Deep Learning-based Side Channel Attack on HMAC SM3

Xin Jin[1], Yong Xiao[1], Shiqi Li[2]*, Suying Wang[2]

[1] CSG Electric Power Research Institute, Guangzhou (China)
[2] Open Security Research, Inc., Shenzhen (China)

Received 1 July 2020 | Accepted 2 November 2020 | Published 16 November 2020

Abstract

SM3 is a Chinese hash standard. HMAC SM3 uses a secret key to encrypt the input text and gives an output as the HMAC of the input text. If the key is recovered, adversaries can easily forge a valid HMAC. We can choose different methods, such as traditional side channel analysis, template attack-based side channel analysis to recover the secret key. Deep Learning has recently been introduced as a new alternative to perform Side-Channel analysis. In this paper, we try to recover the secret key with deep learning-based side channel analysis. We should train the network recursively for different parameters by using the same dataset and attack the target dataset with the trained network to recover different parameters. The experiment results show that the secret key can be recovered with deep learning-based side channel analysis. This work demonstrates the interests of this new method and show that this attack can be performed in practice.

Keywords

CNN, Neural Network, HMAC, Side Channel Analysis, HMAC-SM3.

DOI: 10.9781/ijimai.2020.11.007

I. Introduction

Side channel analysis is a powerful technique that helps an adversary recover sensitive information without damaging the target. Target device leaks information (e.g. power consumption [1], Electromagnetic emanation [2], temperature [3], acoustic [4], etc.) that is related to sensitive data during calculation [5]. An adversary can make use of the leakage to recover sensitive data. In order to decrease the leakage, several countermeasures such as masking and hiding are used in the cryptographic implementation. However, even with countermeasures, adversaries can come up with more powerful methods to recover the sensitive information.

Following the current trend in the side channel analysis research area, recent works have demonstrated that deep learning algorithms were very efficient to conduct security evaluations of embedded systems and had many advantages compared to the other methods.

Nowadays, machine learning becomes a popular topic in many areas. It is usually divided into three classes: supervised learning, unsupervised learning and semi-supervised learning. In most situation, supervised learning is mainly used. There are many kinds of structures which are used in machine learning, such as Support Vector Machine (SVM) and Random Forest, etc. Deep learning is a kind of machine learning. It extracts features by several non-linear layers. Deep learning becomes popular since AlexNet [6] is proposed in 2012. Then, more and more complex network structures are proposed, such as VGGNet [7], GoogLeNet [8] and ResNet [9], etc. These networks work well in many areas, e.g. image recognition area, face recognition area and so on.

* Corresponding author.

E-mail address: shiqi.li@osr-tech.com

In recent years, it appears that deep learning techniques are applied in side channel analysis research area [10], [11]. Comparing to traditional side channel method, deep learning-based side channel analysis performs better. Deep learning-based side channel analysis performs better especially when the implementation has mask or jitter [12]. Without alignment or pre-processing, neural network can recover sensitive information as well, which is much more convenient than the traditional side channel method. Many researches are done in recent years. In 2013, Martinasek. et al., play an attack on an AES implementation working on PIC16F84A with only one hidden layer [13]. Another work [14] compares different kind of machine learning method on DPA contest V2 dataset. A research [15] proposed a CNN based side channel analysis and claim that this method is robust to trace misalignment. Different structures of network are applied on the dataset and come up with a CNN-best structure for the dataset. Picek et al. compare different methods for the class imbalance situation on DL-SCA [16]. Last, a research [17] come up with correlation-based loss function.

In this paper, we use a deep learning-based side channel analysis technique to analyze HMAC SM3 algorithm implementation. The structure of this paper is as follows: In Section II, we introduce the SM3 algorithm, HMAC SM3, basic idea of CNN as well as the attack path. This section will help readers have a basic understanding of the algorithm and the attack method. The attacks on real traces are demonstrated in Section III. In this section, the target, the structure of the network and the attack are illustrated. In the end, conclusion and future work are presented in Section IV.

II. Background

A. SM3 Algorithm

SM3 is the Chinese hash standard [18]. The structure of the algorithm is shown in Fig. 1. The input data of the function is padded such that it can be split into N blocks of 512 bits. Each block will be treated in a same procedure: the former block calculates a new IV for the latter block through function f(), and the output of block N is the hash result of the algorithm.

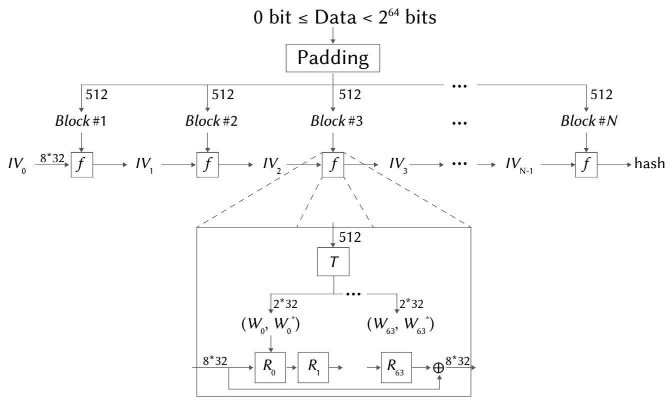


Fig. 1. Structure of SM3 algorithm.

The structure of function f() is show in Fig 2. The function T() convert the 512-bit input into 64 32-bit word pairs. Each pair (Wi, Wi*) are used during round Ri, and the result of each round is used as input of the next round. When the 64th round is completed, a final transformation is applied by adding the input of the first round and the output of the last round together as the output of the function f().

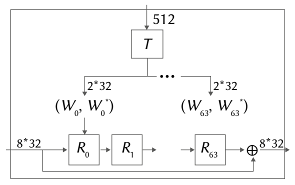


Fig. 2. Structure of the f function.

In order to explain the detail of each loop, we define the loop by function: $IV_i = f(IV_{i-1}, Block_i)$

The first constant IV_0 is:

$$IV_{0,0} = 0x7380166F \quad (1)$$

$$IV_{0,1} = 0x4914B2B9 \quad (2)$$

$$IV_{0,2} = 0x172442D7 \quad (3)$$

$$IV_{0,3} = 0xDA8A0600 \quad (4)$$

$$IV_{0,4} = 0xA96F30BC \quad (5)$$

$$IV_{0,5} = 0x163138AA \quad (6)$$

$$IV_{0,6} = 0xE38DEE4D \quad (7)$$

$$IV_{0,7} = 0xB0FB0E4E \quad (8)$$

The detail of each loop is as follows:

First, initialize the 8 32-bit local variables named a to h:

$$a_0 = IV_{i-1,\cdots 0} \quad (9)$$

$$b_0 = IV_{i-1,\cdots 1} \quad (10)$$

$$c_0 = IV_{i-1,\cdots 2} \quad (11)$$

$$d_0 = IV_{i-1,\cdots 3} \quad (12)$$

$$e_0 = IV_{i-1,\cdots 4} \quad (13)$$

$$f_0 = IV_{i-1,\cdots 5} \quad (14)$$

$$g_0 = IV_{i-1,\cdots 6} \quad (15)$$

$$h_0 = IV_{i-1,\cdots 7} \quad (16)$$

For each round R_j, $j \in [0,63]$, we compute:

$$SS1_j = ((a_j <<< 12) + e_j + (T_j <<< j) <<< 7) \quad (17)$$

$$SS2_j = SS1_j \oplus (a_j <<< 12) \quad (18)$$

$$TT1_j = FF_j(a_j, b_j, c_j) + d_j + SS2_j + W_j^* \quad (19)$$

$$TT2_j = GG_j(e_j, f_j, g_j) + h_j + SS1_j + W_j \quad (20)$$

$$a_{j+1} = TT1_j \quad (21)$$

$$b_{j+1} = a_j \quad (22)$$

$$c_{j+1} = b_j <<< 9 \quad (23)$$

$$d_{j+1} = c_j \quad (24)$$

$$e_{j+1} = P_0(TT2_j) \quad (25)$$

$$f_{j+1} = e_j \quad (26)$$

$$g_{j+1} = f_j <<< 19 \quad (27)$$

$$h_{j+1} = g_j \quad (28)$$

where all additions are done modulo 2^{32} <<< n means left rotation of n bits, constants T_j is:

$$T_j = \begin{cases} 0x79CC4519, & 0 \le j \le 15 \\ 0x7A879D8A, & 15 < j < 64 \end{cases} \quad (29)$$

Function FF_j is:

$$FF_j(X,Y,Z) = \begin{cases} X \oplus Y \oplus Z, & 0 \le j \le 15 \\ (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z) & 15 < j < 64 \end{cases} \quad (30)$$

Function GG_j is:

$$GG_j(X,Y,Z) = \begin{cases} X \oplus Y \oplus Z, & 0 \le j \le 15 \\ (X \wedge Y) \vee (\neg X \wedge Z) & 15 < j < 64 \end{cases} \quad (31)$$

Function P_k is:

$$P_k(X) = \begin{cases} X \oplus (X \lll 9) \oplus (X \lll 17), & k = 0 \\ X \oplus (X \lll 15) \oplus (X \lll 23), & k = 1 \end{cases} \quad (32)$$

Input plaintext of each block $Plain_{Block}$ is split into 32-bit words $Plain_{Block} = \{PB_0, PB_1, \ldots, PB_{15}\}$. Then the parameter W_j is computed as:

$$W_j = \begin{cases} PB_j, & 0 \le j \le 15 \\ P_1(W_{j-16} \oplus W_{j-9} \oplus (W_{j-3} \lll 15)) \oplus (W_{j-13} \lll 7) \oplus W_{j-6}, & 15 < j < 68 \end{cases} \quad (33)$$

And the parameter W_j^* is computed as:

$$W_j^* = W_j \oplus W_{j+4} \quad (34)$$

The function f() is finished by 32-bit XOR with the initial state:

$$IV_{i,0} = IV_{i-1,0} \oplus a_{64} \quad (35)$$

$$IV_{i,1} = IV_{i-1,1} \oplus b_{64} \quad (36)$$

$$IV_{i,2} = IV_{i-1,2} \oplus c_{64} \quad (37)$$

$$IV_{i,3} = IV_{i-1,3} \oplus d_{64} \quad (38)$$

$$IV_{i,4} = IV_{i-1,4} \oplus e_{64} \quad (39)$$

$$IV_{i,5} = IV_{i-1,5} \oplus f_{64} \quad (40)$$

$$IV_{i,6} = IV_{i-1,6} \oplus g_{64} \quad (41)$$

$$IV_{i,7} = IV_{i-1,7} \oplus h_{64} \quad (42)$$

B. SM3 Based HMAC

The HMAC stands for keyed- Hash Message Authentication Code and is a NIST standard which can be found in [19]. Fig. 3 presents the process of HMAC SM3.

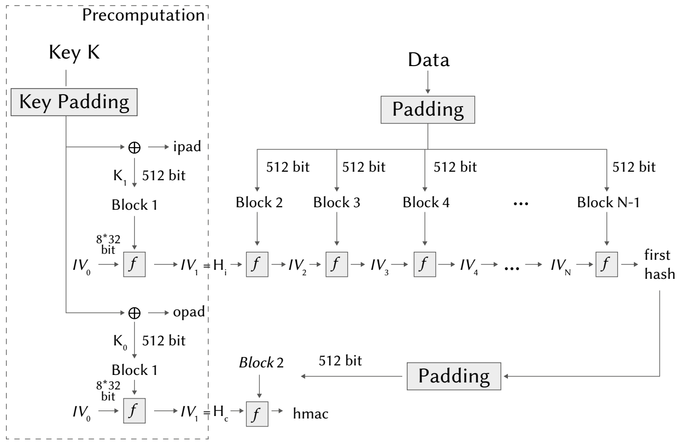


Fig. 3. Structure of HMAC SM3.

The process of HMAC SM3 is as follows:

Derive key pair (K_i, K_o) from the key K.

Calculate the first hash with K_i and input data: first hash = H(K_i | T)

Calculate HMAC with K_o and first hash: HMAC = H(K_o | first hash), where H() is the SM3 hash function.

C. Side Channel Analysis

Side Channel Analysis (SCA) is first proposed by Kocher et al. in 1996[22]. It is a technique to retrieve the secret information of an algorithm by monitoring the physical information of a device (such as power, heating, time consuming, electromagnetic signals, etc., as shown in Fig. 4. The reason that SCA can recover secret is that the physical signal of a cryptographic device demonstrates correlation with the internal statement.

It is much easier to recover information from side channel signals than directly breaking the core implementation. There are several kinds of SCA, e.g. simple power analysis, correlation power analysis, template attack, etc. Simple power analysis [23] is an easy way to recover secret information. By observing the side channel signals, the attacker can find the difference and recover the sensitive information according to the differences. Correlation power analysis (CPA) [24] needs much more traces. When using CPA to recover sensitive information, we need to guess the secret key to calculate a certain mid-value. Since different traces correspond to different plaintext, we can have a set of mid-value for every guesses. By computing the correlation between mid-values and the side channel signals, we can figure out the correct guess. Template attack (TA) is another kind of passive attack. It has two stages: first, template building, second, template matching. Deep learning based SCA is similar to TA. We will discuss TA and deep learning based SCA in the following section.

D. Deep Learning Based Side Channel Analysis

Template attack [20] is a traditional method of side channel analysis. During the attack, we should take a reference set of trace from a reference device first in the learning phase. For this set, we know the key, the plaintext and all the details of every trace. We can set up templates for each mid-value using the reference set. For attacking phase, we can use the templates in the learning phase to attack the target trace set to recover the mid-value such that the secret key can be recovered as well.

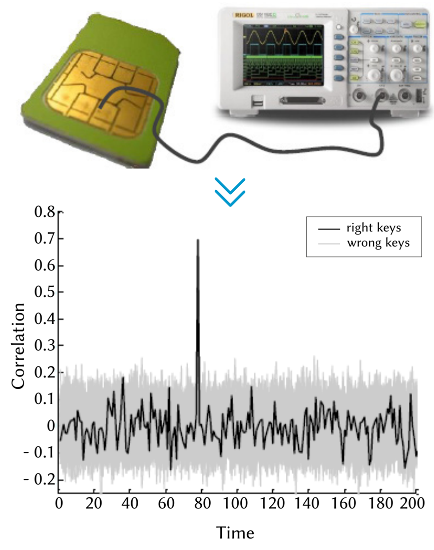


Fig. 4. Side Channel Leakage.

Deep learning-based side channel analysis is similar to the traditional template at-tack, which has two phases: a learning phase and an attacking phase. The whole procedure is shown in Fig. 5.

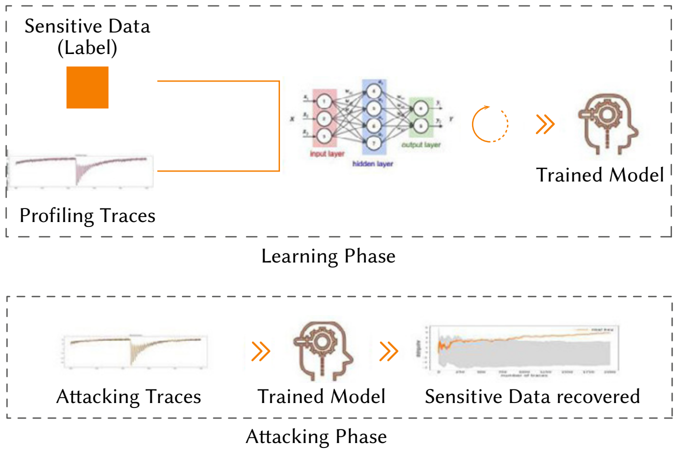


Fig. 5 .The procedure of deep learning-based side channel analysis.

In the learning phase, a trace set is collected from the reference device. For each trace, we add a label that is related to the sensitive data (e.g. key). The neural network will be trained with the traces and labels. Parameters in the neural network is updated. The goal of the learning phase is to try to make the output (prediction) of the neural network be closer to the true label. After the learning phase, the parameters updated in the network will be saved.

The network saved at the end of learning phase can be applied to the attacking phase. In the attacking phase, we have a trace set with only traces but we do not know the label. With each trace, the network can give us a prediction of the label using the parameters saved in the learning phase. We can recover the sensitive data (e.g. key) according to the predictions.

E. Neural Network Architecture

In this paper, we will primarily focus on Convolutional Neural Network. We will introduce the basic idea of the Convolutional Neural Network.

Convolutional Neural Network (CNN) [21] is a popular network in deep learning do-main. Usually, CNN is consisting of three kinds of layers: convolutional layers, pooling layers, as well as fully connected layers. The convolutional layers work as pattern detectors by convolution operations. Kernels in Convolutional layers can be updated by backpropagation. In this way, we can get different kernels using the same set of input. These kernels can detect different kinds of edges to track different features. Usually one kernel corresponds to one feature and a convolutional layer contains several kernels. In the meantime, since the kernels doing convolution by moving through the whole dataset, same pattern in different positions can be detected by the same kernel. We should notice that the kernels in the convolutional layer always have a small size compare to the input data, which reduce the computation complexity of the neural network.

Pooling layers always come after convolutional layers, which reduce the size of the inputs. We can choose average pooling to reduce the size by local averaging or max pooling to reduce the size by picking up the max value in a certain area. By averaging or max pooling, the pooling layer extract features of the input and reduce the size. This operation makes the CNN more robust to the shift and deformation of the input. In addition, it can reduce the possibility of overfitting since it reduces the size of the input data.

Fully connected layer usually comes at the end of the neural network. Each neural in fully connected layers are connect to every input.

We can choose the number of convolutional layers, pooling layers and fully connected layers arbitrarily. With more layers, the neural network can learn more com-plex features. However, with more layers, the network will be easier to get over-fit. Thus, the structure should be chosen carefully according to the input data.

The detailed architecture of MLP_{best}, $MLP_{monobit}$, $MLP_{multi-label}$, CNN_{best}, $CNN_{monobit}$ and $CNN_{multi-label}$ is shown in Fig. 6. The mark "FC-200" means a fully connected layer of 200 neurons, "conv11-64, Relu" means 64 convolutional kernels of size 11 using Relu activation function, and "average pooling, 2 by 2" means an average pooling layer, whose pooling window size is 2 and the stride is 2.

For a deep insight into the differences of identity model and our multi-label model, the architecture of the output layers of CNN_{best} and $CNN_{multi-label}$ (the same for MLP_{best} and $MLP_{multi-label}$) is depicted in Fig. 7, the output layer of CNN_{best} has 256 output neurons with softmax activation function while the output layer of $CNN_{multi-label}$ has 8 neurons with sigmoid activation function. Correspondingly, CNN_{best} uses cross-entropy as the loss function and $CNN_{multi-label}$ utilizes the binary cross-entropy since there are 8 binary labels.

F. Attack Path

Since SM3 algorithm has no secret key, we cannot attack SM3 directly. We can only attack HMAC SM3 to recover the key K used in the HMAC process. In order to recover the key K, we should recover the key pair (K_i, K_o) first. Thus, we should recover the first hash IV: Hi and the second hash IV: Ho to recover the key pair.

We use Ho and first hash result to calculate the HMAC. To recover Ho, we should know the first hash result first. In order to get the first hash result, we should recover the first hash IV(H_i) first. Recovering Hi and Ho can use the same process. If H_i is recovered, Ho can be easily recovered as well. In this paper, we only consider recovering H_i and our target is to recover a_0, b_0, c_0, d_0, e_0, f_0, g_0 and h_0 related to H_i.

MLP_{best}	$MLP_{monobit}$	$MLP_{multi-label}$	CNN_{best}	$CNN_{monobit}$	$CNN_{multi-label}$
6 weights layers			8 weights layers		
352,456 params	301,201 params	302,608 params	66,652,544 params	65,607,809 params	65,636,488 params
input					
FC-200, Relu			conv11-64, Relu		
			average pooling, 2 by 2		
FC-200, Relu			conv11-128, Relu		
			average pooling, 2 by 2		
FC-200, Relu			conv11-256, Relu		
			average pooling, 2 by 2		
FC-200, Relu			conv11-512, Relu		
			average pooling, 2 by 2		
FC-200, Relu FC-200, Relu			conv11-512, Relu		
			average pooling, 2 by 2		
			FC-4096, Relu		
			FC-4096, Relu		
FC-256 softmax	FC-1, sigmoid	FC-8, sigmoid	FC-256, sigmoid	FC-1, sigmoid	FC-8, sigmoid

Fig. 6. Details of NN architecture.

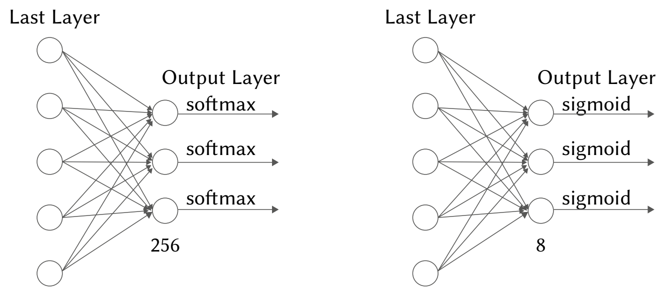


Fig. 7. The difference of output layers between CNN_{best} and $CNN_{multi-label}$. Left: the output layer of CNN_{best}. Right: the output layer of $CNN_{multi-label}$.

In order to illustrate the attack path more clearly, we denote part of the $TT1_j$ and $TT2_j$ computation as $\delta_{1,j}$ and $\delta_{2,j}$ respectively:

$$\delta_{1,j} = FF_j(a_j, b_j, c_j) + d_j + SS2_j \tag{43}$$

$$\delta_{2,j} = FF_j(e_j, f_j, g_j) + h_j + SS1_j \tag{44}$$

We can first recover $\delta_{1,0}$ and $\delta_{2,0}$ according to Equation (3) and Equation (4) respectively when j is equal to 0. With $\delta_{1,0}$ and $\delta_{2,0}$ known, $TT1_0$ and $TT2_0$ can be easily calculated since W_0^* and W_0 are known. Then, we can recover a_0 by targeting at $TT1_0 \oplus a_0$, recover b_0 by targeting at $TT1_0 \oplus a_0 \oplus (b_0 <<< 9)$. c_0 can be recovered through targeting at the computation $c_0 + FF_0(TT1_0, a_0, (b_0 <<< 9) + SS2_1 + W_1^*$. After $\delta_{1,0}$, a_0, b_0 and c_0 are recovered, we can simply recover d_0 by computing $d_0 = \delta_{1,0} - (a_0 \oplus b_0 \oplus c_0) - SS2_0$. Similarly, we can recover e_0, f_0, g_0 and h_0 with $TT2_0$ and W_1. Thus, the IV H_i can be recovered.

III. Attack on Real Traces

A. Experiment Setup and Data Set

The testing target is a software HMAC SM3 running on a 32-bit microprocessor Infineon TC1782. The experiment setup consists of a high-performance Digital Storage Oscilloscope (DSO), high-precision XYZ stage and near-field high-bandwidth EM probe, as shown in Fig. 8.

EM traces are acquired when the HMAC SM3 is running. A single measurement contains 50,000 points, representing the computation of first hash.

Fig. 8. Experiment equipment and devices.

We collect two set of traces: Set A: 200,000 traces with input data and the first hash IV Hi varying; Set B: 50,000 traces with variance input data and fixed first hash IV Hi. Set A is used in the learning phase, while Set B is used for the attack phase. In the training phase, 180,000 traces of Set A are used as training set while the rest 20,000 traces are used as validation set to choose the best network parameter.

B. Neural Network Structure

Fig. 9 shows the structure of the network. We only use one convolutional layer with kernel size 3 and 32 convolutional filters. For the pooling layer, we use set both the pooling size and the stride to 2. The first fully connected layer has 1024 neuros while the second has 512 neuros. The input layer contains 5000 neuros while the output layer contains 9 neuros, which stands for Hamming Weight 0 to Hamming Weight 9.

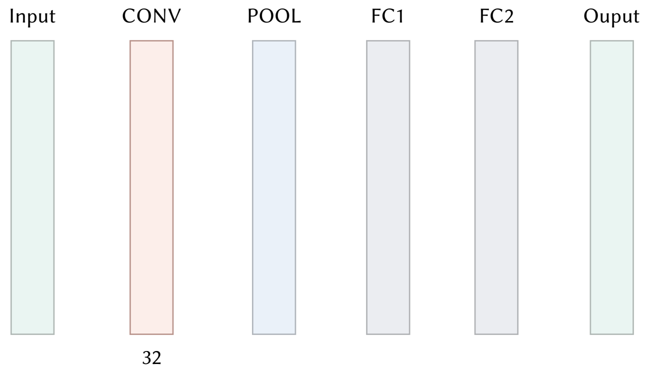


Fig. 9. Structure of the Neural Network.

The network contains 82,450,569 parameters in total, as shown in Fig. 10.

C. Experimental Result

We try to recover $\delta_{1,0}$ first. Instead of recover all 32-bits of $\delta_{1,0}$, we recover $\delta_{1,0}$ byte by byte. With learning rate 0.0001, batch size 200, we trained each model 10 epochs using Set A. The training result is shown in Fig. 11. the blue line corresponds to the training set while the orange line corresponds to the validation set. We can find that for every byte, the loss increases and the accuracy decrease in the validation set after several epochs. Thus, we save the network with best performance instead of the network obtained when training is finished.

```
-------------------------------------------------------------------------
Layer (type)                 Output Shape                 Param #
=========================================================================
Input (InputLayer)           (None, 5000, 1)              0
-------------------------------------------------------------------------
block1_conv (Conv1D)         (None, 5000, 32)             128
-------------------------------------------------------------------------
block1_pool (AveragePoolingl (None, 2500, 32)             0
-------------------------------------------------------------------------
flatten (Flatten)            (None, 80000)                128
-------------------------------------------------------------------------
fc1 (Dense)                  (None, 1024)                 81921024
-------------------------------------------------------------------------
fc2 (Dense)                  (None, 512)                  524800
-------------------------------------------------------------------------
predictions (Dense)          (None, 9)                    4617
=========================================================================
Total params: 82, 450, 569
Trainable params: 82, 450, 569
Non-trainable params: 0
-------------------------------------------------------------------------
```

Fig. 10. Parameters and Structure of the Neural Network.

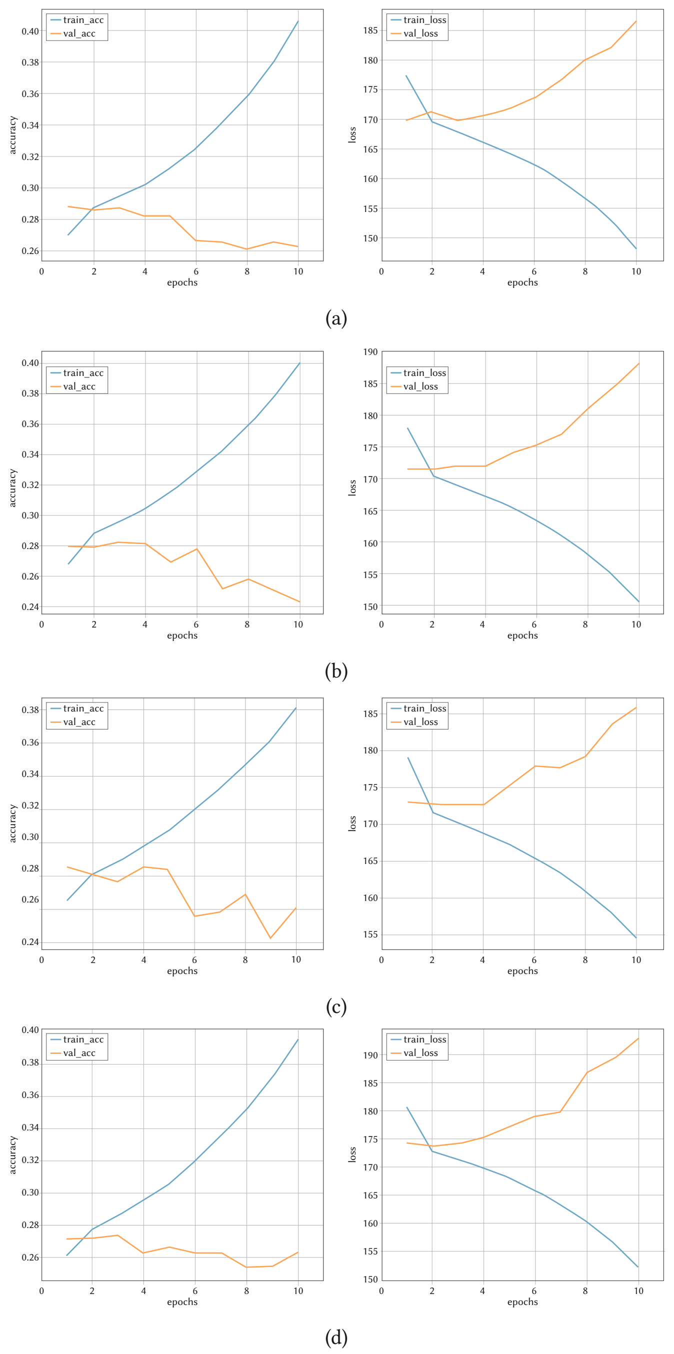
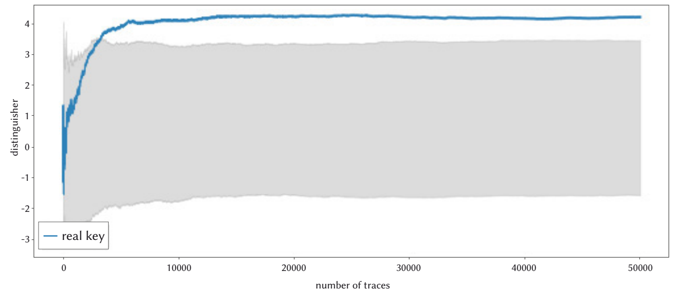
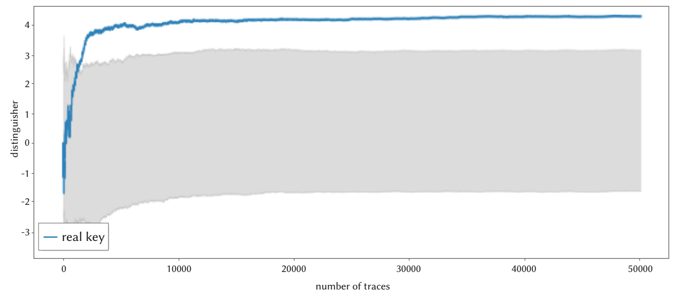


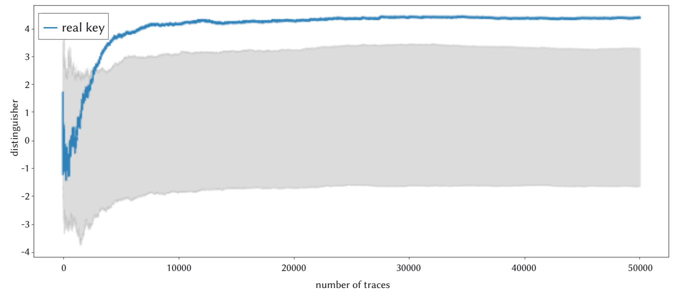
Fig. 11. Training result of $\delta_{1,0}$: (a) Byte3 (b) Byte 2 (c) Byte 1 (d) Byte 0.



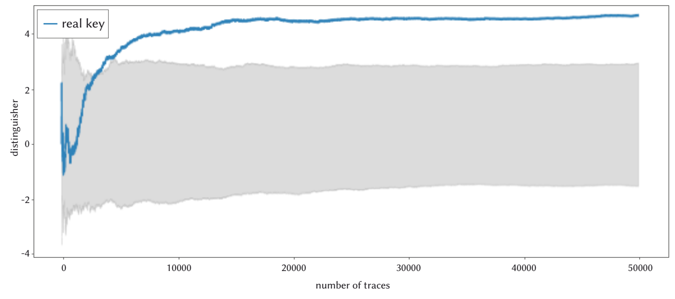
(a)



(b)



(c)



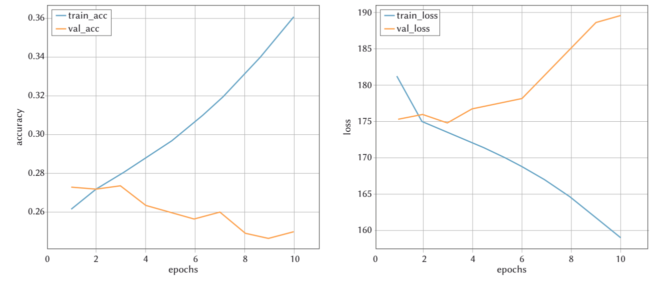
(d)

Fig. 12. Attack result of $\delta_{1,0}$: (a) Byte 3 (b) Byte 2 (c) Byte 1 (d) Byte 0.

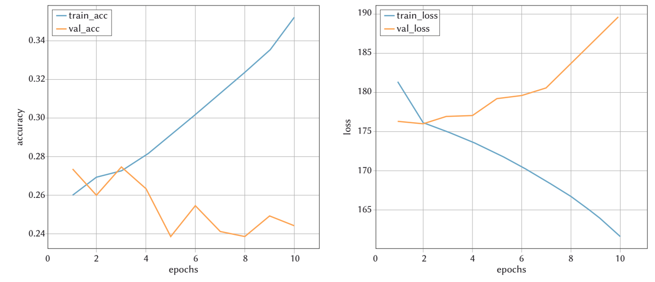
The attack results on Set B are shown in Fig. 12. The line in blue indicates the expected value of different bytes. We can find that all bytes in $\delta_{1,0}$ can be recovered with only several thousands of traces in Set B.

With $\delta_{1,0}$ recovered, we can calculate $TT1_0$ for every trace according to the corresponding W_0^*. The EM traces leaks information related to a_0 when calculating $a_0 \oplus TT1_0$. The training result is shown in Fig. 13.

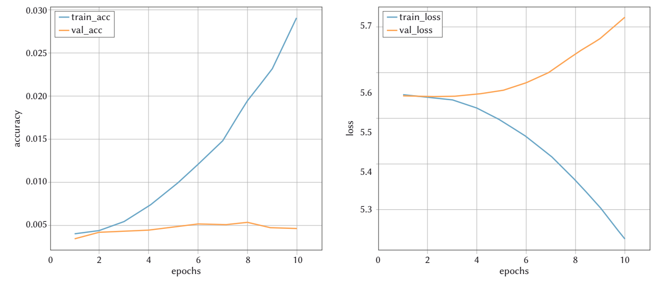
We recover a_0 byte by byte as well. The result is shown in Fig. 14. The line in blue indicate the expected value of different a_0 bytes. We can find from the result that we need to use almost all traces in Set B to recover all the four bytes of a_0. Unlike the result of $\delta_{1,0}$, the correct candidates of a_0 are not very distinguishable from other candidates. The result of $\delta_{1,0}$ seems more distinguishable than that of a_0, the reason may be that the process that leaks information about a_0 is a XOR operation while the leakage about $\delta_{1,0}$ is an ADD operation.



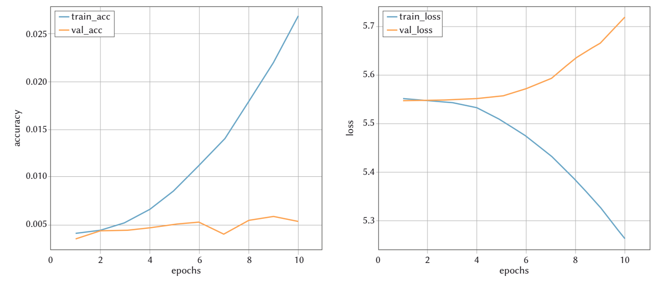
(a)



(b)

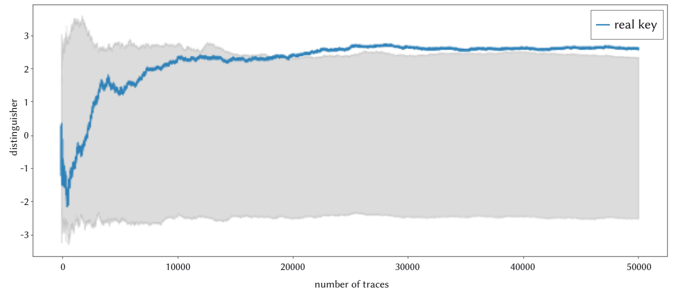


(c)

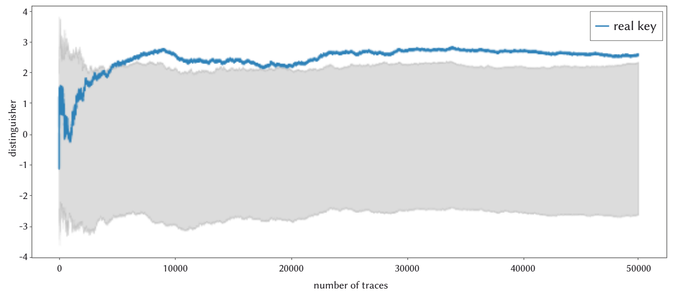


(d)

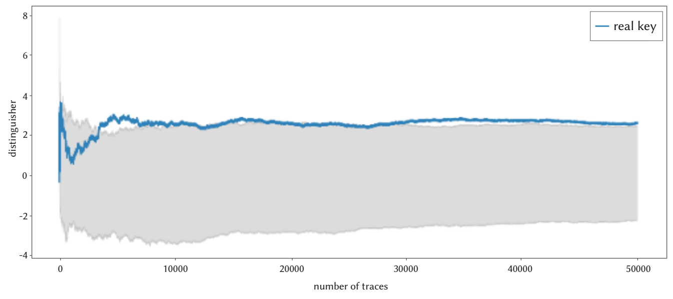
Fig. 13. Training result of a_0: (a) Byte 3 (b) Byte 2 (c) Byte 1 (d) Byte 0.



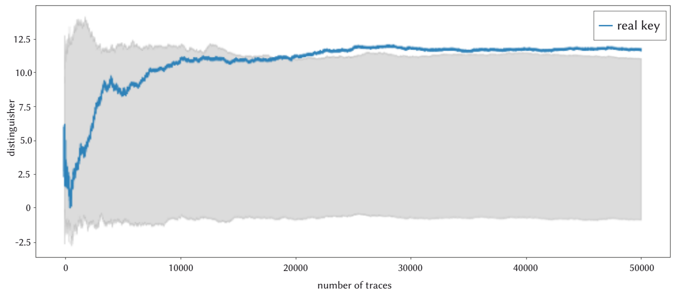
(a)



(b)



(c)



(d)

Fig. 14. Attack result of a_0: (a) Byte 3 (b) Byte 2 (c) Byte 1 (d) Byte 0.

We can repeat the attack on all the other parameters using the same set of data by changing the target label to recover b_0, c_0, $\delta_{2,0}$, e_0, f_0 and g_0. Then, d_0 and h_0 can be calculated simply.

IV. Conclusion and Future Work

In this paper, we demonstrate a Deep Learning-based Side Channel Attack on HMAC SM3 algorithm. In order to recover the key used in HMAC SM3, the attacker should recover two IVs: Hi and Ho. In this paper, we only focus on recovering Hi since the method of recovering the two IVs are the same. We try to recover $\delta_{1,0}$, $\delta_{2,0}$, a_0, b_0, c_0, d_0, e_0, f_0, g_0 and h_0 to recover H_i. The experiment result shows that we can recover the IV with 50,000 traces. In addition, we can find that when we focus on an add operation, the attack result is much better than focusing on a XOR operation. Thus, we need more traces to recover parameters when focusing on XOR operations. Although the correct candidate for XOR operation is not quite distinguishable from other candidates, we can recover the correct candidate. This situation may be solved if more traces are added to the attacking set.

In this paper, we focus on a software implementation of HMAC SM3 without any countermeasures. In future work, we can try several different HMAC implementations: (a) hardware implementation without countermeasures; (b) software implementation with some countermeasures; (c) hardware implementation with countermeasures. By doing experiments on different implementations, we can check whether deep learning works well on both unprotected and protected situations. In addition, we can try to figure out the difference of the structure of the network when attacking a hardware implementation and a software implementation.

Acknowledgment

This work was supported by Electric Power Research Institute from China Southern Power Grid in Guangzhou, China under project “Security analysis research on smart meter in power grid system”.

References

[1] Kocher P, Jaffe J, Jun B. “Differential power analysis, advances in cryptology”-CRYPTO’99, Proc.19th Annual International Cryptology Conf, pp. 388-397, 1999.
[2] Gandolfi K, Mourtel C, Olivier F. “Electromagnetic analysis: Concrete results”, International workshop on cryptographic hardware and embedded systems. Springer, Berlin, Heidelberg, pp. 251-261, 2001.
[3] Brouchier J, Kean T, Marsh C, et al. “Temperature attacks.” IEEE Security & Privacy, vol. 7, no. 2, pp. 79-82, 2009.
[4] Thompson, J.N. “Insect Diversity and the Trophic Structure of Communities”. In: Ecological Entomology. New York. pp. 165-178, 1994.
[5] Robyns, Pieter, Peter Quax, and Wim Lamotte. “Improving CEMA using correlation optimization.”, 2018.
[6] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. “Imagenet classification with deep convolutional neural networks.” Advances in neural information processing systems, 2012.
[7] Simonyan, Karen, and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition.” arXiv preprint arXiv: pp. 1409-1556, 2014.
[8] Szegedy, Christian, et al. “Going deeper with convolutions.” Proceedings of the IEEE conference on computer vision and pattern recognition, 2015.
[9] He, Kaiming, et al. “Deep residual learning for image recognition.” Proceedings of the IEEE conference on computer vision and pattern recognition, 2016.
[10] Bartkewitz T, Lemke-Rust K. “Efficient template attacks based on probabilistic multi-class support vector machines”, International Conference on Smart Card Research and Advanced Applications. Springer, Berlin, Heidelberg, pp. 263-276, 2015.
[11] Heuser A, Zohner M. “Intelligent machine homicide”, International Workshop on Constructive Side-Channel Analysis and Secure Design. Springer, Berlin, Heidelberg, pp. 249-264, 2012.
[12] Prouff, Emmanuel, et al. “Study of Deep Learning Techniques for Side-Channel Analysis and Introduction to ASCAD Database.” IACR Cryptology ePrint Archive, vol. 53, 2018.
[13] Martinasek, Zdenek, and Vaclav Zeman. “Innovative method of the power analysis.” Radio engineering, vol. 22.2, pp. 586-594, 2013.
[14] Maghrebi, Houssem, Thibault Portigliatti, a nd Emmanuel Prouff. “Breaking cryptographic implementations using deep learning techniques.” International Conference on Security, Privacy, and Applied Cryptography Engineering. Springer, Cham, 2016.
[15] Cagli, Eleonora, Cécile Dumas, and Emmanuel Prouff. “Convolutional neural networks with data augmentation against jitter-based countermeasures.” International Conference on Cryptographic Hardware and Embedded Systems. Springer, Cham, 2017.
[16] Picek, Stjepan, 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, 2018.
[17] Robyns, Pieter, Peter Quax, and Wim Lamotte. “Improving CEMA using correlation optimization.” IACR Transactions on Cryptographic Hardware and Embedded Systems, vol. 2019, no. 1, pp. 1-24, 2018. Doi: 10.13154/tches.v2019.i1.1-24
[18] China’s Office of Security Commercial Code Administration: Specification of SM3 Cryptographic Hash Function, http://www.oscca.gov.cn/UpFile/20101222141857786. Pdf, 2010.
[19] Turner, James M. “The keyed-hash message authentication code (HMAC).” Federal Information Processing Standards Publication, vol.

198-1, 2008.

[20] Chari S, Rao J R, Rohatgi P. "Template attacks", International Workshop on Cryptographic Hardware and Embedded Systems. Springer, Berlin, Heidelberg, pp. 13-28, 2002.

[21] O'Shea K, Nash R. "An introduction to convolutional neural networks", arXiv preprint arXiv:1511.08458, 2015.

[22] Kocher P C. "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems", Annual International Cryptology Conference. Springer, Berlin, Heidelberg, pp. 104-113, 1996.

[23] Mayer-Sommer R. "Smartly analyzing the simplicity and the power of simple power analysis on smartcards", International Workshop on Cryptographic Hardware and Embedded Systems. Springer, Berlin, Heidelberg, pp. 78-92, 2002.

[24] Brier E, Clavier C, Olivier F. "Correlation power analysis with a leakage model", International workshop on cryptographic hardware and embedded systems. Springer, Berlin, Heidelberg, pp. 16-29, 2004.

Suying Wang

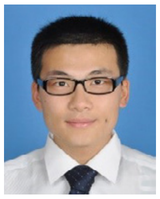
Suying Wang (1991-), She received her B.S. degree in Zhejiang University in 2014, and M.S. degree in Royal institute of Technology in 2016. She currently works at Open Security Research, Inc. Her main research interests include Side Channel Analysis and Deep Learning-based Side Channel Analysis. Email: suying.wang@osr-tech.com

Yong Xiao

Yong Xiao was born in Jingzhou, Hunan, China, in 1978. He received the Ph.D. degree from Wuhan University. He is currently a Senior Engineer. His research interests include intelligent electric power technology and measurement automation technology.

Shiqi Li

Shiqi Li (1985-), He received the B.S. and M.S. degrees in Electrical Engineering from Katholieke Universiteit Leuven, Belgium in 2007 and 2009 respectively. His main research interests include Side Channel Analysis, Fault Injection and Cache timing analysis on TEE systems. He currently works as a senior security researcher in the Security Laboratory from Open Security Research, Inc. in Shenzhen, China. Email: shiqi.li@osr-tech.com



Xin Jin

Xin Jin, He received the Master degree from the North China Electric Power University (NCEPU), Bao Ding, China, in 2011. He is currently a senior engineer of the China Southern Power Grid Research Institute and serves as the director of the Intelligent Measurement and New Technology Laboratory. His main research areas are power line carrier communication and wireless communication technology. Email: jinxin1@csg.cn