GAP: Born to Break Hiding

Ju-Hwan Kim, Ji-Eun Woo, Soo-Jin Kim, So-Yeon Park and Dong-Guk Han

Univ Kookmin, Seoul, Republic of Korea,
{zzzz2605,dnwldms928,suzin22,soyeonp,christa}@kookmin.ac.kr

Abstract. Recently, Machine Learning (ML) is widely investigated in the side-channel analysis (SCA) community. As an artificial neural network can extract the feature without preprocessing, ML-based SCA methods relatively less rely on the attacker's ability. Consequently, they outperform traditional methods.
Hiding is a countermeasure against SCA that randomizes the moments of manipulating sensitive data. Since hiding could disturb the neural network's learning, an attacker should design a proper architecture against hiding. In this paper, we propose inherently robust architecture against every kind of desynchronization. We demonstrated the proposed method with plenty of datasets, including open datasets. As a result, our method outperforms state-of-the-art on every dataset.

Keywords: Side-Channel Analysis · Deep Learning · Global Average Pooling · Desynchronization · Hiding

1 Introduction

Side-Channel Analysis (SCA) is a class of cryptanalysis that reveals secret information via physical leakage of a cryptographic device, such as power consumption [KJJ99], electromagnetic [GMO01], acoustic [GST14], or photon [FH08]. The fundamental fact of SCA is that the leakage is related to the manipulated data. An attacker discloses the secret key by exploiting the relationship between the leakage and the sensitive intermediate value. For that, the attacker should firstly characterize the time when sensitive information leaks (also known as points of interest (PoI)). Hiding is a countermeasure against SCA that randomizes the moments of manipulating the sensitive intermediate value, for instance, shuffling [CK09] or random insertion of dummy operations [VMKS12]. As the traditional SCA methods, such as Differential Power Analysis (DPA) [KJJ99] or Template Attack (TA) [CRR02], require a leakage concentration, the attacker should realign the trace by proper preprocessing technique, like elastic alignment [vWWB11] or Fourier transform [MG10]. Consequently, traditional SCAs highly rely on the attacker's ability.

Recently, Machine Learning (ML) is widely applied to SCA [HGM+11,LBM14,LPMS18, Tim19]. As an artificial neural network can automatically characterize PoI, unlike the traditional SCAs, preprocessing is not mandatory. Instead of preprocessing, the attacker should optimize hyperparameters, which determine the neural network model structure and learning process structure. Significantly, the attacker should design a proper model architecture to the characteristics of the data. For instance, convolutional neural network (CNN) is robust to desynchronized trace because the hidden layer of CNN is translational equivariance [CDP17]. Zaid *et al.* proposed the method to design efficient CNN architecture by the data characteristics: the number of PoI and the maximum amplitude of desynchronization [ZBHV20]. Their method outperformed existing works, but it is hard to apply universally because the attacker might not analyze the data characteristics due to countermeasures. To the best of our knowledge, the model design method against all of the hiding countermeasures does not exist; state-of-the-arts focused on desynchronization.

Contributions. In this paper, we present the most efficient neural network architecture against hiding, *i.e.*, random delay, random insertion of dummy operations, and shuffling. We design the neural network to identically operate regardless of the moment of leaking sensitive information by replacing the fully-connected layer of CNN with Global Average Pooling (GAP) proposed by Lin *et al.* [LCY14]. To demonstrate our method can extract the secret information regardless of the existence of hiding countermeasure, we applied the proposed method to the main public datasets and our CW-Lite (ChipWhisperer-Lite) [Inc] datasets with strong hiding countermeasures. As a result, our method significantly outperforms state-of-the-art [ZBHV20] even though we did not exhaustively search hyperparameter: we just use a single hyperparameter for every dataset.

Organization. The rest of this paper is organized as follows. Section 2 briefly describes the CNN and the translational equivariance property of the convolutional layer. Section 3 compares the operations of the fully-connected layer and the GAP to present the reason that GAP makes the neural network robust to the hiding countermeasures. In Section 4, to demonstrate the proposed network can break hiding countermeasures, we apply the proposed neural network to the open datasets and our CW-Lite datasets and compare it with the state-of-the-art. Finally, Section 5 concludes the paper.

2 Preliminaries

2.1 Machine Learning-based Profiled Side-Channel Attacks

A profiled side-channel attack is a powerful SCA method that priory profiles the relationship between leakage of a cryptographic device and sensitive information. It assumes that the attacker can access the profiling device, which is identical to the target device and can be fully controlled by the attacker. Profiled SCA can be divided into two phases: profiling step and attack step. Firstly, the attacker generates a profile of the device. ML-based profiled SCA generates the profile via a neural network: the attacker trains the neural network to model the relationship between the physical leakage and the sensitive intermediate value. The attacker trains the network treating the leakage as an input (feature in the ML terms) and the intermediate value as an output (*i.e.*, label). After creating the profile, the attacker can calculate the probability of each intermediate value via the trained neural. Consequently, the attacker calculates the probability of each key hypothesis by combining the intermediate value and plaintext (ciphertext).

Guessing Entropy (GE), which is commonly used metric to measure the performance of an attack, is defined as average rank of the right key [SMY09]. That is, zero GE indicates that the attacker disclosed the secret key for every test. The attacker can evaluate the attack performance by investigating the trend of GE according to the number of traces; the attacker can estimate the minimum number of traces to disclosure (MTD). Let us denote MTD the minimum number of traces to disclose the secret key for every attack. That is, the guessing entropy is constant zero if the number of attack traces is greater than MTD. In the rest of this paper, we use MTD as a metric to compare the models' performance.

2.2 Convolutional Neural Network

Convolutional Neural Network (CNN) is generally used architecture for image classification problems [LBBH98]. Generally, CNN consists of three components: convolutional layer, pooling layer, Fully-Connected (FC) layer. Conventional CNN extracts the feature at the hidden layer, composed of convolutional layers and pooling layers, and decides a class by FC layers.

The convolutional layer extracts the feature by convolution, which is a sum of element-wise multiplication between kernel and feature. For instance, the output's first element of the left-side of Figure 1 is calculated by $0 \times 1 + 0.3 \times 2 + 0.2 \times 3 = 1.2$. As the convolutional layer shares the weight, it ignores a location of data; consequently, it has a translational equivariance property. Figure 1 shows the example of data is moved one point. Note that the output's internal elements of the right-side are moved one point as the input's elements are moved one point. Therefore, the convolutional layer can extract the feature regardless of the position of the important feature.

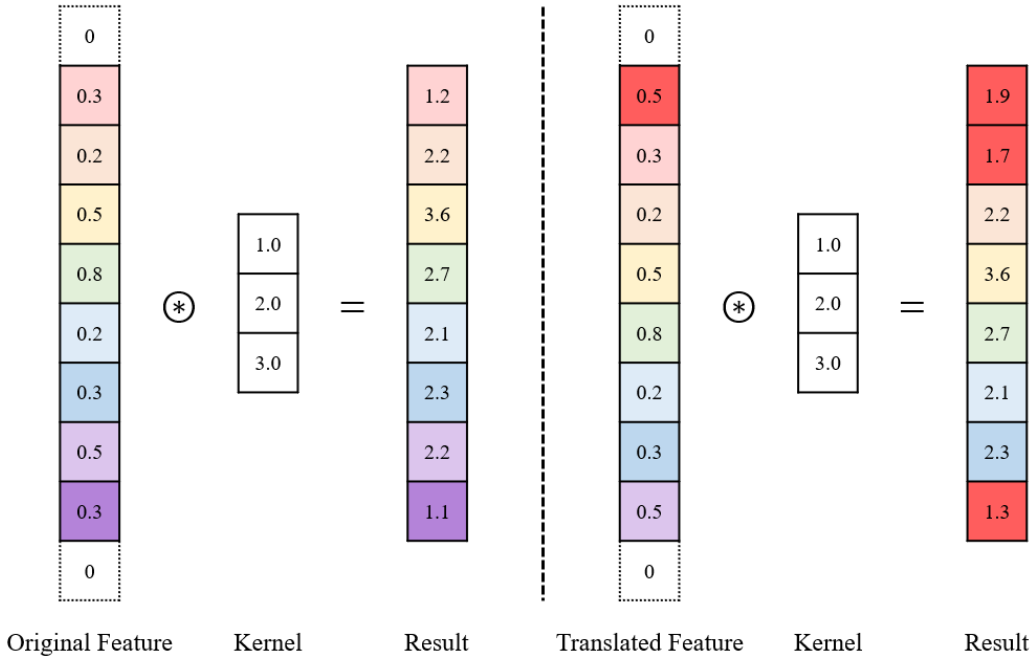


Figure 1: Example of the translational equivariance property

The pooling layer compresses the data after the convolutional layer by replacing part of the data with a statistic, such as maximum (max pooling) or average (average pooling). As the pooling layer reduces the dimension of the feature, it makes the model faster and lighter. Moreover, it can reduce noise, such as data translation [**?**].

FC layer is an identical architecture to the Multi-Layer Perceptron (MLP) [MP87]. As FC layer has a element-wise weight, unlike convolutional layer, this layer makes the neural network sensitive to data translation.

In this paper, we propose the model building methodology against hiding countermeasure by removing the FC layer.

3 GAP: Most Efficient Way to Break Hiding

As we explained in Section 2.2, the FC layer makes the CNN sensitive to data translation, while the convolutional layer and pooling layer makes the network robust to data translation. To design the network that can break hiding countermeasure, we replace the FC layer with the Global Average Pooling (GAP), whose output is defined as the average of each feature map, as shown in Figure 2.

Note that the GAP layer does not consider the position of the feature, unlike the FC layer; consequently, GAP is robust to data translation. In fact, the output of GAP is only affected by newly inserted elements and removed elements, not the position when data is

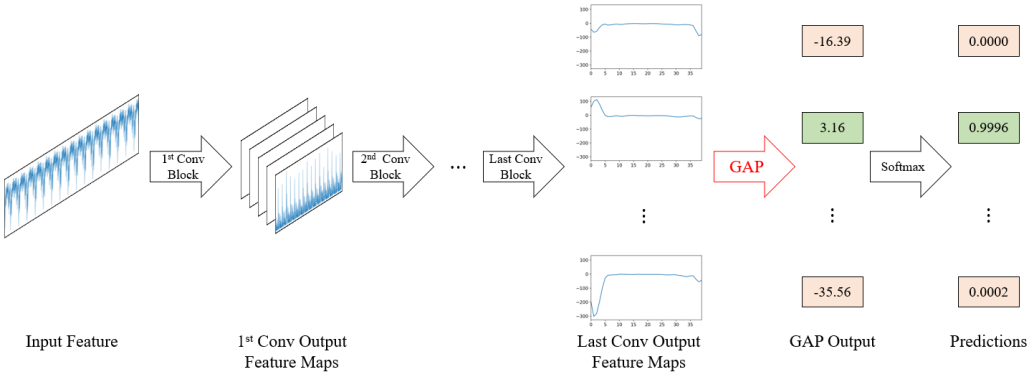


Figure 2: Architecture of the Proposed CNN against Hiding

moved. Let us denote D the feature whose dimension is n, D_i the i-th point of the D, TD p point translated data ($TD_i = D_{i+p}$), and $W_{i,j}$ weight of the FC layer. The difference between outputs of GAP when D is moved p point is as follow:

$$\begin{aligned}
&\mathrm{GAP}\,(D) - \mathrm{GAP}\,(TD) \\
&= \frac{1}{n}\sum_{i=0}^{n-1} D_i - \frac{1}{n}\sum_{i=0}^{n-1} TD_i \\
&= \frac{1}{n}\left(\sum_{i=0}^{p-1} D_i + \sum_{i=p}^{n-1} D_i\right) - \frac{1}{n}\left(\sum_{i=0}^{n-p-1} TD_i + \sum_{i=n-p}^{n-1} TD_i\right) \\
&= \frac{1}{n}\sum_{i=0}^{p-1} D_i + \frac{1}{n}\sum_{i=0}^{n-p-1} D_{i+p} - \frac{1}{n}\sum_{i=0}^{n-p-1} TD_i - \frac{1}{n}\sum_{i=n-p}^{n-1} TD_i \\
&= \frac{1}{n}\sum_{i=0}^{p-1} D_i + \frac{1}{n}\sum_{i=0}^{n-p-1} TD_i - \frac{1}{n}\sum_{i=0}^{n-p-1} TD_i - \frac{1}{n}\sum_{i=n-p}^{n-1} TD_i \\
&= \frac{1}{n}\sum_{i=0}^{p-1} D_i - \frac{1}{n}\sum_{i=n-p}^{n-1} TD_i.
\end{aligned} \tag{1}$$

As shown in Equation 1, when data is moved by p point, GAP's output is slightly different because of the newly inserted data $TD_{n-p}, \cdots, TD_{n-1}$ and removed data $D_0, \cdots, D_{p-1}$. On the other hand, the FC layer's output can be significantly changed as the feature is moved:

$$\begin{aligned}
&\mathrm{FC}\,(D)_j - \mathrm{FC}\,(TD)_j \\
&= \frac{1}{n}\sum_{i=0}^{n-1} w_{i,j} D_i - \frac{1}{n}\sum_{i=0}^{n-1} w_{i,j} TD_i \\
&= \frac{1}{n}\sum_{i=0}^{p-1} w_{i,j} D_i + \frac{1}{n}\sum_{i=0}^{n-p-1} w_{i+p,j} TD_i - \frac{1}{n}\sum_{i=0}^{n-p-1} w_{i,j} TD_i - \frac{1}{n}\sum_{i=n-p}^{n-1} w_{i,j} TD_i \\
&= \frac{1}{n}\sum_{i=0}^{p-1} w_{i,j} D_i + \frac{1}{n}\sum_{i=0}^{n-p-1} \left(w_{i+p,j} - w_{i,j}\right) TD_i - \frac{1}{n}\sum_{i=n-p}^{n-1} w_{i,j} TD_i.
\end{aligned} \tag{2}$$

Figure 3 shows the example of behavior of two layers when feature is moved one point. As the GAP layer is not affected by data's position, the difference between two output is only 0.1, while the difference between FC layer is 2.49.

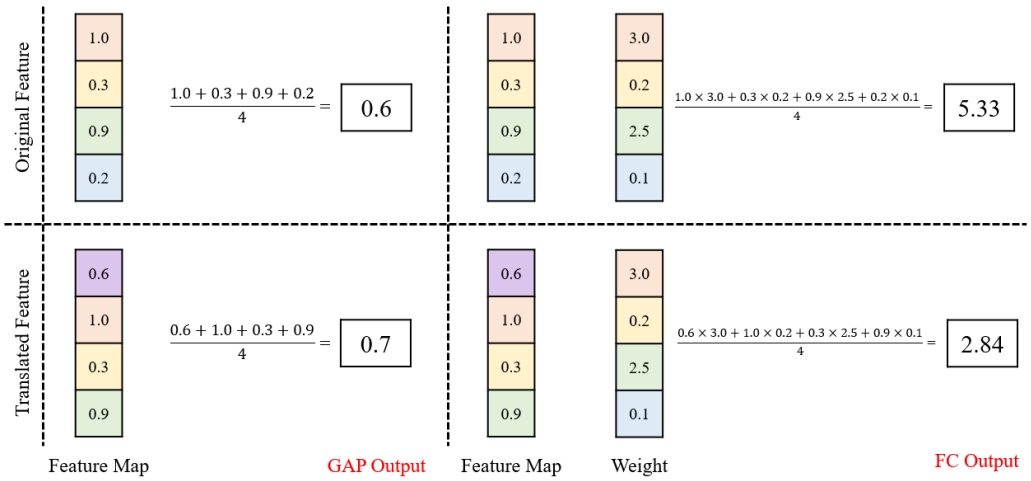


Figure 3: Comparison of GAP and FC behavior when data is translated

Recall that the hidden layer of the CNN (i.e., convolutional layer and pooling layer) is translational equivariance. Thus, replacing the FC layer with GAP allows the neural network to extract features regardless of the position of the feature. Therefore, the proposed network can break hiding countermeasures.

4 Experimental Results

In this section, we demonstrate that the proposed CNN can break hiding countermeasure. We compare the performance of our network with state-of-the-art methodology [ZBHV20] via *MTD*. We set the identical hyperparameters of the proposed method for all datasets while exhaustively searching for optimal hyperparameters for state-of-the-art, as described in Appendix A. As the open datasets are not implemented strong hiding countermeasure, we collected the data with ChipWhisperer-Lite board [Inc]. We implemented the AES [Sta02] with three hiding countermeasures: shuffling, random jitter, and insertion of dummy operation. And we applied the proposed method to the open datasets for impartial comparison.

4.1 CW-Lite Dataset

We collected power consumption when performing the first *SubBytes* transformation with a sample rate of 29.538MS/s. We recorded 50000 power consumption with variable keys for training divided into two subsets: 45000 traces for the training, 5000 traces for validation, and 10000 power consumption with fixed key for the attack. Each trace is classified with the first byte of the first SubBytes output: $\texttt{SBox}[P_0 \oplus k]$, where P_0 is the first byte of the plaintext and k is the secret key. Three hiding countermeasures are implemented:

- **Shuffling** We uniformly shuffle the order of operation. For instance, the order of substitute for each state is uniformly random in the *SubBytes* transformation. We assume that Shuffling increases the number of PoI to (# of PoI) × (# of operation) to apply state-of-the-art methodology.
- **Random Jitter (RJ)** We implemented the random jitter countermeasure based on sobsen's implementation presented at CHES Challenges 2016[1]. The random jitter

[1] https://ctf.newae.com/flags/

is inserted every before an operation, and the maximum length of the jitter is 428. Thus, we set the length of the filters and the pooling stride defining in the second block of state-of-the-art are configured as 214.

- **Insertion of Dummy Operations** We inserted the dummy operation, which performs an identical operation as an actual operation but calculates with a random value. Two types of countermeasures are implemented: a random number of dummy operations (RD) and a fixed number of dummy operations (FD). The number of dummy operations of RD is chosen from discrete uniform distribution $u\{0, 16\}$, and it of FD is fixed at 16.

As shown in Table 1 MTDs of the proposed method are 2. Note that $MTD = 1$ is practically impossible because it indicates that the neural network can correctly classify the trace for every trace (*i.e.*, the validation accuracy is 100%.): the performance cannot be more optimized in realistic. These results show that the proposed architecture can classify the trace regardless of sensitive leakage position: the proposed method can break the hiding. On the other hand, state-of-the-art success revealed the secret key with few traces when the hidden countermeasures' intensity is low, i.e., when only one countermeasure exists.

Table 1: Comparison of the minimum traces to disclosure on CW-Lite datasets

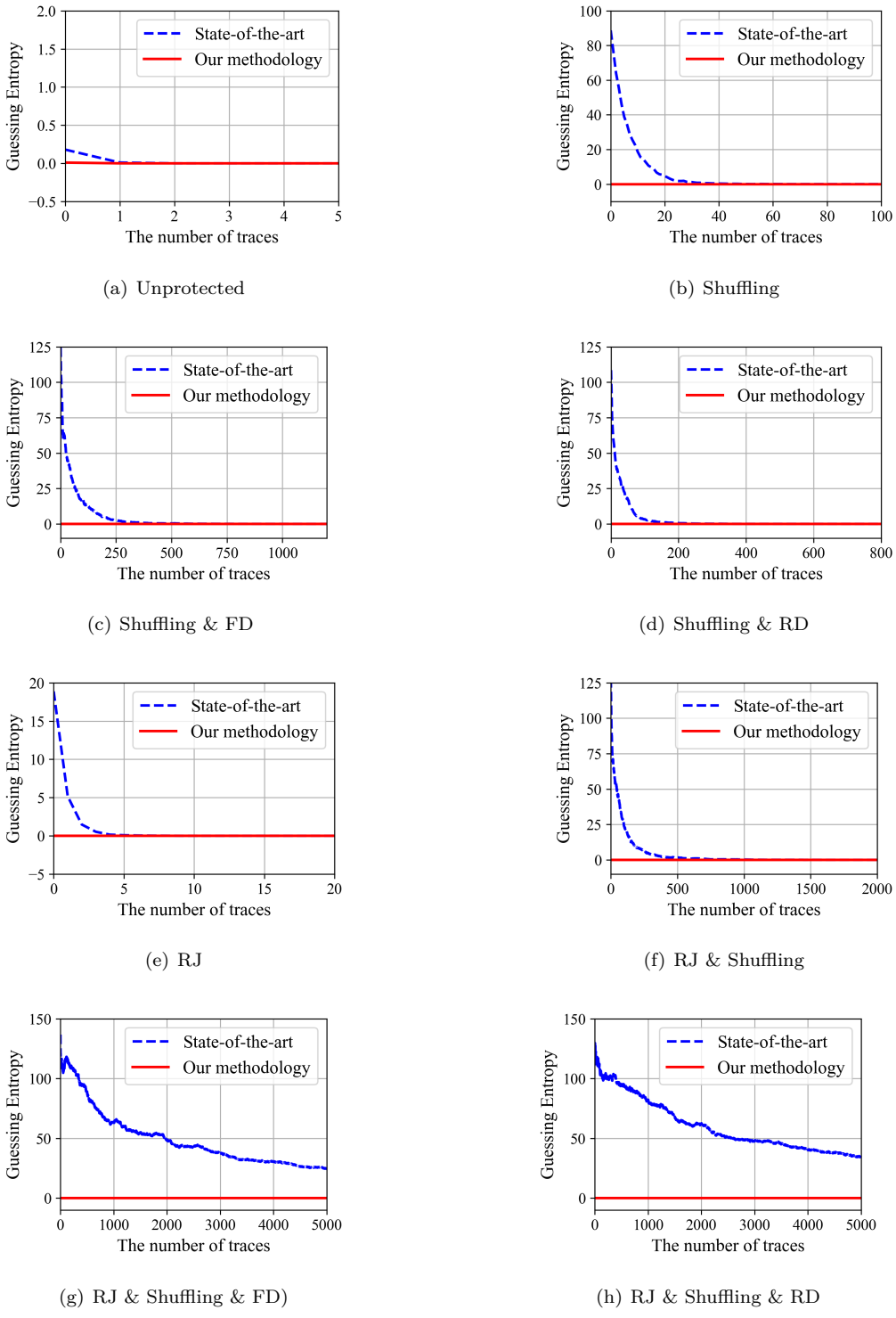
	State-of-the-art [ZBHV20]	**Our methodology**
Unprotected	3	2
Shuffling	79	2
Shuffling & FD	1085	2
Shuffling & RD	527	2
RJ	10	2
RJ & Shuffling	1713	2
RJ & Shuffling & FD	>5000	2
RJ & Shuffling & RD	>5000	2

4.2 Open Datasets

We used four commonly used datasets: ASCAD[2], DPA-contest v4[3], AES_RD[4], and AES_HD[5]. For the comparison, we used the identical number of traces and the target intermediate value to the state-of-the-art.

- **ASCAD** is obtained from an 8-bit AVR microcontroller where a masking countermeasure is implemented [PSB+18]. It contains the desynchronized trace that was manually desynchronized with a 50 or 100 samples window maximum jitter. We target the third byte of the first *SubBytes* transformation: $\texttt{SBox}\left[P_3 \oplus k\right]$.
- **DPA-contest v4** is obtained from an Atmel ATMega-163 smart-card where a masking countermeasure is implemented. As the existing works targeted the masked intermediate value, we classify each trace by the first byte of the first *SubBytes* transformation: $\texttt{SBox}\left[P_0 \oplus k\right] \oplus M$, where M is the mask.
- **AES_HD** is obtained from FPGA without any countermeasure [PHJ+19]. The target intermediate value is the difference between the last round's input and output: $\texttt{SBox}^{-1}\left[C_1 2 \oplus k\right] \oplus C_8$, where C_i is the i-th byte of the ciphertext.

[2] https://github.com/ANSSI-FR/ASCAD
[3] http://www.dpacontest.org/v4/42_traces.php
[4] https://github.com/ikizhvatov/randomdelays-traces
[5] https://github.com/AESHD/AES_HD_Dataset



(a) Unprotected

(b) Shuffling

(c) Shuffling & FD

(d) Shuffling & RD

(e) RJ

(f) RJ & Shuffling

(g) RJ & Shuffling & FD)

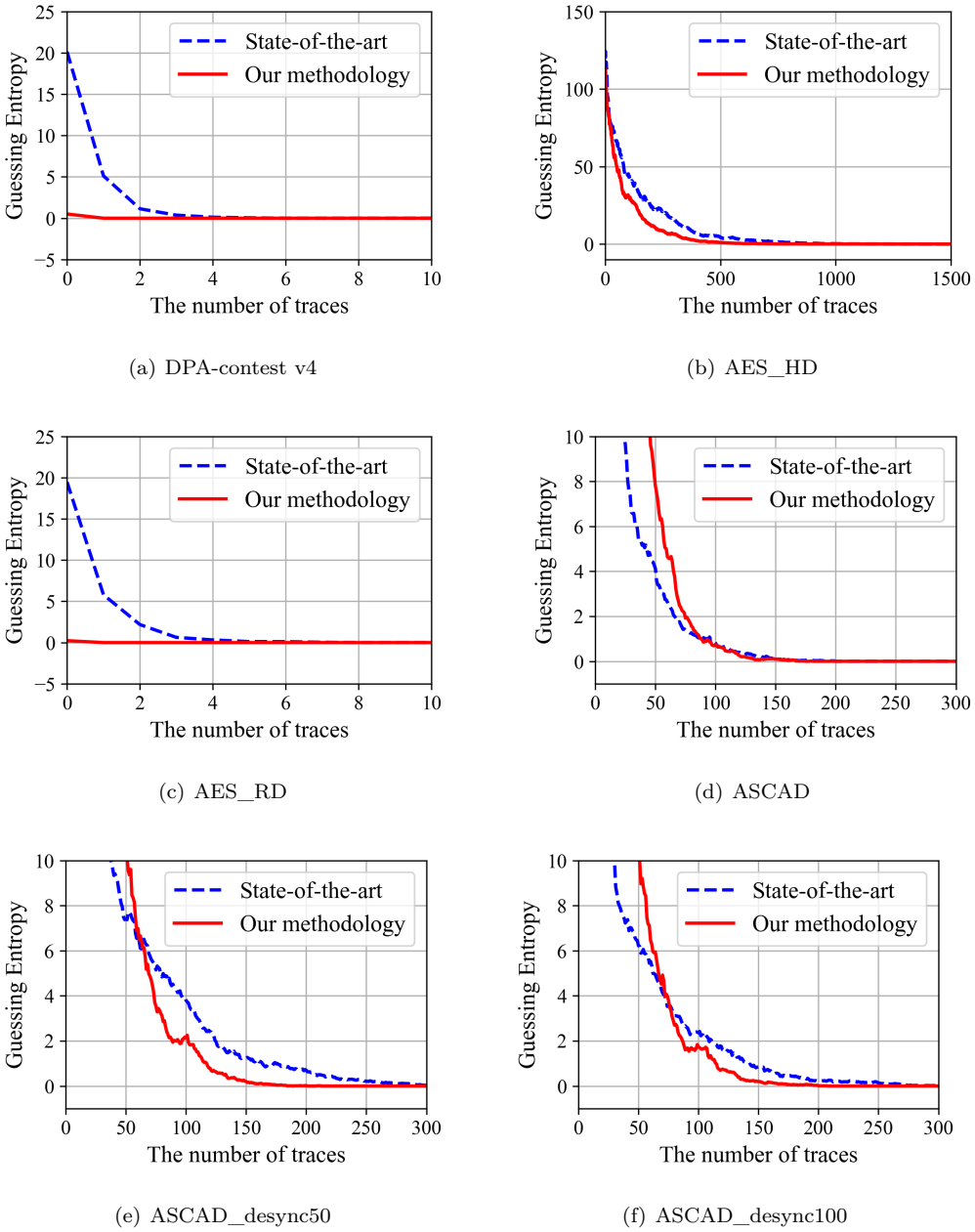
(h) RJ & Shuffling & RD

Figure 4: Guessing entropy of the CW-Lite datasets

- **AES_RD** is obtained from 8-bit AVR microcontroller where a random delay countermeasure is implemented [CK09]. We target the first byte of the first *SubBytes*

transformation: $\texttt{SBox}[P_0 \oplus k]$.

As shown in Table 2, our methodology outperforms state-of-the-art, especially when the hiding countermeasure is applied (AES_RD, ASCAD_desync50, ASCAD_desync100). Note that *MTD* of state-of-the-art on ASCAD increases as the strength of the jitter increases. On the other hand, our method is not affected by the strength. This phenomenon shows that our architecture can extract proper features regardless of the position of sensitive information.



(a) DPA-contest v4

(b) AES_HD

(c) AES_RD

(d) ASCAD

(e) ASCAD_desync50

(f) ASCAD_desync100

Figure 5: Guessing entropy of the open datasets

To search for proper hyperparameters of state-of-the-art methodology, we investigated

Table 2: Comparison of the minimum traces to disclosure on open datasets

	State-of-the-art [ZBHV20]	Our **methodology**
DPA-contest v4	3	2
AES_HD	1050	925
AES_RD	5	2
ASCAD	191	181
ASCAD_desync50	244	214
ASCAD_desync100	270	209

216 combinations for each dataset, whereas we set identical hyperparameters of the proposed method for every dataset. Nevertheless, our method significantly outperformed state-of-the-art for every dataset. As we described in the introduction, Zaid's method requires prior knowledge of the data, such as the number of PoI and the maximum amplitude of desynchronization, which is challenging when the hiding countermeasure exists. Contrastively, our method does not require any prior knowledge; moreover, the proposed architecture outperforms the existing method with a single hyperparameter. It indicates that our method allows the attacker to search optimal hyperparameter without searching a huge hyperparameter set. Consequently, our method allows the attacker to train the network much faster than the existing method, practically.

5 Conclusion

In this paper, we present the neural network architecture that inherently robust to hiding countermeasures. As our model operates regardless of the physical leakage position, it can extract sensitive information even though the hiding countermeasure exists.

We demonstrated the robustness of the proposed network with plenty of datasets, including four main SCA open datasets: ASCAD, DPA-contest v4, AES_RD, and AES_HD. Our network outperforms on every dataset even though we exhaustively searched optimal hyperparameters of the existing method for each dataset, while our method's hyperparameter is fixed for every dataset. This phenomenon indicates that the attacker can find optimal hyperparameter easily: the attacker can practically train the neural network faster.

References

[CDP17] Eleonora Cagli, Cécile Dumas, and Emmanuel Prouff. Convolutional neural networks with data augmentation against jitter-based countermeasures - profiling attacks without pre-processing. In Wieland Fischer and Naofumi Homma, editors, *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, volume 10529 of *Lecture Notes in Computer Science*, pages 45–68. Springer, 2017.

[CK09] Jean-Sébastien Coron and Ilya Kizhvatov. An efficient method for random delay generation in embedded software. In Christophe Clavier and Kris Gaj, editors, *Cryptographic Hardware and Embedded Systems - CHES 2009, 11th International Workshop, Lausanne, Switzerland, September 6-9, 2009, Proceedings*, volume 5747 of *Lecture Notes in Computer Science*, pages 156–170. Springer, 2009.

[CRR02] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic*

Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28. Springer, 2002.

[FH08] Julie Ferrigno and Martin Hlavác. When AES blinks: introducing optical side channel. *IET Information Security*, 2(3):94–98, 2008.

[GMO01] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic analysis: Concrete results. In *Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings*, number Generators, pages 251–261, 2001.

[GST14] Daniel Genkin, Adi Shamir, and Eran Tromer. RSA key extraction via low-bandwidth acoustic cryptanalysis. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, pages 444–461, 2014.

[HGM+11] Gabriel Hospodar, Benedikt Gierlichs, Elke De Mulder, Ingrid Verbauwhede, and Joos Vandewalle. Machine learning in side-channel analysis: a first study. *J. Cryptogr. Eng.*, 1(4):293–302, 2011.

[Inc] NewAE Techonology Inc. ChipWhisperer-Lite. https://rtfm.newae.com/Capture/ChipWhisperer-Lite/.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, pages 388–397, 1999.

[LBBH98] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[LBM14] Liran Lerman, Gianluca Bontempi, and Olivier Markowitch. Power analysis attack: an approach based on machine learning. *Int. J. Appl. Cryptogr.*, 3(2):97–115, 2014.

[LCY14] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[LPMS18] Liran Lerman, Romain Poussier, Olivier Markowitch, and François-Xavier Standaert. Template attacks versus machine learning revisited and the curse of dimensionality in side-channel analysis: extended version. *J. Cryptogr. Eng.*, 8(4):301–313, 2018.

[MG10] Edgar Mateos and Catherine H. Gebotys. A new correlation frequency analysis of the side channel. In *Proceedings of the 5th Workshop on Embedded Systems Security, WESS 2010, Scottsdale, AZ, USA, October 24, 2010*, page 4. ACM, 2010.

[MP87] Marvin Minsky and Seymour Papert. *Perceptrons - an introduction to computational geometry*. MIT Press, 1987.

[PHJ+19] Stjepan Picek, Annelie Heuser, Alan Jovic, Shivam Bhasin, and Francesco Regazzoni. The curse of class imbalance and conflicting metrics with machine learning for side-channel evaluations. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(1):209–237, 2019.

[PSB+18] Emmanuel Prouff, Rémi Strullu, Ryad Benadjila, Eleonora Cagli, and Cécile Dumas. Study of deep learning techniques for side-channel analysis and introduction to ASCAD database. *IACR Cryptol. ePrint Arch.*, 2018:53, 2018.

[SMY09] François-Xavier Standaert, Tal Malkin, and Moti Yung. A unified framework for the analysis of side-channel key recovery attacks. In Antoine Joux, editor, *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, volume 5479 of *Lecture Notes in Computer Science*, pages 443–461. Springer, 2009.

[Sta02] William Stallings. The advanced encryption standard. *Cryptologia*, 26(3):165–188, 2002.

[Tim19] Benjamin Timon. Non-profiled deep learning-based side-channel attacks with sensitivity analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(2):107–131, 2019.

[VMKS12] Nicolas Veyrat-Charvillon, Marcel Medwed, Stéphanie Kerckhof, and François-Xavier Standaert. Shuffling against side-channel attacks: A comprehensive study with cautionary note. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, volume 7658 of *Lecture Notes in Computer Science*, pages 740–757. Springer, 2012.

[vWWB11] Jasper G. J. van Woudenberg, Marc F. Witteman, and Bram Bakker. Improving differential power analysis by elastic alignment. In Aggelos Kiayias, editor, *Topics in Cryptology - CT-RSA 2011 - The Cryptographers' Track at the RSA Conference 2011, San Francisco, CA, USA, February 14-18, 2011. Proceedings*, volume 6558 of *Lecture Notes in Computer Science*, pages 104–119. Springer, 2011.

[ZBHV20] Gabriel Zaid, Lilian Bossuet, Amaury Habrard, and Alexandre Venelli. Methodology for efficient CNN architectures in profiling attacks. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(1):1–36, 2020.

A Configuration of neural network

Table 3: Hyperparameters

	State-of-the-art [ZBHV20]	**Our methodology**
Optimizer	*Adam*	
Activation function	SeLU	ReLU
Learning Rate	One-Cycle policy	
Batch size	{64, 256}	64
Epochs	{20, 50, 100}	50
# of kernels (Conv layers)	{2, 4, 8}	
# of neurons (FC layers)	{1, 2, 3}	
# of layers (FC layers)	{2, 10, 15, 20}	
Kernel size		9
Pooling	Average pooling	Max pooling

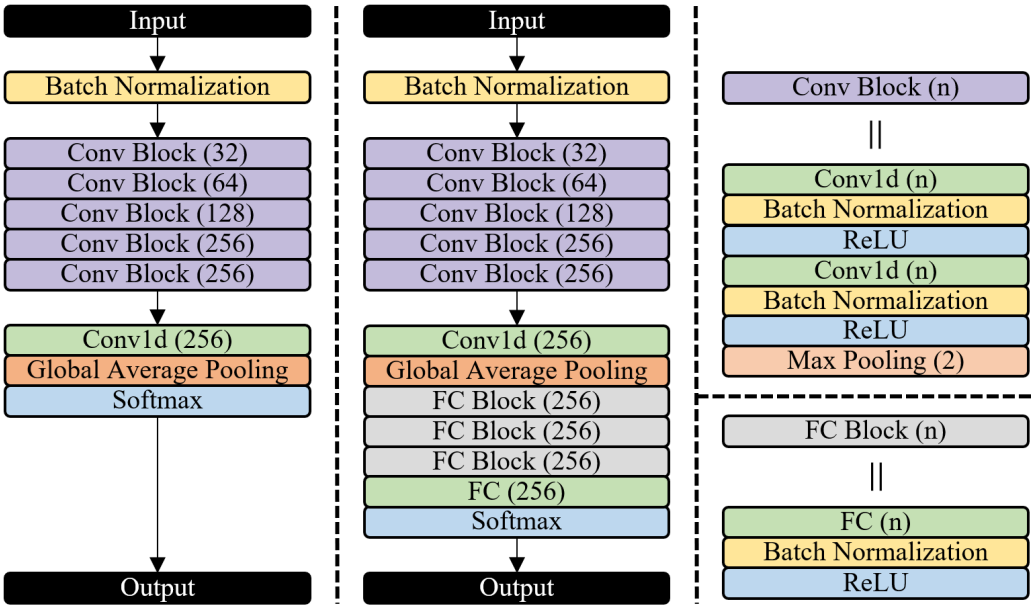


Figure 6: Configuration of the Proposed CNN against hiding (left: for unmasked trace, right: for masked trace)