Side-channel Analysis using Deep Learning on Hardware Trojans

J. Kokila
National Institute of Technology, Tiruchirappalli, Tamil Nadu, INDIA
E-mail: jk.cse09@gmail.com

S. Chithra
National Institute of Technology, Tiruchirappalli, Tamil Nadu, INDIA
E-mail: chithratheertham@gmail.com

N. Ramasubramanian
National Institute of Technology, Tiruchirappalli, Tamil Nadu, INDIA
E-mail: nrs@nitt.edu

Abstract. A power side-channel analysis is proposed to find the correct key in hardware trojan affected AES by evaluating correlation between every sub-key guesses. A deep learning method for side-channel analysis (SCA) is proposed, which consists of two phases-characterization and attack. Multilayer perceptron (MLP) and convolutional neural networks (CNN) are used as the models to determine the SCA and their performances are evaluated. Two kinds of desynchronization were used, where maximum possible value with which the trace can be shifted was 0 in the first and 100 in the second. SCA is performed in these desynchronized traces also.

1. Introduction
Side channels are those values whose analysis will help us to find some secret information. Some side-channel analysis procedures require the details about the internal operation of the system and some others do not. One of the most important usages of SCA is in the detection of hardware trojans. So there are many SCA based methods used for the same. The electromagnetic modeling about the spectrum using side-channel methods was discussed without the use of the golden chips or layouts [1]. This method needs an activated trojan to detect it. The method discussed in [2] also detects hardware trojans on AES circuits. Initially, on a normal FPGA, the AES circuit was running after which a trojan’s leaked information is added. By using some instruments and adjusting the behavior of trojans, the SCA is performed to find the abnormality. In [3], SCA is used in the generation of keys based on biometrics, rather than finding the encryption key, it tries to find the signature of the user. In [4], a cross-device attack using deep learning is discussed on an AVR microcontroller. It takes into consideration, the variation within different devices by making a classifier using 30 devices in two groups. An SCA method [5] on the RSA algorithm was discussed. The data exploited were the power traces and electromagnetic radiation. The profiling attack on RSA algorithm considering the practical issues like preprocessing of observations and
acquisitions. A solution for some prevention techniques against side-channel attacks by creating some misalignment in traces is dealt with in [6]. It proposes a strategy based on CNN which is a replacement for the state of the art attacks, which will include the pre-processing details in the learning phase. The side-channel attack resistance was investigated of various FPGA hardware implementations of the ARIA block cipher in [7]. It showed that the unprotected implementation of the ARIA leads to the recovery of the secret key with the low power measurements.

In this article, the main SCA work that has been done is based on the power for detected hardware trojan. From the obtained traces, the change in the consumption of the power is used to analyze and to thereby get the secret information. The traces that are used for the SCA was collected while encrypting the plain-text. Since the deep learning model is used, the obtained data are used as the basis with which the model is made and trained by using the MLP and CNN described below.

2. Proposed Model- Side-channel Analysis using Deep Learning

2.1. Design Parameter for Side-channel Analysis

In any cryptographic algorithm, if we insert any hardware trojan or any external unwanted circuit, it will make it easy to monitor the changes in the design parameters such as power, path delay, etc [8]. Such vulnerable design parameters are known as side channels. Upon monitoring the values of these side channels with the help of various inputs, it is possible to find some secret information from the cryptographic model. The method of power analysis that has been used in this project was correlation power analysis. In this, we will have an assumed power consumption model which will be used to have a key guess and compare the correlation between the assumed and actual power consumption model. If a correlation exists, that means the guessed key is right.

2.2. Correlation Power Analysis

Initially, while encrypting the plain text, actual power values will be collected. Now, a new power consumption model is assumed. The simplest and most commonly used power consumption model is the Hamming Distance Model. Hamming distance between two binary numbers, x and y is the number of different bits. The attack happens at a particular point x in the encryption algorithm. If that variable has been changed to y by the victim, we calculate the hamming distance between x and y, if the power consumption model is the Hamming distance model. The attack is done sub-key wise. We will have a key guess. For each key guess and the known plain text, we calculate the power consumption according to the guessed model. Then, the correlation coefficient is found between the assumed and actual power consumption and finds the key which correlates the best. The same procedure is done for every 16 subkeys to find the entire key value. Initially, the power traces are collected from the system while encrypting the plain-text. After that, a power model is assumed by the adversary. With the help of the assumed model and a key guess performed by the adversary, an intermediate value is found out. The correlation coefficient of the intermediate value and the normal trace is found. The process of comparing the correlation between the original and assumed model is found out for every key guess and the one that best correlates with the actual model will be the correct key. Since the key-space of the AES circuit is very vast, the exploration space is divided into 16 subkeys each size is 1 byte long. So for getting the best correlation of the key guess, it has to try for every 256 possibilities that too 16 times to get the entire key.

2.3. Side Channel Analysis using Deep Learning

The profiling SCA has two phases - the characterization phase and the attacking phase. We can consider these phases similar to the training and testing phase. In the characterization phase, the attacker has to construct a generative model that estimates a probability value each
for every possible key value. When we are doing a known plain text attack, for a plain text, and a guessable key chunk, we will design a probability function, given those values what is the probability that the leakage variable $L$ is equal to the given value $1$. This model will be created for every possible key value. So the generative model $g(k)$ which is made for every possible key value, for the given leakage $1$ and plain text $p$. In the attack phase, the attacker has to correctly predict the key value. In this, the attacker gets a new attack set in which $(1,p)$ is there, and the key $k^*$ is fixed but unknown. From the generative model, the attacker finds the model which most likely knows the attack set. A maximum-likelihood strategy by estimating the likelihood for every candidate is found. For a subkey $j$, we take $n$ sets of power traces and plain texts and outputs a distinguishing vector $D$ which assigns a score for each possible subkey under consideration. With the help of the scores, the model will find the appropriate generative model and predict the key.

2.4. Dataset Used
The dataset will contain the traces that were collected while encrypting the plain-text in Zynq Board using an oscilloscope. This dataset is divided into two sets. One set, just like the training dataset, is needed to create the generative model (profiling model) as mentioned earlier. That set is known as profiling traces. The other one is the set of attacking traces used to retrieve the correct key from the available traces. Once the dataset is divided, it has to create a profile model. This model is created using neural networks. MLP and CNN are used to do the SCA and the results are compared [9]. After the model has been created using the set of profiling traces, the SCA is performed on the set of attacking traces to find the correct key guess.

3. Implementation Details
The hardware, software requirements along with the hyperparameter values for the proposed model is listed in Table 1.

| System Configuration | Software Requirements | Hyperparameter | Value |
|----------------------|-----------------------|----------------|-------|
| Processor 3.4GHz Octacore | Language Python | Optimizer RMSProp | |
| Memory 8GB RAM | Keras, Tensorflow, Sklear, Matplot | Learning rate 0.00001 | |
| Operating System Ubuntu 16.04 Libraries | Activation function ReLU, softmax | Weight Initialization Random | |
| Dataset Zynq Board | Loss Categorical cross-entropy | Number of hidden layers 4 | |

The dataset was composed of traces and the metadata. This is stored in a hierarchical data format. The metadata in HDF5 is a compound dataset which is similar to a struct in the C language. It will contain the mask, plain-text, cipher-text, etc. This dataset is divided into profiling set and attack set. 50,000 traces are used as profiling set and 10,000 traces are used for attacking set. There are three types of datasets. One is the same thing that has been mentioned before. In this, the traces are synchronized without any jitter. Another two datasets, with desynchronization factors of 0 and 100 respectively. Table 1 shows the hyperparameters that were used for tuning the MLP model. The training set, that is the profiling set, which contains 50,000 tuples, 45,000 tuples were used for the learning purpose and the 5,000 were used for validating the model. Then another 10,000 tuples were used as the test split. This problem of finding the correct key can be treated as a classification problem. That is, prediction of the key value of AES circuit completely in one go is difficult. So we split the keyspace into 16 parts, each part having a size of 1 byte. So the possibility of correct sub-key in one round
is 256. So we have to classify a given trace into any of the possible 256 classes. So that is the reason why we are using categorical cross-entropy as the loss function. Regarding the CNN model, most of the hyperparameter values that have been used are the same. There are 5 pooling layers used on CNN. The optimizer used is RMSProp, which normalizes the gradient by keeping a moving average of root mean square values. An activation function is certain standard mathematical equations that help in finding the output of a neural network [10]. This function will be associated with each neuron in the network and determines if it has to be activated, based on the input given by each neuron is relevant for predicting the final output correctly. Activation functions also help normalize the output of each neuron to a range. The activation functions used in this project are reLU and softmax. ReLU is also known as a rectified linear unit will give an output zero if the input is less than zero and will give output one, otherwise. It has better gradient propagation, efficient computation, and is scale-invariant. In the final layer, we use softmax, because it will give a value between 0 and 1, and is, therefore, probability distribution.

3.1. Feed-forward Neural Network
The weights on each edge are initialized into a random value at first. Then each layer will feed its output to the next layer as its input and final output is predicted. This is known as a feed-forward network because the flow of information takes place in the forward direction, as x is used to calculate some intermediate function in the hidden layer which in turn is used to calculate y. In this, after the neural network predicts an output, the weights of each link are adjusted so that the error rate is reduced and the model will make a better prediction in the next run. For each trace, we are shifting the trace by a random number x [7]. This random number can be chosen between a value of [0..N], where N has been chosen as 50 and 100 in the two data sets that have been used. This is used to find the efficiency of the model. After we find a random value x for every trace, we shift the trace to the left. In this way, 2 new data sets are created from the original one. Once the data set is created, we train for the MLP and CNN models and run on the attack set. Finally, the rank of the output is compared to confirm if the key that has been found is correct or not. The dataset with desynchronized traces helps to find which of the neural network model outperforms the other in the presence of jitters.

4. Results and Discussion
The figures show the results of the power analysis. Figure 1 and figure 2 show the key guess for one byte and the correlation coefficient for all the values respectively. It is plotted between the correlation coefficient and the number of traces. For each key guess, the correlation coefficient is shown, the one for which it shows a peak coefficient is a right key. This shows the 16 correlation power analysis retrieving all the key bytes by showing the progression of each attack depending
Figure 2. Correlation coefficient of a power trace for possible key values

Figure 3. MLP with no desynchronization and MLP with desynchronization value of 100

Figure 4. CNN with no desynchronization and CNN with desynchronization value of 100

on the number of traces. Figure 2 shows the second-order correlation power analysis to retrieve AES key byte. The second-order means that the sensitive data is masked. A centered product processing method is used to recombine the samples and finally, classical CPA is used to find the key byte. In AES-128, the size of the key is 128 bits. So, the keyspace will be in the order of $2^{128}$, which is very high. In such a case, the guessing of the key from that big range is extremely difficult. So, to make the key guess possible, we divide the entire key into 16 subkeys each of size 8 bits. So, the sub keyspace is reduced to 256, which is computable. So, in the following, each sub-key byte is calculated. In figure 2, we are drawing the correlation coefficient of a power trace concerning every possible key value, and it is found that for a particular key, it is showing a very high correlation factor which shows that it is a correct key.

The evaluation metric used is the rank. For every key guess, a hypothesis function is run. The key which gets the highest value will be considered as a candidate key. Now, the rank of the same is measured. That is the number of values that give the value greater than the
hypothesis which means if the key guess was right, then the candidate key will have a rank zero. The results obtained when the model is trained with CNN and MLP are shown in figures 3 and figure 4 respectively. We can see that in the case of without desynchronization if we compare the MLP and CNN, the rank of the candidate key is approaching zero with the number of traces around 100 on CNN. But, in the case of MLP, although the rank is getting converged to zero at a point, it takes around 400 traces. So, we can see that CNN outperforms MLP in the case without desynchronization. In the case of desynchronization, where a random shift of the traces is made, to allow jitter, both the methods are taking more time to converge. In those cases also CNN is found to be better. In figure 3, we can see that the MLP is not getting converged to a rank of zero, but we can see that in figure 4, the rank is getting converged to zero, because of which we can conclude that CNN was the best option among the two models.

5. Conclusion
The performance of SCA is evaluated using deep learning in the presence of jitter, in which two datasets with different levels of de-synchronizations were tried and their rank is compared using both the MLP and CNN models. The desynchronization was added up to a maximum of 50 in one dataset and 100 in the next. The traces are shifted left up to a random value and thus create the new set of data. In this, CNN was found to be converging to a rank zero with the given traces and found to be giving a better performance as compared to MLP. Thus, to conclude, CNN was the better choice. This work can be extended by trying for other neural network or machine learning models. Also, this can be extended to work on an extended keyspace of AES cryptography. Also, for masked implementation of the AES, the work can be tried out.

References
[1] J. He, Y. Zhao, X. Guo, and Y. Jin. 2017. Hardware trojan detection through chip free electromagnet- ic-side-channel statistic analysis. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 25(10): 2939-2948.
[2] B. Hanindhito and Y. Kurniawan. 2019. Hardware trojan design and its detection using side-channel analysis on cryptographic hardware aes implemented on fpga. In 2019 International Conference on Electrical Engineering and Informatics (ICEEI) :191196.
[3] D. Delivasilis and S. Katsikas. 2006. Side channel analysis on biometric-based key generation algorithms on resource constrained devices. International Journal of Network Security 3(1).
[4] A. Golder, D. Das, J. Danial, S. Ghosh, S. Sen, and A. Raychowdhury. 2019. Practical approaches toward deep-learning-based cross-device power side-channel attack IEEE Transactions on Very Large Scale Integration (VLSI) Systems 27(12): 2720-2733.
[5] M. Carbone, V. Comin, and M. Cornelie. 2019. Deep learning to evaluate secure rsa implementtions. IACR Cryptology ePrint Archive.
[6] E. Prouff, E. Cagli, and C. Dumas. 2017. Convolutional neural networks with data augmentation against jitter-based countermeasures -proliting at-tacks without pre processing. in Cryptographic Hardware and Embedded Systems - CHES 2017 19th International Conference: 4568.
[7] M. S. ChangKyun Kim and S. Moon. 2008. Differential side channel analysis attacks on fpga implementations of ari. in ETRI journal 30(2): 315-325.
[8] A.Kulkarni, Y.Pino, and T.Mohsenin. 2016. Adaptive real time Trojan detection framework through machine learning, in 2016 IEEE International Symposiumon Hardware Oriented Security and Trust (HOST), : 120123
[9] R. Benadjila, E. Prouff, R. Strullu, E. Cagli, and C. Dumas. 2018. Study of deep learning techniques for side-channel analysis and introduction to ascad database in Cryptology ePrint Archive.
[10] L.T.Vanschoren. 2019. Openml: networkedscienceinmachinelearning. In SIGKDD Explorations, vol. 15.