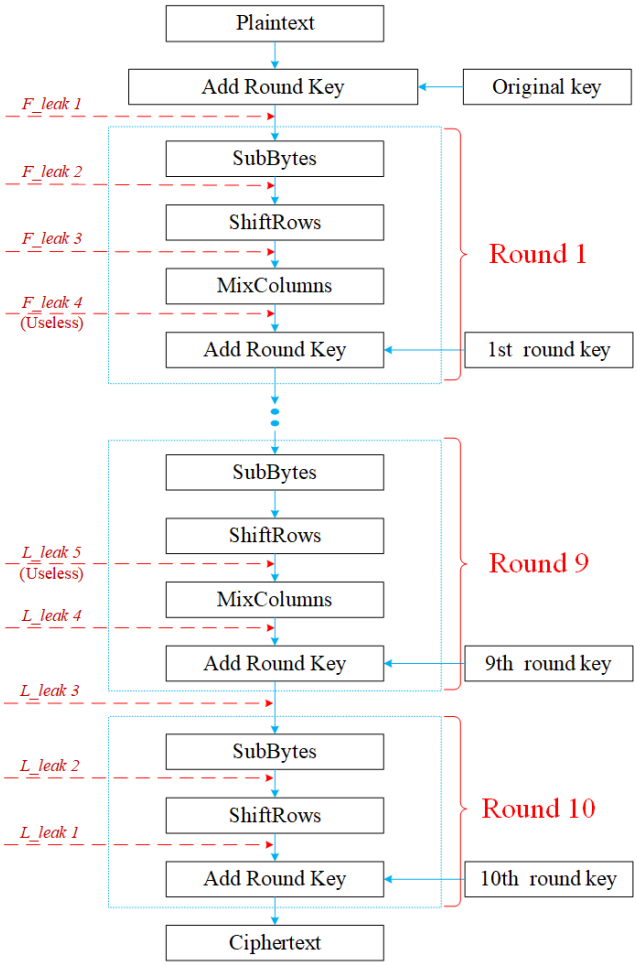


Fig. 1. The flow of AES-128 and some potential attack points (intermediate values of $F\_leak\ 4$ and $L\_leak\ 5$ have keys of different subkeys, which cannot be used by attackers and are only used for experimental analysis).

In Fig.1, $F\_leak\ 2$ represents the output of the *SubBytes* in the first round of AES (the output of the SBox), and the leakage function $V_{F\_leak\ 2}$ for $F\_leak\ 2$ is expressed as:

$$V_{F\_leak\ 2} = SBox(Pt \oplus Key\ 0) \tag{2}$$

$F\_leak\ 3$ represents the output of the *ShiftRows* in the first round of AES. The *ShiftRows* is a cyclic shift operation performed on different rows. Keeping the first line unchanged; the second line shifts one byte to the left; the third line shifts two bytes to the left; the fourth line shifts three bytes to the left. Thus the leakage function $V_{F\_leak\ 3}$ for $F\_leak\ 3$ as a point of attack is expressed as:

$$V_{F\_leak\ 3_i} = \begin{cases} V_{F\_leak\ 2_{i+0}} & i = 1, 5, 9, 13 \\ V_{F\_leak\ 2_{i+4}} & i = 2, 6, 10, 14 \\ V_{F\_leak\ 2_{i+8}} & i = 3, 7, 11, 15 \\ V_{F\_leak\ 2_{i+12}} & i = 4, 8, 12, 16 \end{cases} \tag{3}$$

where i denotes the ith byte, and $i + n = i + n - 16$ $(n = 0, 4, 8, 12)$ when $i + n > 16$.

$F\_leak\ 4$ represents the output of *MixColumns* in the first round of AES. *MixColumns* is achieved by multiplying matrices in a finite field GF(2^8) as follows, where $\alpha = V_{F\_leak\ 3}$ denotes the middle state of the *ShiftRows*'s output and $\beta = V_{F\_leak\ 4}$ denotes the middle state of the *MixColumns*'s output.

$$\begin{bmatrix}\beta_{1,r}\\ \beta_{2,r}\\ \beta_{3,r}\\ \beta_{4,r}\end{bmatrix} = \begin{bmatrix}02 & 03 & 01 & 01\\ 01 & 02 & 03 & 01\\ 01 & 01 & 02 & 03\\ 03 & 01 & 01 & 02\end{bmatrix} \begin{bmatrix}\alpha_{1,r}\\ \alpha_{2,r}\\ \alpha_{3,r}\\ \alpha_{4,r}\end{bmatrix} \tag{4}$$

In Equation (4), the first corner labels of α and β represent the rows of the 4×4 matrix and the second corner labels ($r \in 1, 2, 3, 4$) represent the columns of the matrix (α's left multiplication matrix is a fixed matrix of *MixColumns*). Therefore, when $5th$, $6th$, $7th$ and $8th$ subkeys are used as target subkeys, the leakage function $V_{F\_leak\ 4}$ is expressed as:

$$V_{F\_leak\ 4_i} = \begin{cases}[2 \times \alpha_5] \oplus [3 \times \alpha_6] \oplus \alpha_7 \oplus \alpha_8 & i = 5\\ \alpha_5 \oplus [2 \times \alpha_6] \oplus [3 \times \alpha_7] \oplus \alpha_8 & i = 6\\ \alpha_5 \oplus \alpha_6 \oplus [2 \times \alpha_7] \oplus [3 \times \alpha_8] & i = 7\\ [3 \times \alpha_5] \oplus \alpha_6 \oplus \alpha_7 \oplus [2 \times \alpha_8] & i = 8\end{cases} \tag{5}$$

In Equation (5), $a = V_{F\_leak\ 3}$ and i represents the ith byte. '$\times$' denotes a multiplication operation in a finite field and '$\oplus$' denotes the XOR operation.

In the last round of AES-128, the leakage function V_{L_leak} is calculated in a similar way to the leakage function V_{F_leak} in the first round. The leakage function for the last round V_{L_leak} is represented by Equation (6-10).

$$V_{L\_leak\ 1} = Ct \oplus Key\ 10 \tag{6}$$

$$V_{L\_leak\ 2_i} = \begin{cases}V_{L\_leak\ 1_{i+0}} & i = 1, 5, 9, 13\\ V_{L\_leak\ 1_{i+12}} & i = 2, 6, 10, 14\\ V_{L\_leak\ 1_{i+8}} & i = 3, 7, 11, 15\\ V_{L\_leak\ 1_{i+5}} & i = 4, 8, 12, 16\end{cases} \tag{7}$$

$$V_{L\_leak\ 3} = SBox^{-1}(V_{L\_leak\ 2}) \tag{8}$$

$$V_{L\_leak\ 4} = V_{L\_leak\ 3} \oplus Key\ 9 \tag{9}$$

$$V_{L\_leak\ 5_i} = \begin{cases}[E \times \gamma_5] \oplus [B \times \gamma_6] \oplus [D \times \gamma_7] \\ \quad \oplus [9 \times \gamma_8] \quad i = 5\\ [9 \times \gamma_5] \oplus [E \times \gamma_6] \oplus [B \times \gamma_7] \\ \quad \oplus [D \times \gamma_8] \quad i = 6\\ [D \times \gamma_5] \oplus [9 \times \gamma_6] \oplus [E \times \gamma_7] \\ \quad \oplus [B \times \gamma_8] \quad i = 7\\ [B \times \gamma_5] \oplus [D \times \gamma_6] \oplus [9 \times \gamma_7] \\ \quad \oplus [E \times \gamma_8] \quad i = 8\end{cases} \tag{10}$$

In Equation (6-10), Ct is the ciphertext. $Key\ 10$ and $9th$ represents the $10th$ and $9th$ round key, respectively. $SBox^{-1}$ denotes the inverse of SBox. i denotes the ith byte. In Equation (10), γ is used instead of $V_{L\_leak\ 4}$. 9, B, D, E are the values in the reverse column obfuscation in AES-128.

B. Label for Multi-input Model

The SCAs are usually divided analysis into non-profiled and profiled analysis, with the profiled analysis divided into two stages. The first stage is called profiling, in which a deep-learning model is trained to learn a leakage profile between traces and the secret. Afterwords, the second stage is called attack stage, in which the attacker uses the trained model to classify traces from victim device. To obtain a well-trained model, profiling traces are required to be labeled properly. As we mentioned before, there are three commonly used power models: HW, HD and ID. HW and HD models are reasonable estimations but suffer from the issue of class imbalance in practice owing to Bernoulli Distribution [23]. The value model (identity (ID) model) assumes the power consumed by the device is propotional to the data processed at the attack point. We use ID model in our experiment to distinguish different power traces.

There is a correspondence between the attack point and the leakage function (the leakage function are defined as label in deep learning), and we can build a network model for recovering the key according to each attack point. However, in the process of this attack, the connection between different attack points in the AES algorithm and the feature that each attack point is correlated with the same key are not taken into account. In order to combine information from multiple attack points, we propose a multi-input model. Because each attack point corresponds to a different leakage function (e.g. the leakage function when the output of the *AddRoundKey* is used as the attack point is different from the leakage function of the SBox's output), and because there is only one output in the multi-input model, the leakage function V_{F_leak} of multiple attack points need to be unified. The following describes the way to unify the leak functions of different attack points and the reason why multiple leakages can exist at one attack point.

The first round of AES is used as an example to investigate the relationship between leakage functions at different attack points. We use the leakage function of $V_{F\_leak\ 2}$ instead of the other leakage functions as label for the multi-input model. By replacing $V_{F\_leak\ 1}$ with $V_{F\_leak\ 2}$, the one-to-one non-linear transformation of SBox does not affect the classification of the network model (e.g. after replacing all the labels of cats with dogs and all the labels of dogs with pigs in image classification, results show that the accuracy of the network model training does not change). The *ShiftRows*'s leakage function simply shifts $V_{F\_leak\ 2}$ without changing it, so $V_{F\_leak\ 2}$ can be used instead of $V_{F\_leak\ 3}$. The leakage function $V_{F\_leak\ 4}$ for one subkey in *MixColumns* is obtained from four different subkeys of the $V_{F\_leak\ 2}$ by the XOR operation, and $V_{F\_leak\ 2}$ is used as part of the *MixColumns* leakage function, so $V_{F\_leak\ 2}$ can be used instead of $V_{F\_leak\ 4}$ (e.g. when the *AddRoundKey* is used as a specific attack point, the Key can be used as the model's label). The leakage function for multiple attack points is unified as $V_{F\_leak\ 2}$. This is the first step in building a multi-input model, which is described below.

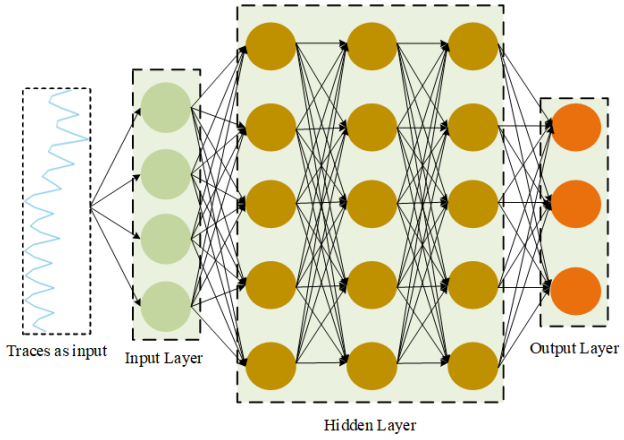


Fig. 2. Single-input model structure.

C. Construction of Multi-input Model

The basic architecture used in this work is a CNN with multiple input layers, as shown in Fig.3. The multiple inputs are merged and connected to a *Convolutional layer* consisting of 32 neurons. Then the extracted features are expanded by a *Flatten layer* after passing through a three-strides *MaxPooling layer*. Afterwards, two *Dense layers* are connected to the *Flatten layer* and each dense layer contains 128 neurons. The *Output layer* is also a dense layer but with 256 neurons for prediction and the activation function is set to *Softmax*. The *Convolutional layer* and *Dense layers* are activated by function with Rectified Linear Units (*ReLU*). The single input model does not contain a *Merge layer* and consists of a single input layer connected to a *Convolutional layer*. The rest of the model structure is the same as the multi-input model structure, as shown in Fig.2.

Merge layer is an important aspect when building multi-input models. It can be used to combine two different neural networks which are trained for the same task but on different datasets. Two fusion techniques are commonly used in existing works, one is called early fusion and another one is late fusion [24], [25]. Early fusion merges layers of different neural networks at an early stage while late fusion merges layer lately. Early fusion is the combination of multiple inputs which are then connected to the first layer of the DNN. In the late fusion architecture, features are first extracted from the input data of individual channels. The specific information of the channels is eventually merged and processed in further network model layers responsible for the classification based on the extracted features.

We find that the late-fusion model is less accurate than the early-fusion model, as shown in VI. Therefore, in this paper we only conduct experiments for the early fusion network model.

There are different types of methods to merge layers within DNN architectures, which are listed below:

- `Add`: returns the element-wise sum of two inputs
- `Subtract`: returns element-wise subtracts two inputs
- `Multiply`: returns element-wise multiplication of inputs
- `Average`: returns element-wise average of the inputs
- `Maximum`: returns element-wise maximum of the inputs
- `Minimum`: returns element-wise minimum of the inputs
- `Concatenate`: returns concatenation of the inputs

D. Method for Multi-input Model

The following steps are required to complete the application of the multiple input model to the multiple leakage intervals. In the first step, a suitable attack point is selected and the intermediate value function for that attack point is derived by an energy model. In the second step, the index of the leaky intervals on the traces for that intermediate value function is found by finding the leaky interval. In the third step, the leakage intervals are sliced and a traditional DL model is trained for each leakage interval. In the fourth step, the two leakage intervals corresponding to the models with the strongest classification accuracy on the testing sets are selected and used to explore which fusion method is optimal for improving the classification accuracy of the multi-input models. In the fifth step, all the leaked intervals are used for training the multi-input network model using this fusion method to obtain the optimal multi-input network model.

III. Experimental Setup

In this section, we first introduce the datasets and the evaluation metrics we used for the experiments. Afterwards, we test the proposed multi-input model on traces captured from a CW308T-STM32F3 board. Next, we further validate the performance of our model on the STM32 implementation of the 32bit AES-128 dataset and the AES_GPU [26] public dataset.

A. Datasets

In our experiments, we use three datasets in total. Power traces in the first dataset are captured from a CW308T-STM32F3 board implementation of TinyAES-128. The board contains an Arm Cortex M4 microcontroller. The mode of operation is set to Electronic CodeBook (ECB) mode. The training set involves $50K$ traces represening the first round of AES-128 and $50K$ traces for the last round. The testing set contains $10K$ traces with random plaintexts and fixed keys for both the first and last rounds respectively. Each power trace contains $4,000$ sampling points as shown in Fig.4(a), (b).

The second dataset was captured with the same equipment as the first dataset, implementing AES-128 for 32bit parallel processing. The training set involves $50K$ traces and the testing set involves $10K$ traces, generated from random plaintext and fixed keys. Each power trace contains $1,000$ sampling points as shown in Fig.4(c).

The third dataset is an NVIDIA GeForce GT620 graphics card (GPU) connected to the host with a PCIe bus. The AES parallel implementation (32 threads in a warp) and trace acquisition details are stated in [26]. There are $34,511$ traces for profiling and $5,000$ traces for the attack. We call

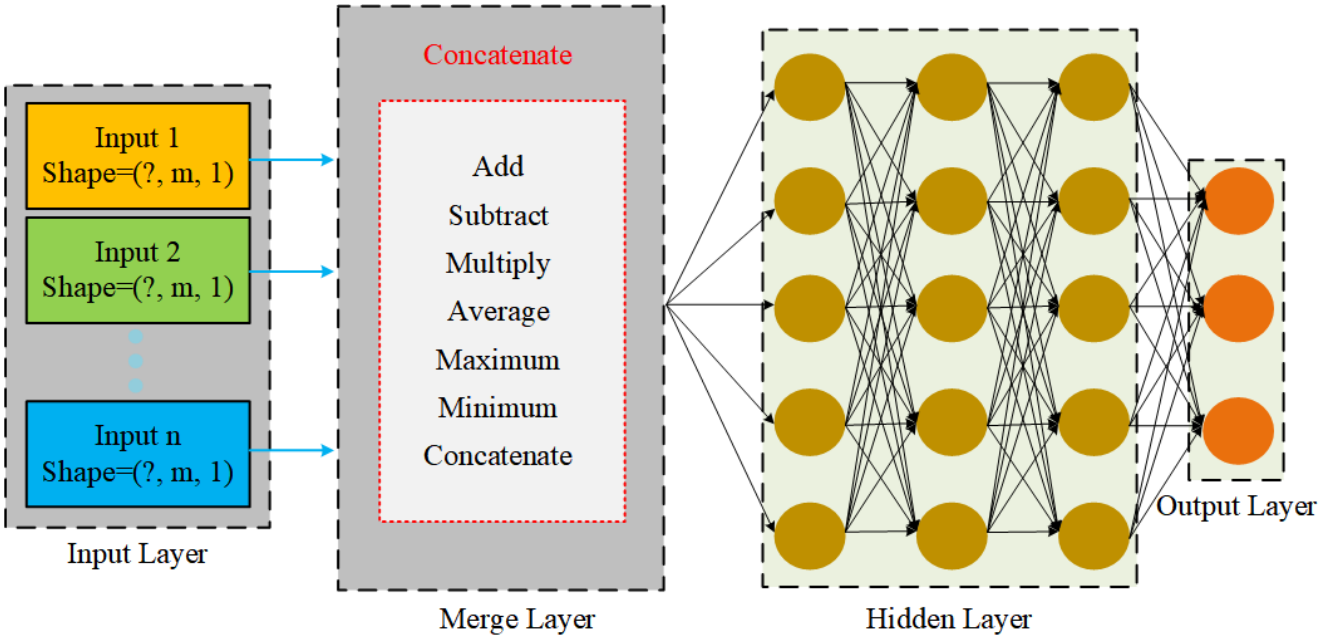


Fig. 3. Multi-input model structure.

this homemade dataset AES_GPU in brief. Each power trace contains $15,001$ sampling points as shown in Fig.4(d).

B. Evaluation Metrics

The first metric used in our experiments to evaluate how the trained model performs on the testing set is the classification accuracy or sometimes called attack accuracy. The attack accuracy is defined as the fraction of correct predictions when using the trained model to classify traces from the testing set. The formula of the attack accuracy is shown below:

$$acc(X_{attack}) = \frac{|\{X_{correct} \in X_{attack}\}|}{|X_{attack}|} \tag{11}$$

In Equation (11), X_{attack} denotes the testing dataset. $X_{correct}$ is the set of power traces when the guessed keys are all equal to the correct key.

However, when traces are noisy [27], it might be difficult for the model to predict the key with a single traces. In that case, partial guessing entropy (PGE) becomes a more suitable evaluation criterion. PGE indicates the mean rank of the real subkey sorted by the predicted probabilities of all possible subkeys. During the attack stage, we use the trained model to classify traces from the testing set and obtain the probabilities of different keys for each trace. For trace $x_i \in X_{attack}$, the obtained probability matrix is denoted as $P_i = [p_{i,1}, p_{i,2}, ..., p_{i,255}]$, where $p_{i,j}$ in P_i is the predicted probability of k=j for trace x_i. Where P_i is the correct Key Rank, which is usually used as an evaluation criterion for datasets with better signal-to-noise ratios, as the number of traces used to recover the correct key for datasets with higher signal-to-noise ratios is usually in the single digits, and using the Key Rank provides a more intuitive evaluation of the results. The lower the number of traces in the Key Rank, the better the model.

Afterwards, we apply an element-wise multiplication for all P_i to obtain a cumulative probability:

$$\mathbf{P} = \prod_{i=1}^{m} P_i = [\mathbf{P}_0, \mathbf{P}_1, ..., \mathbf{P}_{255}] \tag{12}$$

where m is the number of traces we used for classification. Then, PGE can be represented as the averaged rank of real key k^* sorted by $\mathbf{P}$.

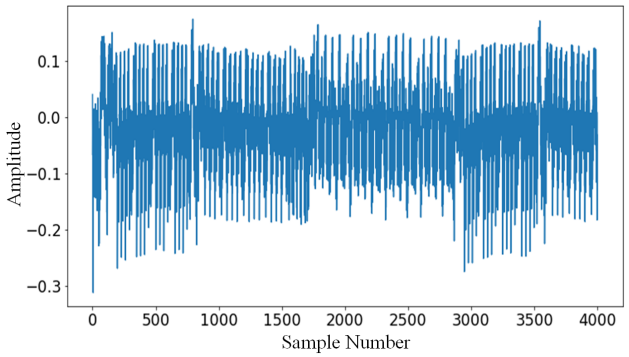
IV. Experimental Result

A. TinyAES-128 implementation on a STM32F3

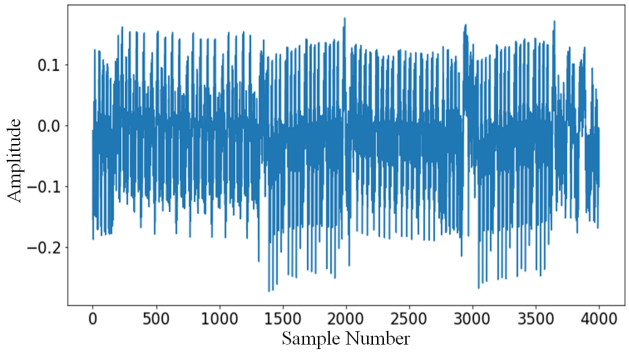
In the first experiment, the target is a STM32F3 implementation of TinyAES-128. During network training, we used the *Adam* optimizer with a learning rate of 0.0005. The mini-batch size is 256 and the maximum iteration epoch is 500.

1) First Round of AES: We use the ρ-test [28] as the leakage detection method to find the Point of Interest (POI) of each subkey for the attack point. The POI of the first round of AES is shown in Fig.5. In our experiments, we randomly choose the 5th subkey as an example for illustration and others will be the same.

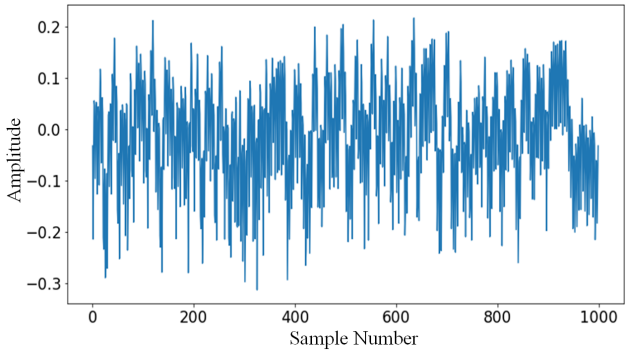
Fig.5 shows intervals divided by red lines as A, B, C, D and E, representing the *AddRoundKey* operation before the first round of AES, and the *SubBytes*, *ShiftRows*, *MixColumns* and *AddRoundKey* operation of the first round. From Fig.5, we can see that interval D still leaks information about the attack point. However, when it comes to interval E, traces does not contain any leakage. Since traces in interval E represent *AddRoundKey* operation of the first round encryption of AES-128 and the attack point related input for this procedure is the



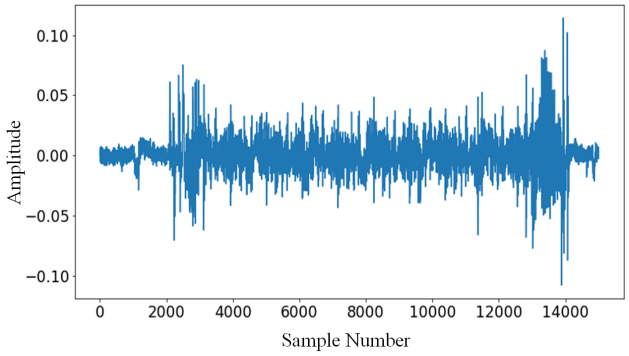
(a)



(b)



(c)



(d)

Fig. 4. (a) Waveform of the first round traces of the STM32F3 implementation of TintAES-128; (b) Waveform of the last round traces of the STM32F3 implementation of TintAES-128; (c) Waveform of the last round traces of the STM32F3 implementation of 32bit AES-128; (d) Waveform of the last round traces of the NVIDIA GeForce GT620 graphics card implementation of TintAES-128.

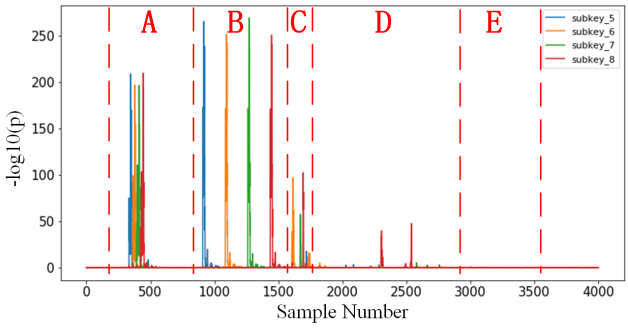


Fig. 5. ρ-test results of the 5th, 6th, 7th and 8th subkey for the first round traces of a STM32F3 implementation of TintAES-128.

TABLE I
LEAKAGE INTERVALS OF THE 5TH SUBKEY WHEN USING THE SBOX OUTPUT IN THE FIRST ROUND OF AES AS THE ATTACK POINT AND CLASSIFICATION ACCURACIES OF SINGLE-INPUT MODELS TRAINED ON THE CORRESPONDING LEAKAGE INTERVAL.

Input section	POI interval	Accuracy
AddRoundKey (A)	[315:375]	11.40%
SubBytes (B)	[890:950]	48.27%
ShiftRows (C)	[1690:1750]	0.85%
MixColumns (D)	[2037:2097]	0.47%

output of *MixColumns* operation, this verifies the statement that *MixColumns* procedure is side-channel resistant [22].

Next, we first train 4 conventional single-input Convolutional Neural Network (CNN) models (the model structure is shown in Fig.2) on traces with different POI intervals separately. The ρ-test of the 5th subkey is illustrated by Fig.5, divided by the red line dividing 4 leakage intervals. For each CNN model, we train it on traces with a specific leakage interval. Four leakage intervals for these 4 models are listed in Table.I. Afterwards, we use these four models to classify testing traces separately and the classification results are also shown in Table.I.

From Table.I, we can find that all the leakage intervals can contribute to bulid a leakage profile between traces and the selected attack point. We consider the model to be effective for the task when the classification accuracy on the testing sets is higher than $1/256 \approx 0.39\%$.

Afterwards, we train and test the propased multi-input CNN models for different combination of leakage intervals. In Table.I, we can find that leakage interval B achieves the best classification result. So our 1-input models are trained on traces with leakage interval B. For the 2-input model, we use interval A and B as the two inputs since interval A has the second best classification accuracy. By following this rule, our 3-input model is made by using interval A, B and C as the inputs. The 4-input model makes use of all listed leakage intervals.

Since there are multiple fusion methods and not all of them have an improvement in model classification accuracy, we used the leakage interval A and B of the 5th subkey to explore which fusion method is more effective in improving

TABLE II

CLASSIFICATION ACCURACY OF THE 2-INPUT (A&B) MODEL ON THE TESTING SETS AND THE NUMBER OF TRACES WITH KEY RANK ¡ 5 ON THE TESTING SETS (A TOTAL OF 10K TRACES WERE USED AS THE TESTING SET)

Fusion Methods	Add	Subtract	Multiply	Average	Maximum	Minimum	Concatenate (axis=1)	Concatenate (axis=2)
Test Accuracy	44.26%	57.10%	36.61%	44.01%	53.06%	62.54%	81.60%	82.67%
Key Rank <5	7894	8670	7067	7840	8546	9012	9632	9880

TABLE III

LEAKAGE INTERVALS FOR THE 5TH ,6TH ,7TH AND 8TH SUBKEYS FOR THE LAST ROUND TRACES OF A STM32F3 IMPLEMENTATION OF TINTAES-128.

Leakage Interval	5th Subkey	6th Subkey	7th Subkey	8th Subkey
AddRoundKey (A),	[315:375]	[345:405]	[375:435]	[410:470]
SubBytes (B),	[890:950]	[1070:1130]	[1240:1300]	[1415:1475]
ShiftRows (C),	[1690:1750]	[1585:1645]	[1645:1705]	[1670:1730]
MixColumns (D)	[2037:2097]	[1770:1830]	[2595:2655]	[2310:2370]

TABLE IV

RESULTS OF CLASSIFICATION ACCURACIES ON TESTING SETS FOR SINGLE-INPUT AND MULTI-INPUT MODELS TRAINED USING 5TH, 6TH, 7TH AND 8TH SUBKEYS FOR THE LAST ROUND TRACES OF A STM32F3 IMPLEMENTATION OF TINTAES-128.

	Leakage Interval	5th Subkey	6th Subkey	7th Subkey	8th Subkey
1-input Model	A	11.40%	21.01%	23.10%	18.04%
	B	48.27%	47.20%	53.77%	48.86%
	C	0.85%	7.63%	1.93%	4.04%
	D	0.47%	2.85%	1.32%	2.82%
	A&B&C&D	79.21%	84.57%	77.62%	80.82%
2-input Model	A&B (axis=1)	81.60%	83.27%	86.39%	81.11%
	A&B (axis=2)	82.67%	83.73%	85.62%	82.06%
3-input Model	A&B&C (axis=1)	78.14%	88.29%	86.69%	85.30%
	A&B&C (axis=2)	82.51%	90.15%	86.71%	85.41%
4-input Model	A&B&C&D (axis=1)	69.79%	83.79%	75.23%	69.48%
	A&B&C&D (axis=2)	82.87%	91.69%	87.38%	85.64%

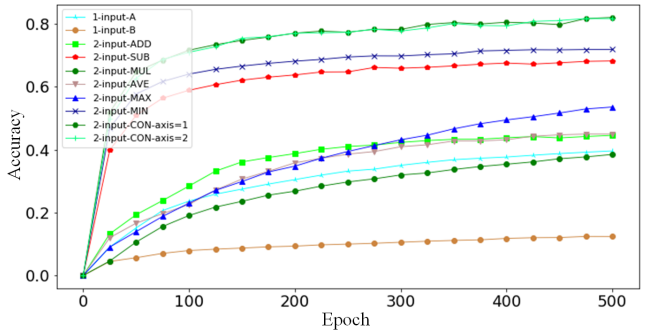


Fig. 6. Classification accuracy of 1-input and 2-input models on the validation set of the 5th subkey.

the classification accuracy of the multi-input network model. Fig.6 shows the classification accuracy of the 1-input model and the 2-input model (model structure shown in Fig.3), on the validation set. Table.II shows the classification accuracy of the 2-input model on the test set and the Key Rank of the different fusion methods. As can be seen from Table.II, *Concatenate Layer* as the optimal fusion method, in the later experiments we focus on using *Concatenate Layer* to merge neural networks trained on different leakage intervals.

For the *Concatenate Layer*, there are two common settings for fusing the inputs: axis = 1 for row-wise concatenation and axis = 2 for column-wise concatenation. Note that axis = 0 is the batch axis.

Afterwards, we test the trained 1-input and multi-input models on trace in the testing set. Table.IV shows the classification accuracies of these models to recover 5th, 6th, 7th and 8th subkeys of AES.

From Table.IV, we can find that the multi-input models can always achieve a higher classification accuracies than the single-input models. The last row of the 1-input in Table.IV shows the classification accuracy on the testing sets for models trained using the full leakage intervals containing the target subkeys. This indicates that by utilizing multiple leakage intervals of traces, it is capable of further improving the attack efficiency of deep-learning models in side-channel attacks' context. However, we can also see that leakage intervals A and B, which represent *AddRoundKey* and *SubBytes* operations separately, contribute the most to the multi-input model. For leakage interval C and D, a well-trained model can also use them to a fine-tune the classification accuracy. For the setting of the Concatenate Layer, it seems better to use the column-wise approach (axis = 2), as we can see from Table.IV. Next, we show experiments on the last round of AES for the proposed multi-input models.

2) Last Round of AES: The POI for the last round of AES for the STM32F3 implementation of TinyAES-128 is shown in Fig.7. In this section, the experiments are as the same as in

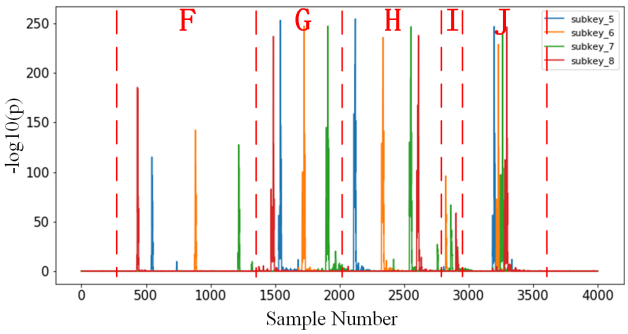


Fig. 7. ρ-test results of the 5th, 6th, 7th and 8th subkey for the last round traces of a STM32F3 implementation of TintAES-128.

the first-round experiments. So we keep using 5th, 6th, 7th and 8th subkeys as the targets to show the classification results.

In this experiment, we also use the ρ-test as the leakage detection approach. We plot the ρ-test results of all subkeys for the last round of TinyAES-128 in Fig.7. The attack point for the ρ-test of the traces is set to the SBox input of the last round of AES. In Fig.7, we can divide traces to leakage intervals as shown in Fig.7. Notice that compared to the ρ-test results of the first-round traces, the last-round traces contain two more leakage intervals which are denoted as interval F and G, for the 9th round encryption of AES. This indicates that these two operations in the 9th round of AES also contain information related to the attack point. For traces representing the last round of AES, there are only three leakage intervals: H, I, J. This is because that the last round does note *MixColumns*.

Next, we train 5 conventional single-input CNN models (the model strcture in shown in Fig.2) on the last-round traces with different leakage intervals separately. Afterwards, we use these 5 models to classify testing traces separately and the classification results are shown in Table.VI.

Because the model trained on leakage interval F cannot achieve a classification accuracy larger than 0.39% on the testing set, this interval will will not be involved in the training of the multi-input models. The conclusion that the multi-input models a more efficient attack in the presence of multiple leaks in traces is verified in the last round of AES.

From Table.IV and Table.VI, it is easy to draw the following conclusions: For models trained using a single leaky interval with a classification accuracy below 0.39%, the addition of that leaky interval to the multi-input model does not improve the classification accuracy of the multi-input model; For the setting of the *Concatenate layer*, it seems better to use the column-wise approach (axis = 2); The classification accuracy using the multi-input model in the last round of AES-128 is higher than using the multi-input model in the first round of AES-128 because the leakage intervals exist in the 9th and last round of AES-128 when the input to the SBox in the last round of AES-128 is the attack point.

From Table.IV and Table.VI, it can be concluded that the classification accuracies of the 4-input model is just lower than that of the 2-input model when using the concatenation axis=1 fusion method, and according to the experiments we have done, the increase of the network training parameters is not very effective in improving the classification accuracies. It can be noted that the classification accuracies of the 4-input model in Tables 4 and 6 of the paper decreases when using the concatenation axis=1 fusion method compared to the 3-input model, and does not decrease when using the concatenation axis=2 fusion method for the 4-input model. The conclusion is that when using the concatenation axis=1 fusion method, multiple leaked intervals are connected to form a longer trace, but each leaked interval contributes differently to the recovery key (i.e. the classification accuracy of the model trained using that leaked interval), and this connection leads to a decrease in classification accuracies when some of the leaked intervals contribute too little. For the concatenation axis=2 fusion method, each leaked interval is concatenated on a channel, and each leaked interval belongs to a different dimension, which does not degrade the classification accuracies when passed into the neural network for training, and as long as the leaked interval contributes to the recovery key, then the classification accuracy is reduced using The final classification accuracies of the multi-input model is improved by adding the leaked interval using the concatenation axis=2 fusion method. The detailed parameters of the models are shown in VII.

Next, we show the results of implementing 32bit AES-128 in STM32 for the proposed multi-input model.

B. 32bit AES-128 implementation on a STM32F3 (AES_32bit)

We aim at the leakage operation of the last round 5th byte register writing: $v_5 = SBox^{-1}[c_5 \oplus k^*]$, where c_5 is the 5th ciphertext byte. We call this homemade dataset AES_32bit in brief. During network training, we used the *Adam* optimizer with a learning rate of 0.0005. The mini-batch size is 256 and the maximum iteration epoch is 500.

We use the ρ-test as a leakage detection method to find the leakage intervals corresponding to v_5 in the traces, as shown in Fig.8. In Fig.8, the leakage intervals are divided into A, B and C using the green lines. The leakage interval A is indexed as [320: 390], the leakage interval B is indexed as [470: 540] and the leakage interval C is indexed as [760: 830]. Since it has been shown in IV-A that the *Concatenate layer* as a *Merge layer* can most effectively improve the classification accuracy of the multi-input model, in the next experiments, only the single-input model with the highest classification accuracy and the multi-input model using the *Concatenate layer* are shown. Fig.9, shows the classification accuracy on the validation set for the single-input model and the multi-input model during training. Fig.10 shows the Key Rank comparison between the single-input, multi-input deep learning (DL) model and the single-input, multi-input template attack (TA) on 10K test set traces.

Because of the STM32 implementation of 32bit AES-128, *SubBytes* and *ShiftRows* operations will not leak on the traces, the multi-input model only improves the classification accuracy by 19.65% over the single-input model on the test set in the AES_32bit dataset.

TABLE V

LEAKAGE INTERVALS FOR THE 5TH ,6TH ,7TH AND 8TH SUBKEYS FOR THE LAST ROUND TRACES OF A STM32F3 IMPLEMENTATION OF TINTAES-128.

Leakage Interval	5th Subkey	6th Subkey	7th Subkey	8th Subkey
AddRoundKey (J),	[3170:3230]	[3200:3260]	[3230:3290]	[3260:3320]
ShiftRows (I),	[2890:2950]	[2800:2860]	[2835:2895]	[2880:2940]
SubBytes (H),	[2090:2150]	[2305:2365]	[2520:2580]	[2580:2640]
AddRoundKey (G),	[1510:1570]	[1695:1755	[1875:1935]	[1450:1510]
MixColumns (F)	[520:580]	[855:915]	[1190:1250]	[410:470]

TABLE VI

RESULTS OF CLASSIFICATION ACCURACIES ON TESTING SETS FOR SINGLE-INPUT AND MULTI-INPUT MODELS TRAINED USING 5TH, 6TH, 7TH AND 8TH SUBKEYS FOR THE LAST ROUND TRACES OF A STM32F3 IMPLEMENTATION OF TINTAES-128.

	Leakage Interval	5th Subkey	6th Subkey	7th Subkey	8th Subkey
1-input Model	J	14.73%	13.71%	23.22%	15.41%
	I	0.39%	2.58%	2.48%	2.38%
	H	45.32%	42.04%	48.59%	45.51%
	G	14.27%	22.45%	10.58%	18.64%
	F	0.37%	0.37%	0.33%	0.38%
	J&I&H&G&F	84.76%	89.66%	87.21%	85.71%
2-input Model	H&J (axis=1)	80.41%	88.70%	74.80%	71.49%
	H&J (axis=2)	82.20%	88.97%	76.16%	83.10%
3-input Model	H&J&G (axis=1)	94.71%	91.63%	92.90%	88.83%
	H&J&G (axis=2)	95.17%	96.49%	94.75%	96.49%
4-input Model	H&J&G&I (axis=1)	87.96%	94.02%	93.77%	89.47%
	H&J&G&I (axis=2)	94.27%	97.38%	95.10%	96.57%

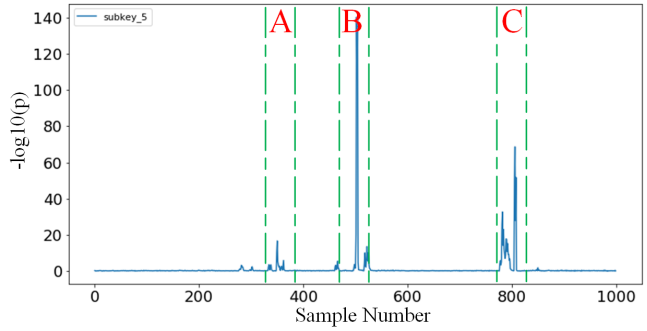


Fig. 8. ρ-test results for v_5 (AES_32bit).

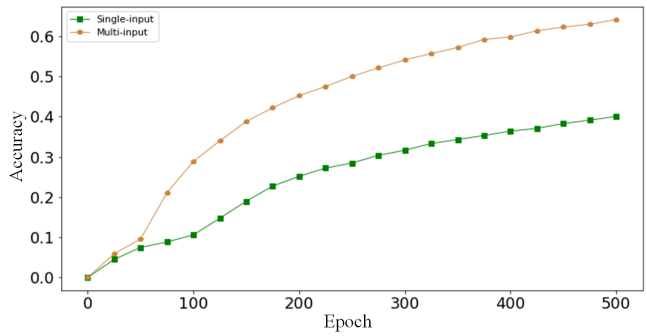


Fig. 9. Classification accuracy of single- and multi-input models on validation sets (AES_32bit).

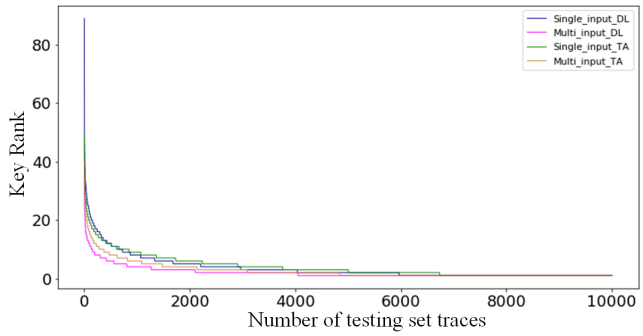


Fig. 10. Comparison of DLSCA and TA in a testing set with 10K traces Key Rank (AES_32bit).

Finally, the results for the single-input, multi-input DL model and the single-input, multi-input TA are shown in Table.VII. It can be noted that the classification accuracy of the multi-input model on the validation set is 1.75% higher than that of the single-input model, due to the fact that a model combining multiple inputs can learn more information related to the intermediate values in the data. Although multiple inputs are introduced into the template attack, they can effectively increase the efficiency of the template attack. However, in Table.VII, the multi-input deep learning model has 1399 more traces with Key Rank¡5 on the testing set than the multi-input template attack. This indicates that the multi-input deep learning model is currently a more reasonable solution to the problem of multiple leakage intervals.

Next, we show experiments on the AES_GPU datasets for the proposed multi-input models.

TABLE VII
NUMBER OF TRACES WITH KEY RANK ¡ 5 ON THE TESTING SET (10K) FOR SINGLE-INPUT, MULTI-INPUT DL MODELS AND SINGLE-INPUT, MULTI-INPUT TA (AES_32BIT).

Methods	Single-DL	Multi-DL	Single-TA	Multi-TA
Key Rank <5	7793	9192	7094	8520

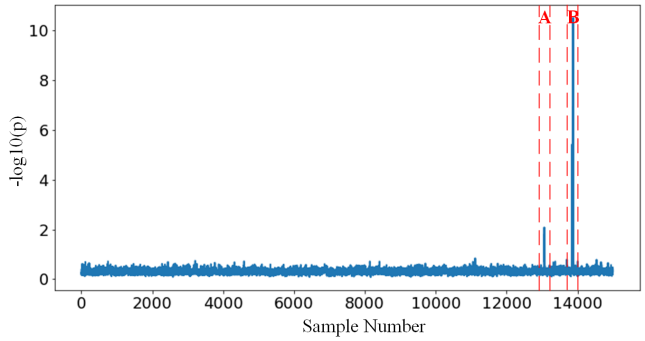


Fig. 11. ρ-test results for v_{16} (AES_GPU).

C. AES_GPU

We aim at the leakage operation of the last round 16th byte register writing: $v_{16} = SBox^{-1}[c_{16} \oplus k^*]$ where c_{16} is the 16th ciphertext byte. During network training, we used the *Adam* optimizer with a learning rate of 0.0001. The mini-batch size is 256 and the maximum iteration epoch is 500.

We use the ρ-test as a leakage detection method to find the leakage intervals corresponding to v_{16} in the traces, as shown in Fig.11. For the intermediate value v_{16}, there are two discontinuous leakage intervals on the AES_GPU dataset, which we denote as leakage intervals A and B. The index of leakage interval A is [13000: 13100] and the index of leakage interval B is [13800: 13900]. Similarly, we show only the single-input and multi-input models with the highest classification accuracy on the validation set. The classification accuracy of the single-input and multi-input models on the validation set is shown in Fig.12. To further investigate the effects of single and multiple inputs on DLSCA and TA, we investigated the number of traces on the AES_GPU dataset for single-input and multiple-input DL models and single-input and multiple-input TA on the testing set for Key Rank ¡ 5 respectively, and the results are shown in Fig.13.

Finally, the number of traces required to recover the target subkey (PGE) on the AES_GPU dataset is compared between this work and other researches and the results are shown in Table.VIII. The multi-input model requires only 14 traces in the AES_GPU dataset to recover the target subkey. This is 31 fewer traces than the most current state-of-the-art result for this dataset (CDAE proposed by Yang et al.).

V. CONCLUSION AND FUTURE WORK

In this paper, we propose a multi-input deep-learning model for side-channel attacks, which is dedicated for the case where multiple leakage intervals exist in traces. By utilizing these

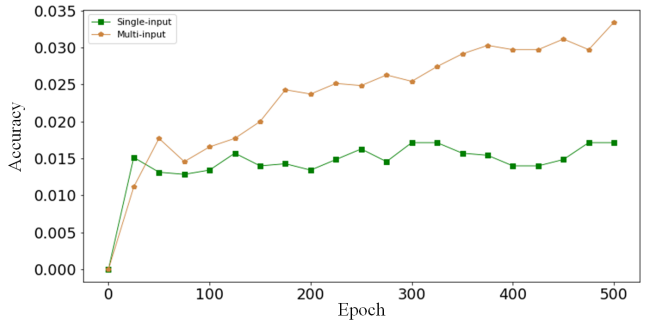


Fig. 12. Classification accuracy of single- and multi-input models on validation sets (AES_GPU).

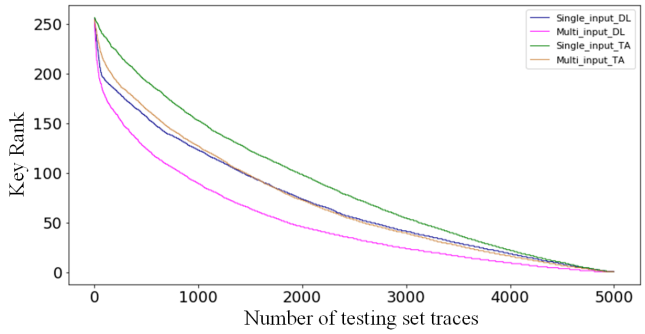


Fig. 13. Comparison of DLSCA and TA in a testing set with 5K traces Key Rank (AES_GPU).

leakages as separate inputs instead of using the entire trace for profiling, the trained model can focus more on these leakages. One well-known publicly available dataset and traces captured from a STM32F3 implementation of AES are used in our experiments. We show that the proposed multi-input model achieves a 2-fold improvement over the previous single-input attacks. Besides, we further compare different fusion layers for connecting leakage intervals. The result shows that concatenating leakage intervals in parallel outperforms other approaches.

Future work includes testing the proposed multi-input model on implementations of other cryptographic algorithms and mounting similar attacks on devices supporting AES with other countermeasures. Besides, we plan to further investigate the multi-leakage phenomena by training new models on other attack points. Certainly, the most important future work should be designing countermeasures to against deep-learning based side-channel attacks.

TABLE VIII
COMPARISON OF PGE RESULTS ON AES_GPU DATASET.

Methods	Accuracy	DL-PGE	TA-PGE
Original [26]	-	-	80
CDAE [27]	-	-	45
Single-input	1.64%	37	49
Multi-input	3.68%	14	23

VI. Appendix A. Comparison of the results of the early- and late- fusion models

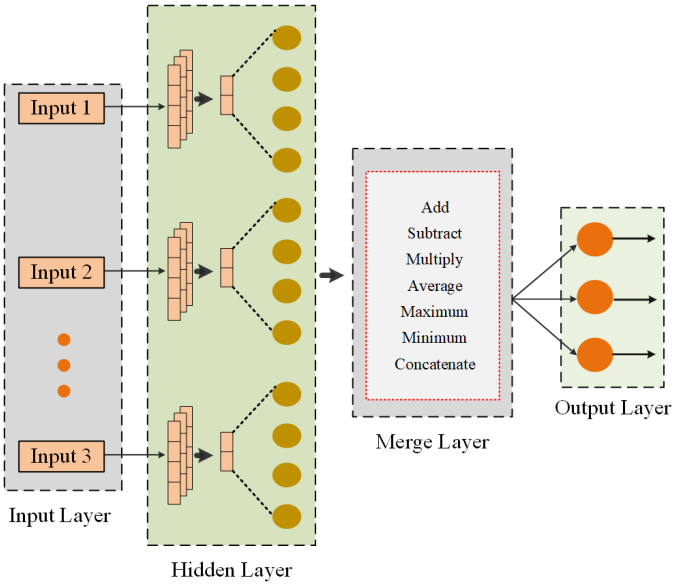


Fig. 14. Late-fusion multi-input model structure.

TABLE IX
Comparison of classification accuracies between early- and late-fusion models for the 2-input (leakage intervals A&B) models on the testing sets of the first round of the STM32 implementation of the AES-128 dataset.

	Early-fusion	Late-fusion
Add	44.26%	36.81%
Subtract	57.10%	46.88%
Multiply	36.61%	39.81%
Average	44.01%	39.21%
Maximum	53.06%	42.73%
Minimum	62.54%	51.07%
Concatenate	82.67%	47.20%

VII. Appendix B. Model structure and model parameters (STM32F3 implementation of AES-128 dataset)

TABLE X
1-input model structure and parameters.

Layer Type	Output Shape	Parameter #
Input Layer	(None, 60, 1)	0
Conv1D	(None, 54, 32)	256
Max Pooling	(None,27, 32)	0
Flatten	(None, 864)	0
Dense 1	(None, 128)	110720
Dense 2	(None, 128)	16512
Output (Dense)	(None, 256)	33024
Total Parameters: 160,512		

TABLE XI
2-input model structure and parameters (Add). Subtract, Multiply, Average, Maximum and Minimum have the same parameters as Add.

Layer Type	Output Shape	Parameter #
Input Layer 1	(None, 60, 1)	0
Input Layer 2	(None, 60, 1)	0
Merge Layer (Add)	(None, 60, 1)	0
Conv1D	(None, 54, 32)	256
Max Pooling	(None,27, 32)	0
Flatten	(None, 864)	0
Dense 1	(None, 128)	110720
Dense 2	(None, 128)	16512
Output (Dense)	(None, 256)	33024
Total Parameters: 160,512		

TABLE XII
2-input model structure and parameters (Concatenate axis=1).

Layer Type	Output Shape	Parameter #
Input Layer 1	(None, 60, 1)	0
Input Layer 2	(None, 60, 1)	0
Merge Layer (con axis=1)	(None, 120, 1)	0
Conv1D	(None, 144, 32)	256
Max Pooling	(None, 57, 32)	0
Flatten	(None, 1824)	0
Dense 1	(None, 128)	233600
Dense 2	(None, 128)	16512
Output (Dense)	(None, 256)	33024
Total Parameters: 283,392		

References

[1] P. C. Kocher, "Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems," in *Annual International Cryptology Conference*. Springer, 1996, pp. 104–113.

[2] A. Shamir and E. Tromer, "Acoustic cryptanalysis," 2004.

[3] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Annual international cryptology conference*. Springer, 1999, pp. 388–397.

[4] K. Gandolfi, C. Mourtel, and F. Olivier, "Electromagnetic analysis: Concrete results," in *International workshop on cryptographic hardware and embedded systems*. Springer, 2001, pp. 251–261.

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

[6] A. Schlösser, D. Nedospasov, J. Krämer, S. Orlic, and J.-P. Seifert, "Simple photonic emission analysis of aes," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2012, pp. 41–57.

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

[8] Z. Martinasek and V. Zeman, "Innovative method of the power analysis," *Radioengineering*, vol. 22, no. 2, pp. 586–594, 2013.

[9] T. Wilmshurst, *Designing embedded systems with PIC microcontrollers: principles and applications*. Elsevier, 2006.

[10] H. Wang, M. Brisfors and S. Forsmark, *et al.* "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.

[11] D. Das, A. Golder and J. Danial, *et al.* "X-deepsca: Cross-device deep learning side channel attack," in *Proceedings of the 56th Annual Design Automation Conference 2019*, 2019, pp. 1–6.

TABLE XIII
2-INPUT MODEL STRUCTURE AND PARAMETERS (CONCATENATE AXIS=2).

Layer Type	Output Shape	Parameter #
Input Layer 1	(None, 60, 1)	0
Input Layer 2	(None, 60, 1)	0
Merge Layer (con axis=2)	(None, 60, 2)	0
Conv1D	(None, 54, 32)	480
Max Pooling	(None, 27, 32)	0
Flatten	(None, 864)	0
Dense 1	(None, 128)	110720
Dense 2	(None, 128)	16512
Output (Dense)	(None, 256)	33024
Total Parameters: 160,736		

[12] H. Wang, S. Forsmark and M. Brisfors, *et al.* "Multi-source training deep-learning side-channel attacks," in *2020 IEEE 50th International Symposium on Multiple-Valued Logic (ISMVL)*. IEEE, 2020, pp. 58–63.

[13] T. Kubota, K. Yoshida and M. Shiozaki, *et al.* "Deep learning side-channel attack against hardware implementations of aes," *Microprocessors and Microsystems*, p. 103383, 2021.

[14] 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.

[15] J. Kim, S. Picek and A. Heuser, *et al.* "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.

[16] 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.

[17] E. Cagli, C. Dumas, and E. Prouff, "Convolutional neural networks with data augmentation against jitter-based countermeasures," in *International Conference on Cryptographic Hardware and Embedded Systems*. Springer, 2017, pp. 45–68.

[18] D. Moonen, "Little or large?: The effects of network size on ai explainability in side-channel attacks," 2020.

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

[20] L. Zhang, X. Xing and J. Fan, *et al.* "Multi-label deep learning based side channel attack," in *2019 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*. IEEE, 2019, pp. 1–6.

[21] J. Benesty, J. Chen and Y. Huang, *et al.* "Pearson correlation coefficient," in *Noise reduction in speech processing*. Springer, 2009, pp. 1–4.

[22] J. Daemen and V. Rijmen, "Reijndael: The advanced encryption standard." *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, vol. 26, no. 3, pp. 137–139, 2001.

[23] S. Picek, A. Heuser and A. Jovic, *et al.* "The curse of class imbalance and conflicting metrics with machine learning for side-channel evaluations," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2019, no. 1, pp. 1–29, 2019.

[24] D. Feng, C. Haase-Schuetz and L. Rosenbaum, *et al.* "Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 3, pp. 1341-1360, 2020.

[25] J. Xu and H. M. Heys, "Using deep learning to combine static and dynamic power analyses of cryptographic circuits," *International Journal of Circuit Theory and Applications*, vol. 47, no. 6, pp. 971–990, 2019.

[26] Y. Gao, H. Zhang and W. Cheng, *et al.* "Electro-magnetic analysis of gpu-based aes implementation," in *Proceedings of the 55th Annual Design Automation Conference*, 2018, pp. 1–6.

[27] G. Yang, H. Li and J. Ming, *et al.* "Cdae: Towards empowering denoising in side-channel analysis," in *International Conference on Information and Communications Security*. Springer, 2019, pp. 269–286.

[28] 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.