Journal of Computer Science and Cybernetics, V.37, N.1 (2021), 1–22
DOI 10.15625/1813-9663/37/1/15418

EFFICIENT CNN-BASED PROFILED SIDE CHANNEL ATTACKS

NGOC QUY TRAN*, HONG QUANG NGUYEN

Academy of Cryptography Techniques

Abstract. Profiled side-channel attacks are now considered as a powerful form of side channel attacks used to break the security of cryptographic devices. A recent line of research has investigated a new profiled attack based on deep learning and many of them have used convolution neural network (CNN) as deep learning architecture for the attack. The effectiveness of the attack is greatly influenced by the CNN architecture. However, the CNN architecture used for current profiled attacks have often been based on image recognition fields, and choosing the right CNN architectures and parameters for adaption to profiled attacks is still challenging. In this paper, we propose an efficient profiled attack for unprotected and masking-protected cryptographic devices based on two CNN architectures, called CNNn, CNNd respectively. Both of CNN architecture parameters proposed in this paper are based on the property of points of interest on the power trace and further determined by the Grey Wolf Optimization (GWO) algorithm. To verify the proposed attacks, experiments were performed on a trace set collected from an Atmega8515 smart card when it performs AES-128 encryption, a DPA contest v4 dataset and the ASCAD public dataset.

Keywords. Side channel attack; Convolutional neural network; Grey wolf optimizer; Profiled attack; Points of interest.

1. INTRODUCTION

Side channel attacks (SCAs) are modern cryptanalysis techniques for revealing the secret keys of cryptographic devices by exploiting the physical signals leaked from cryptographic devices during their execution [1]. Typical SCAs contain power analysis attacks [1], timing attacks [2], electromagnetic attacks [3], or combinations of them [4]. Of these attacks, power analysis attacks have attracted the most attention from industry and academia.

Power analysis attacks were first proposed by Kocher et al. [1], who exposed the fact that the instantaneous power consumption of a cryptographic device depends on the data being processed and operations being performed. Many power analysis attack methods have sprung up since then, such as Differential Power Analysis (DPA) [1], Template Attacks (TA) [5], Correlation Power Analysis (CPA) [6], Mutual Information Analysis (MIA) [7], and Stochastic Model based Power Analysis (SMPA) [8]. From the engineering viewpoint, power analysis attacks are of two types, profiled and non-profiled attacks.

Profiled attacks play an important role in the security evaluation of cryptographic implementations [5]. Indeed, they provide a security assessment assuming the worst-case scenario.

*Corresponding author.
E-mail addresses: quytn@actvn.edu.vn (N.Q.Tran); quang27269@gmail.com (H.Q.Nguyen)

That is, the adversary has an identical cryptographic device that is almost completely controlled by him. Profiled attacks consist of two phases: a profiling phase and attack phase. During the profiling phase, the attacker analyzes the profiling device by multiple physical leakages, for example power traces, to build profiles for each possible key so that the key of the target device can be recovered when the attacking phase is performed.

1.1. Related works

Several profiled attacks have been introduced in the literature. A familiar one is the template attack proposed in [5], which is based on the Gaussian assumption. It is known to be the most powerful attack when the Gaussian assumption is verified. For the case when the Gaussian assumption is relaxed, several profiling side-channel attacks have been suggested, including techniques based on Machine Learning, and there are several papers on the application of machine learning techniques to profiled SCA attacks [24, 25, 26, 27, 28]. While different attack scenarios usually require different machine learning techniques, almost all the studies have demonstrated that Support Vector Machines (SVM) and Random Forests (RF) are good algorithms for profiled SCA attacks.

Although machine learning based profiled attacks relax the need for probability distributions of side channel leakage samples, they still require specific extraction techniques to identify points of interest (POIs) on the trace. For unprotected devices, finding POIs is quite easy based on methods such as signal-to-noise ratios (SNR), sum of squared differences (SOSD), and correlation power analysis (CPA) [29, 30]. However, for protected devices, determining POIs is a challenge for SCAs [9]. So far, no effective method has been proposed for selecting POIs for such devices. Fortunately, the deep learning method can solve the problem of modeling without extracting specific features in the pre-processing phase of traces [9, 31]. Therefore, in recent years, deep learning has begun to demonstrate its powerful efficiency in profiled SCA attacks because it almost perfectly approximates arbitrary functions.

Several studies have already investigated the performance of deep neural networks in profiled SCA attacks. Maghrebi et al. [10] first compared the SCA-efficiency of deep learning and machine learning in terms of the number of side channel traces. The work by Cagli et al. [11] evaluates the performance of convolutional neural networks (CNNs) in scenarios where power consumption traces are misaligned due to countermeasures or hardware-related effects. Their research shows that CNNs combined with data augmentation techniques can effectively suppress those misalignment effects. Prouff et al. [12] give an empirical solution to the problem of choosing hyper-parameters for CNNs and multi-layer perceptrons (MLP), and further established the power of applying deep learning to profiled SCA attacks. The other important contribution is the release of the public ASCAD dataset, which provides side-channel traces of a masked 128-bit AES implementation. The ASCAD dataset makes it easy for researchers to improve existing models or compare new deep neural network architectures.

Zaid et al. [13] highlight the importance of configuring the hyperparameters and architecture; without proper configuration, the models do not perform well. They state that when we do not comprehend the influence of a hyperparameter we cannot realize the maximum potential of an architecture. To address this problem, three methods are used, namely weight visualization, gradient visualization, and heatmapping, chosen for the explanatory ability and interpretability of their respective hyperparameters. These techniques allow for an adversary to determine the influence of each hyperparameter and facilitate hyperparameter configuration. Furthermore, they introduce methodologies for protected and unprotected implementations using the three visualization methods. Zaid's CNN architecture applies to profiled attacks on the trace dataset from unprotected AES-128 devices in DPA contest v4 and the trace data of AES-128 implementing a mask defense [12], the best attack results published to date have used only 7 to 8 traces to attack unprotected devices or about 200 traces to attack protected devices. However, Zaid's approach is based on the empirical analyses of the interpretability of convolutional neural networks and has yet not give a reason for CNN architecture for masking-protected devices. These issues will be addressed in this paper. To determine the effectiveness of our method, the results of the attacks in this article will compared to those of Zaid.

1.2. Motivation and contributions

All of the above studies have focused on improving deep learning performance for SCAs. Although deep learning has the potential to be used for SCA, the issue of tuning and configuring the parameters for deep learning architecture should also be studied and applied according to the specific trace sets. In this paper, we investigate the properties of POIs in the power traces of unprotected and protected devices as well as the convolution operations used in CNN to form base for the choice of deep learning architectures for profiled attacks on both devices. The proposed architectures only identify the number of layers, and so the parameters in each layer need to be determined. To achieve improved input-output mapping capabilities of deep learning architecture for trace datasets, an evolutionary optimization technique, i.e., the Grey Wolf Optimizer, has been utilized to determine the optimal values of the CNN parameters in each layer. The main contributions of this paper are as follows:

First, we propose a profiled attack using CNN with minimum architecture on unprotected cryptographic devices.

Second, we propose a CNN architecture for profiled attack on masking-protected devices that can reach higher efficiency in terms of the number of traces than the state-of-the-art profiled attacks.

Third, we propose using GWO to select the parameters for the CNN architectures. To the best of our knowledge, this is the first study applying the evolutionary optimization method to select parameters of CNN architectures for profiled attacks.

The paper is structured as follows: Section 2 introduces the basics of profiled attacks and deep learning. Section 3 presents the method of profiled attacks using deep learning. Experiments and experimental results are presented in Section 4. The conclusions of the paper are presented in Section 5.

2. BACKGROUND

2.1. Profiled attack

For profiled SCA attacks, the adversary is assumed to have a pair of identical devices: a profiling device and a target device. In the attack scenario of our paper, the target device runs a symmetric cryptographic algorithm with a fixed secret key. The attacker has access to control the input and the key of the profiling device, so he has the ability to characterize the leaked information very precisely by applying statistical techniques. The profiled SCA attacks are performed in two phases: the profiling phase and the attack phase.

In the profiling phase, a dataset of N_p profiling traces is acquired on the profiled device. It will be seen as a realization of the random variable $S_p \triangleq \left\{(x_1, z_1), \ldots, \left(x_{N_p}, z_{N_p}\right)\right\} \sim \Pr[\boldsymbol{X} \mid Z]^{N_p}$, where all the x_i are traces corresponding to the intermediate value $z_i = \varphi(P, K)$ processing by device. Based on S_p, a model is built to characterize the side channel leakage of the cryptographic device for each hypothetical values z_i. This can be modeled as $F(X \mid Z) : x \to P(Z)$.

In the attack phase, a dataset of N_a attack traces are acquired on the target device. It will be seen as a realization of $S_a \triangleq (k^*, \{(x_1, p_1), \ldots, (x_{N_a}, p_{N_a})\})$ such that $k^* \in \mathcal{K}$, and for all $i \in [1, N_a]$, p_i $Pr[P]$ and $x_i \sim \Pr\left[\mathbf{X} \mid Z = \varphi\left(p_i, k^*\right)\right]$. After that, a prediction vector is computed for each attack trace, based on a previously built model $y_i = F(x_i)$, $\forall i \in [1, N_a]$. A score, for example the probability, is assigned to each trace for each intermediate value hypothesis z_j, with $j \in [1, |Z|]$. The j-value of y_i describes the probability of z_j according to the model when the attack trace is x_i. These scores are combined over all the attack traces to output a likelihood for each key hypothesis and the candidate with the highest likelihood is predicted to be the right key. The maximum likelihood score can be used for predicting. For every key hypothesis $k \in \mathcal{K}$, this likelihood score is defined by equation (1) and the key with the highest score is the most likely prediction

$$d_{S_a}[k] \triangleq \prod_{i=1}^{N_a} y_i\left[z_i\right] \text{ where } z_i = \varphi\left(p_i, k\right). \tag{1}$$

2.2. Deep learning

Deep learning is a branch of machine learning that has been applied to image classification, speech recognition, and other fields [14]. Machine learning usually requires manual feature engineering while CNNs learn the automatic features directly from raw data. Furthermore, the features extracted by convolutional layers are independent of their position in the data, and dense layers can identify the features related to the labeled traces. Therefore, convolutional neural networks should be robust to jitter effects from unstable clock domains or even desynchronization [11]. The common architecture of CNNs consists of two parts, namely, feature extraction and classification. The main block of a CNN is a CONV layer directly followed by an ACT layer. The former locally extracts information from the input thanks to filters and the latter increases the complexity of the learned classification function through its non-linearity. After the activation, batch normalization (BN) is used to train deep neural networks to be faster and more stable [15]. After some (CONV ∘ ACT∘ BN) blocks, a POOL layer is usually added to reduce the number of neurons. This block is repeated in the neural network until an output of reasonable size is achieved. Then, some fully connected (FC) layers are introduced in order to obtain a global result that depends on the entire input. To sum-up, a common convolutional network can be characterized by the following formula

$$IN \circ [\text{CONV} \circ \text{ACT} \circ \text{BN} \circ \text{POOL}]^{n_1} \circ [\text{FC} \circ \text{ACT}]^{n_2} \text{ o FC o Softmax} \tag{2}$$

when n_1 and n_2 are the numbers of convolution and fully connected layers.

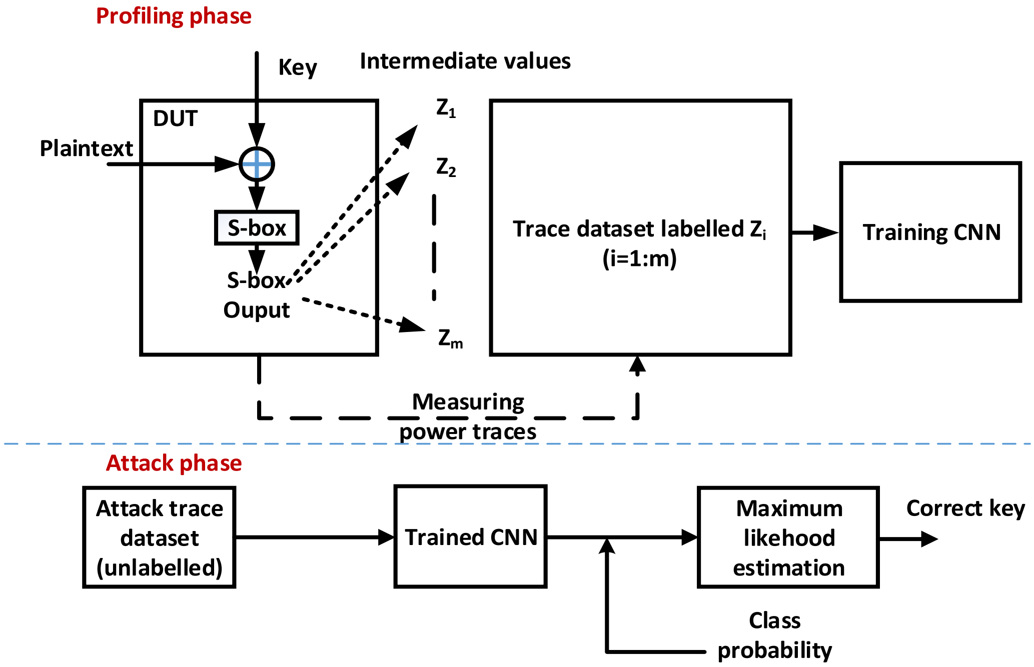

Figure 1. Basic profiled attack by CNN

3. METHODOLOGIES

3.1. CNN-based profiled attacks

The application of deep learning requires carefully analyzing the problem and configuring the neural network. The network for performing SCA attacks on cryptographic devices require at least one section for performing the function of detecting and learning the characteristics of traces and one section for performing the classification. Of the deep learning network architectures, the convolutional neural network CNN satisfies these purposes effectively. In CNN networks, the convolution layers are responsible for detecting the features of traces and the hidden neurons in the MLP network structure are responsible for classifying. Therefore, the proposed deep learning network architecture for use in profiled attacks is CNN, following the general diagram shown in Figure 1. The profiled attack using CNN in Figure 1 proceeds through two phases: a profiling phase and an attack phase. In the profiling phase, traces collected during the operation of the cryptographic algorithm are performed on the device to form a trace set. This trace set is labeled according to the intermediate value of the algorithm that needs to be profiled $Z_1, ..., Z_m$. Usually these intermediate values are taken at the output of S-box. This labeled set of traces is used to train a CNN to obtain a CNN network model describing the dependency characteristic of the intermediate value Z_i on device power consumption. During the attack phase, an unlabeled trace dataset collected from the target is classified by the trained CNN model to determine the probabilities of the traces for classes $Z_1, ..., Z_m$. These class probabilities are then associated with a key byte hypothesis in order to extract the likelihood (equation (1)) for each key byte candidate.

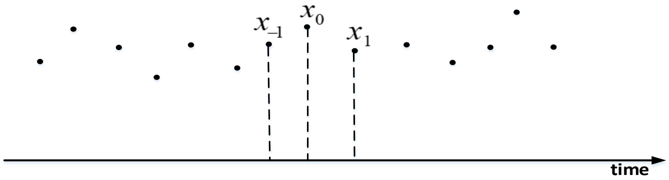

Figure 2. 1D trace signal

3.2. Kernel size of convolution layer selection

The input of the first convolution in the CNN is the trace, that is, 1D signals as illustrated in Figure 2 and formulated in the equation

$$\mathbf{x} = [\ldots, x_{-2}, x_{-1}, x_0, x_1, x_2, \ldots]\,. \tag{3}$$

A signal can be processed by filtering. This involves a convolution operation between the signal and a kernel or filter, as we shall define next. A kernel is a compact support sequence of weights

$$\omega = [\ldots, \omega_{-2}, \omega_{-1}, \omega_0, \omega_1, \omega_2, \ldots]\,.$$

The convolved signal between $\mathbf{x}$ and ω is the signal

$$\mathbf{z} = [\ldots, z_{-2}, z_{-1}, z_0, z_1, z_2, \ldots]$$

denoted by $\mathbf{z} = \mathbf{x} * \omega$, and defined by

$$z_j = \sum_{k=-\infty}^{\infty} x_{j+k}\omega_{k^*}. \tag{4}$$

The above infinite sum makes sense since w has only a finite number of nonzero elements. Each component of the convolved signal z is a weighted sum of the components of the initial signal $\mathbf{x}$. The effect of convolution is to average out a signal using a given weighting system. Equivalently, sliding the filter ω, then multiplying by $\mathbf{x}$ and summing, produces the filtered signal $\mathbf{z}$. It is worth noting that formula (4) is known in signal processing as cross-correlation. This property can be used to determine the filter size of a convolution layer in CNN. We propose the following proposition as the basis for determining the kernel size of the convolution layer in CNN.

Proposition 1. *The features of power consumption traces spread as they pass though the convolution layer.*

Proof.
Assuming the network can optimally detect features or POIs, the number of POIs is often much smaller than the length of the trace. The POIs corresponding to side channel traces are given by

$$\boldsymbol{\ell} = (\ell_{i,0}, \ell_{i,1}, \ldots, \ell_{i,d})\,,\ i = 1 : T.$$

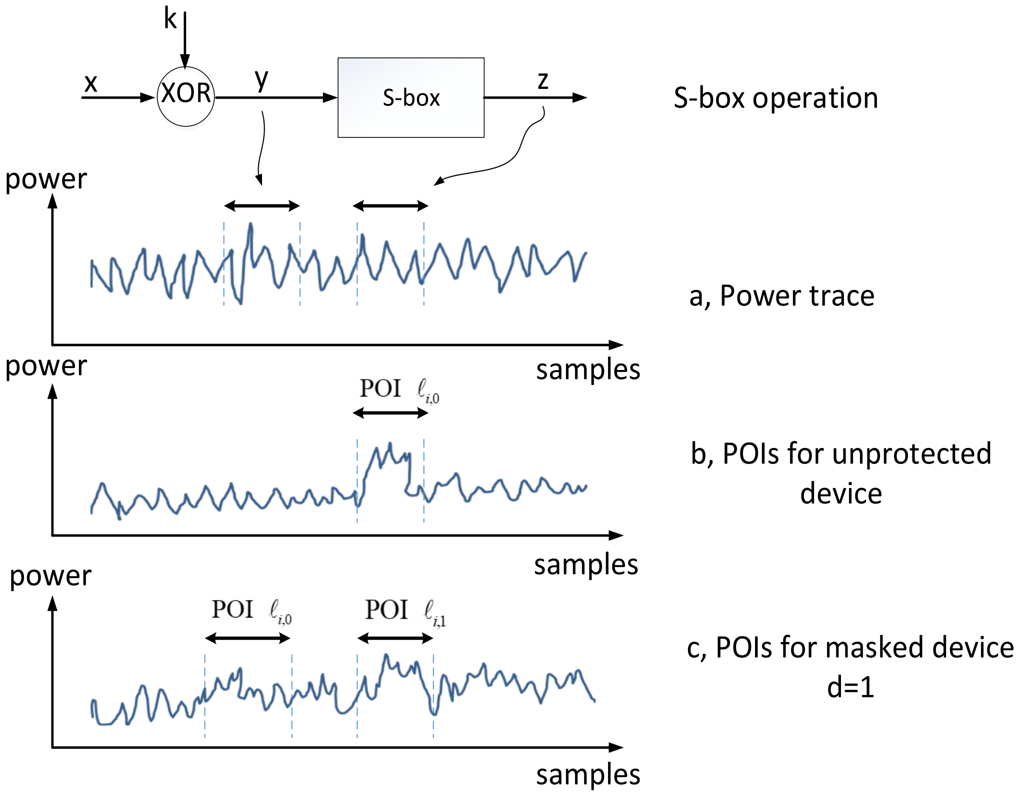

Figure 3. POIs in power trace

In the expression above $\boldsymbol{\ell}$, the sample, contains the POIs of the power consumption trace of length dT, where T is the number of samples considered as POIs when the device processes a share. Figure 3 depicts the POIs found for the traces of an unprotected device corresponding to $d = 0$ and a first-order masked device with $d = 1$.

Let $W \in \mathbb{R}^n$ be a matrix of kernel sizes of n, used by the convolution to detect POIs. Assuming this detection is optimal then

$$(\boldsymbol{x} \otimes \boldsymbol{W})[i] = \sum_{j=0}^{n} \left(\boldsymbol{x}\left[j + i - \tfrac{n}{2}\right] \times \boldsymbol{W}[j]\right),$$

such that

$$x\left[j + i - \tfrac{n}{2}\right] \times W[j] = \begin{cases} w_j \times x\left[j + i - \frac{n}{2}\right] & \text{if } j \in \ell_i \\ \epsilon \quad \text{otherwise} & \end{cases},$$

where $\epsilon \approx 0$, and w_j corresponds to the weight corresponding to the position j.

Considering a leakage point, $\ell_{i,j}$, based on the convolution operation expression, this value will affect n successive samples at its output. Therefore, if the kernel size is large, the information about the POI will be spread out and the detection of the POIs by the network is more complicated.

From Proposition 1, it can be seen that as traces of power consumption pass through the convolution layer, their characteristics will be spread according to filter size. In side channel attacks for unprotected devices require that POIs are detected by the network and for protected device, both POIs and the combination between the POIs must also be detected. Therefore, to optimize the detection of POIs, the convolution kernel size should be equal to the width of the POIs range. To detect the combination between the locations of POIs, the kernel size should be equal to the distance between the POIs bands.

3.3. CNN architecture for unprotected devices

Our proposed CNN architecture needs to perform two functions: to detect features from power consumption traces and to classify those traces into 256 labels corresponding to the intermediate values z_j. The first function is done by the pipeline of convolution and pooling blocks. Since the features in the power consumption traces are POIs. Those are valuable samples such that power consumption depends on the intermediate values. When the values are labeled and learned by the CNN network, only the POIs are used for the classification process. As shown in Figure 3b, for unprotected devices, the POIs are close to each other temporally and only exist at over a small range compared to the overall length of the trace. This POIs could be considered as the local features. As the traces pass though the convolution layer, the POIs will be spread out and the more convolution layer results in a large amount of spreading of information in POIs. More, the information in POIs is affected by the subsampling process of pooling layer that follow the convolution layer. Therefore, the number of convolution and pooling bocks is recommended set to 1 in CNN architecture for unprotected devices.

The convolution layer is responsible for detecting the POIs, and the parameters at this layer include the number of kernels γ_1 and kernel size γ_2. As shown in section 3.2, the optimal kernel size to detect POIs is equal to the width of the POIs range. For unprotected equipment, the POIs of the trace occur at a small number of locations compared to the overall length of the trace. In the best case, there would be only one location, corresponding to when the device processes the intermediate value. However, due to the high sampling rate of the measuring devices, some surrounding points will also be considered as POIs. Therefore, the recommendation of the kernel size is $\gamma_2 = 1 : 10$.

The rest of the CNN architecture is classification part that consists some fully connected layers and one output layer. In our CNN, we set number of fully connected layer of δ_1 neurons to 1 as recommended by [13]. Because, there are 256 possible intermediate values must be classified, the output layer comprises of 256 neurons with softmax activation function.

To sum up, the CNN architecture and its parameters for unprotected devices, called CNNn, are shown in Figure 4 and details are given in Table 1. The parameters γ_1, γ_2 and δ_1 are further heuristically determined by Grey Wolf Optimizer.

3.4. CNN architecture for protected devices

Attacks against the masking-protected devices are known as higher-order side channel attacks, where an attacker needs to combine independent leakage information by the operations that relate to the mask values and masked values [23]. In the power traces, the target leakage information represents in POIs. The POIs of this device usually emerge when the mask values and masked values are processed. So, there are some POIs bands and they often appear unclear and are spaced out to allow the device to process other values. For first

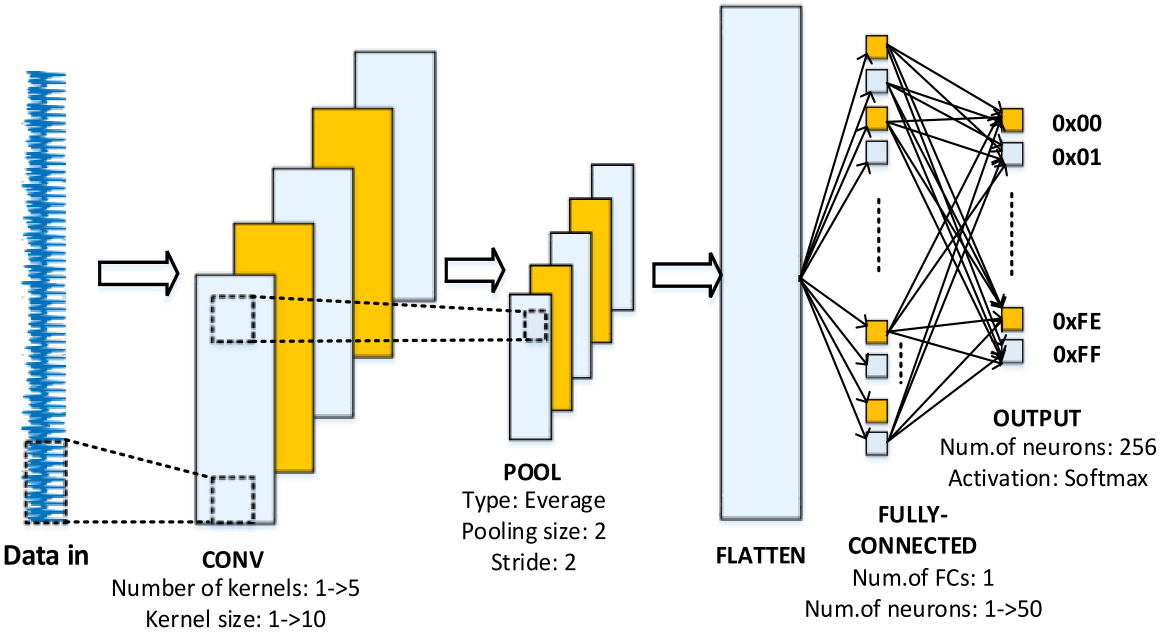

Figure 4. CNNn architecture for the unprotected devices

order masking, there are two POIs as shown in Figure 3c. In order to conduct successfully profiled attacks based on CNN, the CNN network must be able to detect POIs bands and the combination between them. For detecting POIs, we can utilize the CNNn architect for unprotected devices. The features that represent the combination between POIs bands can be considered as global features. Therefore, the based CNNn architecture should require one more convolution layer to detect these global features. As shown in Section 3.2, the kernel size of this convolution layer should be equal to the distance between the POIs bands.

To sum up, the proposed CNN architecture for the profiled attack on a masking-protected device is based on CNNn with one more convolution and pooling blocks. This modified architecture is called CNNd and its recommended parameters are described in Figure 5 and details are given in Table 1. As the same in subsection 3.3, undetermined parameters in CNNd are further selected through optimization by GWO.

3.5. Parameter optimization by GWO

There is no mature method in the literature for determining key CNN parameters. In this paper, we employ a swarm-based method called Grey Wolf Optimizer (GWO), which is one of the latest additions to a group of nature-inspired optimization heuristics, in order to determine the optimum parameters of the trained 1D CNN. The GWO is inspired by the natural leadership hierarchy and hunting mechanism of grey wolves and has demonstrated results comparable to some well-known evolutionary algorithms such as particle swarm optimization, genetic algorithms, and differential evolution [16]. The population of the GWO is divided into four hierarchies. The three most fit solutions are alpha (α), beta (β), and delta

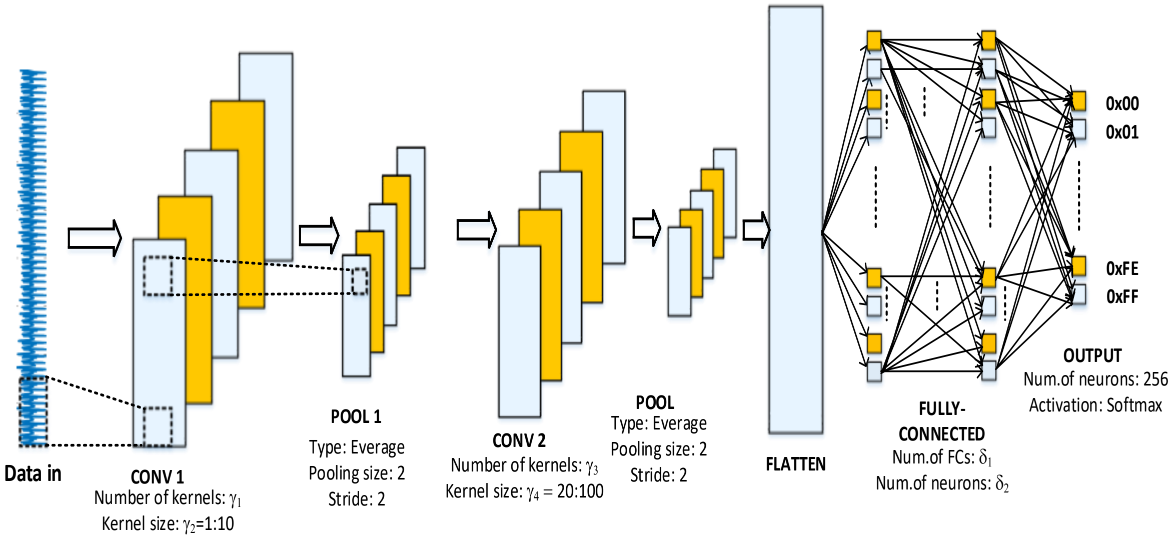

Figure 5. CNNd architecture for for the masking-protected devices

(δ), and they guide the hunting of the other omega (w) wolves. The main hunting phases include: encircling, hunting, attacking, and searching.

In circling, the wolves update their positions with respect to the prey as follows

$$\begin{aligned}&\vec{D} = \left|\vec{C} \cdot \vec{X}_p(t) - \vec{X}(t)\right|,\\ &\vec{X}(t+1) = \vec{X}_p(t) - \vec{A} \cdot \vec{D},\end{aligned} \tag{5}$$

where t is the current iteration, $\vec{X}_p$ is the position vector of the prey, and $\vec{X}$ denotes the position vector of a grey wolf. $\vec{A}$, $\vec{C}$ are coefficient vectors, $\vec{A} = 2\vec{a} \cdot \vec{r}_1 - \vec{a}$, $\vec{C} = 2 \cdot \vec{r}_2$, in which $\vec{a}$ is linearly decreased from 2 to 0 over the course of iterations, and $\vec{r}_1$, $\vec{r}_2$ are random vectors in the range $[0, 1]$.

During hunting, all wolves are obliged to update their positions according to the three best solutions obtained from encircling as follows

$$\begin{aligned}&\vec{D}_\alpha = \left|\vec{C}_1 \cdot \vec{X}_\alpha - \vec{X}\right|,\ \vec{D}_\beta = \left|\vec{C}_2 \cdot \vec{X}_\beta - \vec{X}\right|,\ \vec{D}_\delta = \left|\vec{C}_3 \cdot \vec{X}_\delta - \vec{X}\right|,\\ &\vec{X}_1 = \vec{X}_\alpha - \vec{A}_1 \cdot \vec{D}_\alpha,\\ &\vec{X}_2 = \vec{X}_\beta - \vec{A}_2 \cdot \vec{D}_\beta,\\ &\vec{X}_3 = \vec{X}_\delta - \vec{A}_3 \cdot \vec{D}_\delta,\\ &\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3},\end{aligned} \tag{6}$$

where $\overrightarrow{X_\alpha}$, $\overrightarrow{X_\beta}$ and $\overrightarrow{X_\delta}$ are the positions of alpha, beta and delta, respectively, and $\overrightarrow{D_\alpha}$, $\overrightarrow{D_\beta}$ and $\overrightarrow{D_\delta}$ are calculated using Equation (5) with different coefficient $\vec{C}$.

Attacking occurs when $|A| < 1$, and otherwise, wolves diverge from each other for searching, promoting further global exploration. The optimization procedure of CNN by GWO is presented in Figure 6.

Table 1. Architecture hyperparameters of CNNn, CNNd

CNNn		**CNNd**	
Layer	**Parameters**	**Layer**	**Parameters**
CONV	Activation function: SeLU [19] Number of kernels: $\gamma_1 = 1 : 5$ Kernel Size: $\gamma_2 = 1 : 10$	CONV1	Activation function: SeLU Number of kernels: γ_1 Kernel Size:$\gamma_2 = 1 : 10$
Batch Normalization	\	Batch Normalization	\
POOL	Pooling Size: 2; Stride: 2; Type: Average Pooling[13]	POOL 1	Pooling Size: 2; Stride: 2; Type: Average Pooling [13]
Flatten		CONV 2	Activation function: SeLU Number of kernels: γ_3 Kernel Size: $\gamma_4 = 20 : 100$
Fully Connected	Number of layers: 1 Number of neurons:$\delta_1 = 1 : 50$	Batch Normalization	\
Output	256 neurons, softmax activation	POOL2	Pooling Size: 2; Stride: 2; Type: Average Pooling
		Flatten	
		Fully Connected	Number of FCs = δ_1 Number of neurons/FC = δ_2
		Output	256 neurons, softmax activation

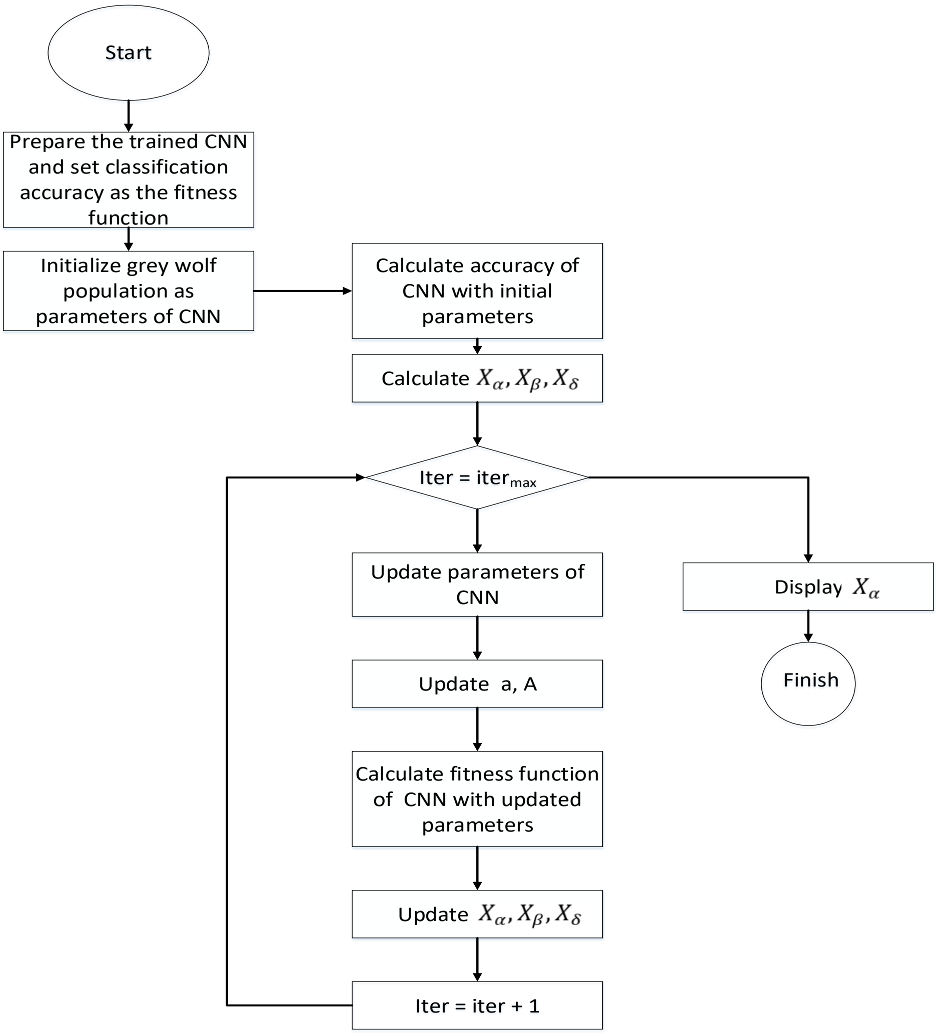
Start
Prepare the trained CNN and set classification accuracy as the fitness function
Initialize grey wolf population as parameters of CNN
Calculate accuracy of CNN with initial parameters
Calculate $X_\alpha, X_\beta, X_\delta$
Iter = iter$_{max}$
Update parameters of CNN
Update a, A
Calculate fitness function of CNN with updated parameters
Update $X_\alpha, X_\beta, X_\delta$
Iter = iter + 1
Display X_α
Finish

Figure 6. CNNn architecture for for the masking-protected devices

4. EXPERIMENTS

4.1. Score for evaluation

In this section, we present the experimental results of implementing profiled attacks based on the proposed CNN architecture in two cases: unprotected and first order masking-protected AES-128. We compared the effectiveness of the proposed method to that of the state-of-the-art published by Zaid et al. [13]. The parameters used to evaluate effectiveness are as follows:

• **The ability to reveal the correct key**. To confirm that our profiled attacks can reveal the correct key used by AES-128, we figure out the probability of the correct key over all keys. The key with highest probability is the best one.

• **The guessing entropy (GE)** [32]. This is widely used to evaluate the effects of attacks in multi-trace experimental scenarios. When using maximum likelihood estimation to recover the secret key, we pay more attention to the final probability output of each side channel trace. The output probability of each key candidate is ranked in descending order. The guessing entropy is then defined as the index or real key's rank within the sorted probabilities. We care about the amount of traces that are required to achieve a guessing entropy of zero, that is, the amount of traces required to recover the key. We estimate such a guessing entropy after 10 independent attacks.

• **The Complexity of the CNN architecture**. This parameter is the number of trainable parameters during CNN training.

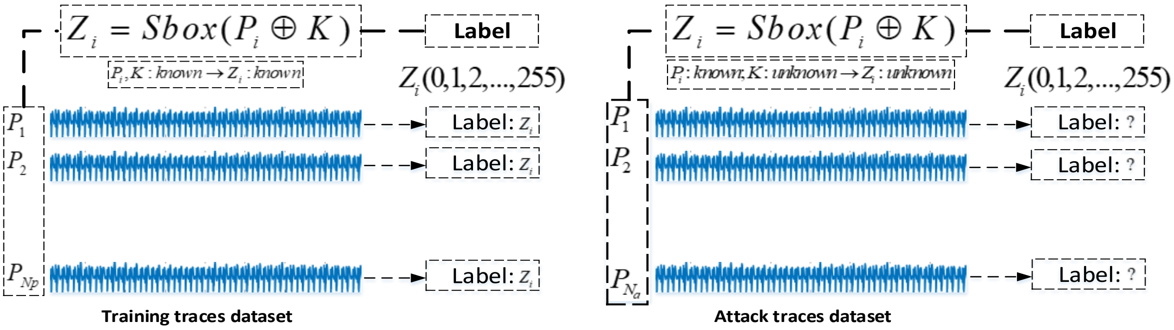

Figure 7. Example of a structure of traces dataset

4.2. Dataset

The dataset for profiled attack contains power consumption traces which is divided into training traces dataset and attack trace dataset as shown in Figure 7 for example. These traces are used as input of CNN. The labels of traces can be viewed as output values of the Sbox and known with training traces. The labels of attack traces are further predicted by CNN in key recovery phase. In this paper, we work on three datasets as follows:

- **DataSet1**. 60000 traces are collected while AES-128 processes the intermediate value at the S-box output. AES-128 was implemented on Smartcard Atmega8515 running on Sakura G/W.

- **DataSet2**. This is DPA contest v4 dataset. The set consists of 100000 traces, each consisting of 4000 features, of a masked AES implementation. However, the traces leak first-order data and this dataset is only used as an unprotected dataset after unmasking the S-box output. The targeted sensitive variable is the output of S-box, $\text{sbox}\,(P_i + k^*) \oplus M$, where M is the known mask. This dataset is publicly available at http://www.dpacontest.org/v4.

- **DataSet3**. This is the ASCAD dataset presented in [12]. The dataset is set up like the MNIST dataset and has 50000 profiling traces, and 10000 attack traces. The traces are recovered from an 8-bit AVR microcontroller from a masked implementation of AES-128. The traces were captured from electromagnetic emanations. The dataset consists of raw traces comprising measurements covering the entire encryption process. From this, the authors have pre-selected a window in the raw traces that corresponds to the S-box operation of third subkey and consists of 700 features. This part of the dataset is used for our experiments. This dataset is publicly available at https://github.com/ANSSI-FR/ASCAD.

4.3. Experimental procedure

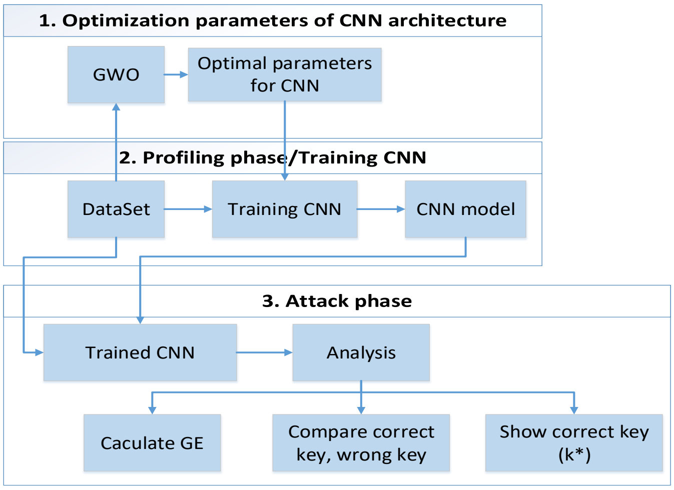
1. Optimization parameters of CNN architecture
GWO
Optimal parameters for CNN
2. Profiling phase/Training CNN
DataSet
Training CNN
CNN model
3. Attack phase
Trained CNN
Analysis
Caculate GE
Compare correct key, wrong key
Show correct key (k*)

Figure 8. Experiments procedure

We implement each experiment though 3 phases as shown in Figure 8, namely, the CNN parameters optimization phase, the profiled phase, and the attack phase. The CNNs are implemented in Python language using the Keras library [17]. During the optimization of the CNNs the MSE loss function is used with the Adam optimization method [14] and a batch size of 50. To choose the appropriate learning speed, the learning rate was selected according to One-Cycle Policy strategies [18]. Using this speed-adjustment strategy allows the use of a large learning rate while avoiding overfitting. The SeLU activation function is used to avoid vanishing and exploding gradient problems [19]. To improve weight initialization, we use He Uniform initialization [20]. All the experiments in this section use the above initialization with the number of epochs initially set at 50.

4.4. Results with unprotected device

Experiment with DataSet1

In this experiment, we use 4000 traces in the parameter optimization and training phases and 500 traces in the attack phase. GWO was employed to optimize the three key parameters, the number of kernels γ_1, the kernel size γ_2, and the number of neurons in the FC layer n. The classification accuracy was set as the fitness function. The CNN parameter setting with GWO is shown in Table 2.

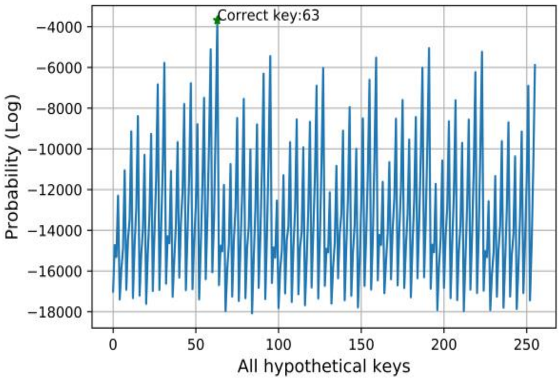

Figure 9. Estimation probability of all hypothetical keys with Dataset1

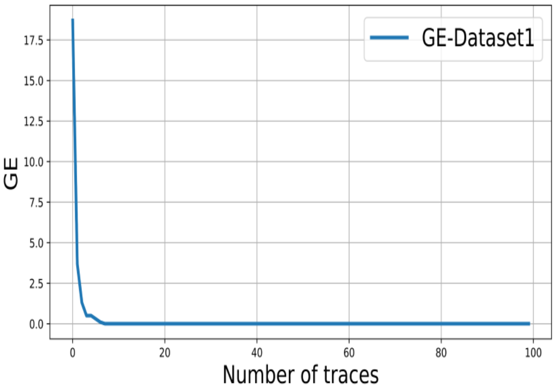

Figure 10. Guessing entropy results for Dataset1

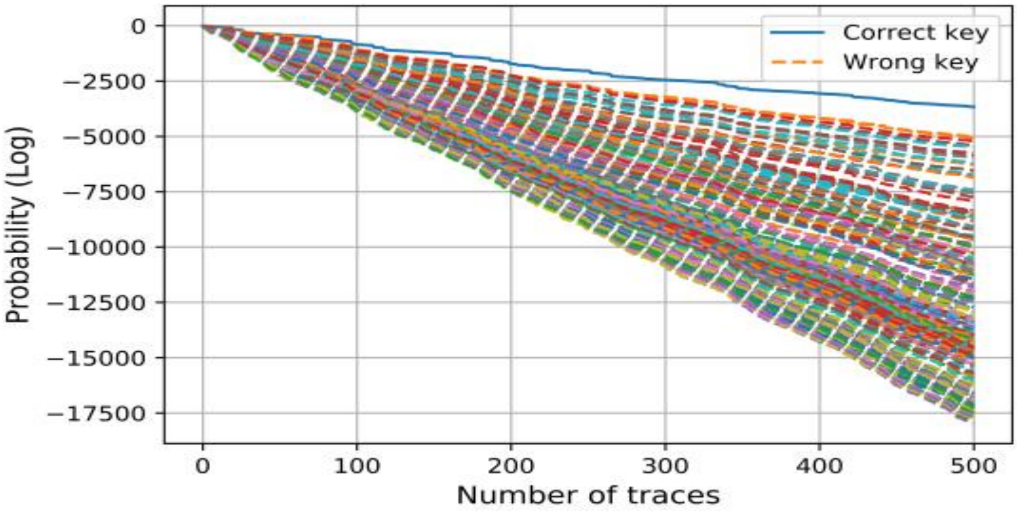

Figure 11. Estimation probability of all hypothetical keys against number of traces with Dataset1

In the attack phase, the estimated probability of the hypothetical keys is determined by the maximum likelihood estimation. The correct key is defined as the key with the highest probability. Figure 7 shows the trace set collected in this experiment on the first key byte of AES-128, with 63 having the largest probability value. Figure 9 and Figure 10 respectively describe the average rank of the correct key and the estimated probability of the keys according to the number of traces used for the attack. In general, as the number of attack traces increases, the estimation probability of the correct key compared to the estimation probability of the wrong key becomes clearly distinguishable. The wrong keys give probability estimates quite similar and indistinguishable from each other. As shown in Figure 11, the average GE value of the correct key reaches 0 very quickly after only 3 to 4 trace attacks.

Table 2. CNNn parameters selected by GWO with Dataset1

Parameter	**Input values of GWO**	**Value after GWO**
Number of kernels γ_1	1:10	4
Kernel size γ_2	1:10	3
Number of neurons in FC (n)	1:50	10

Experiment with DataSet2

The purpose of this experiment is to verify the proposed attack method against different datasets and compare its performance with the currently considered state-of-the-art results described by Zaid et al. The results of using 4000 traces in the optimization phase to find the parameters for CNNn are shown in Table 4. The CNNn with the parameters given in Table 3 is trained with the 4,000-trace dataset above and then used in the attack phase to find the correct key. The estimated probability of the keys given by Figure 12 shows that the correct key value is 130 with the first byte of the key used in AES-128 having the highest estimated probability. The large distinction between the estimated probability of the correct key and the estimated probability of the other keys reflects this dataset being easy to attack. This result is consistent with the claims made in [22]. Table 4 compares the effectiveness of the proposed method with that of Zaid. The GE values obtained by attacks using both methods are shown in Figure 13. Our CNNn architecture is more effective in terms of the number of traces required for GE to reach 0. Our method requires only 2 traces to reach 0 while Zaid's method requires 7 traces. This result demonstrates that our CNNn architecture can learn POIs from power traces more precisely than the CNN architecture proposed by Zaid. However, the number of trainable parameters and the training time is more for CNNn than the proposed method of Zaid. Neither of these CNN architectures is too complicated, but they do have good offensive results. Therefore, for unprotected devices, the CNN architecture does not need to be very complicated; only one convolution layer with a small number of kernels and a small kernel size and one FC layer containing relatively few neurons is needed.

In this section, we present two experiments of profiled attacks on unprotected devices using our dataset and the DPAContestV4 dataset. Both of them use the same CNN architecture. Although, according to the theorem "No Free Lunch" [21], there is no optimal architecture for all problems, according to the analysis as well as the experimental results with the two datasets, the CNNn architecture with the parameters given in Table 5 should be used for profiling with unprotected devices.

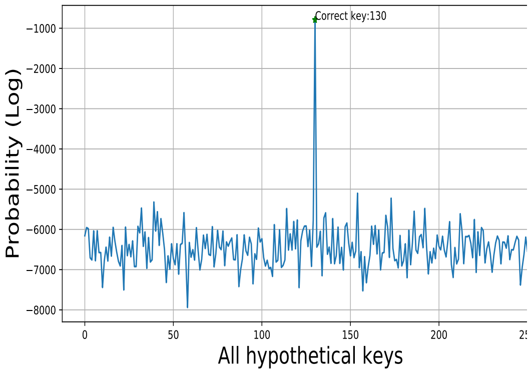

Figure 12. Estimation probability of all hypothetical keys with Dataset2

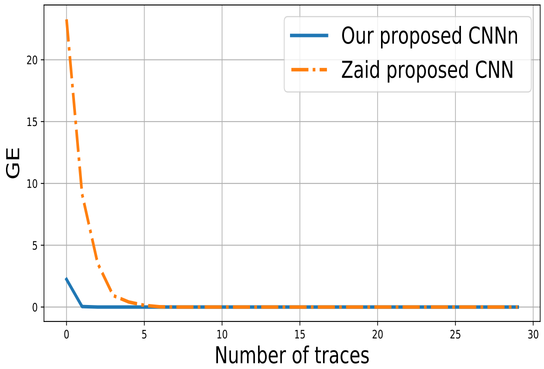

Figure 13. Guessing entropy results for Dataset2

Table 3. CNNn parameters selected by GWO with Dataset2

Parameter	**Input values of GWO**	**Value after GWO**
Number of kernels:γ_1	1:10	4
Kernel size:γ_2	1:10	3
Number of neurons in FC (n)	1:50	10

Table 4. Comparison of performance on Dataset2

	Template Attack [22]	Zaid et al. method [13]	Our proposal
Trainable parameters	\	8.782	8.858
Number of traces for GE=0	3	7	2
Training time (s)	\	103	158

Table 5. Optimal parameters of CNNn for unprotected devices

Layer	**Parameter**
CONV	One layer, Activation function SeLU Number of kernels $\gamma_1 = 4$ Kernel size $\gamma_2 = 3$
Batch Normalization	\

POOL	Pooling size = 2;
	Stride = 2

Flatten	\
Fully Connected	Number of layers =1 Number of neurons $\delta_1 = 10$
Output	256 neurons, *softmax* activation

4.5. Results on masking - protected device

The experiment in this section uses the DataSet3 dataset, which is divided into 3 parts: 45000 traces for training, 5000 traces for validation, and 10000 traces for the attack. The training and validation data are used by the GWO optimization algorithm to find optimal parameters for the CNNd architecture. The basic CNN architecture used in the experiments in this section is the CNNd proposed in Section 3.4.

The parameters optimized for CNNd given in Table 6 are generated by the GWO algorithm. Next, CNNd is trained to create a model for data traces. The probabilities of the 256 hypothetical keys estimated in the attack phase are presented in Figure 14, and it is apparent that the maximum probability value corresponds to that of key 224, which is the actual AES-128 key used. Although the probability difference between the right and wrong keys is not substantial, as Figure 15 shows, when the number of attack traces is increased, the probabilities of the wrong keys remain the same, while that of the correct key significantly increases. Table 7 compares the efficiency of the proposed method to that of Zaid [13] and Prouff [12]. The GE value obtained by attacking using both methods is shown in Figure 16. The CNNd method proposed by us is more effective in terms of the number of traces required to achieve a GE of 0. Our method requires about 183 traces to reach $GE = 0$ while Zaid's method requires 195 traces, which represents an approximately 5% reduction. This result demonstrates that our CNNn architecture can learn POIs from the power traces of masked devices more precisely than the CNN architectures proposed by either Zaid or Prouff. However, the number of trainable parameters and the training time of the proposed CNN are larger than those of the method proposed by Zaid yet much smaller than those of the method proposed by Prouff. This can be explained by the architecture of CNNd being more complex than that the architecture used by Zaid yet much simpler than that used by Prouff.

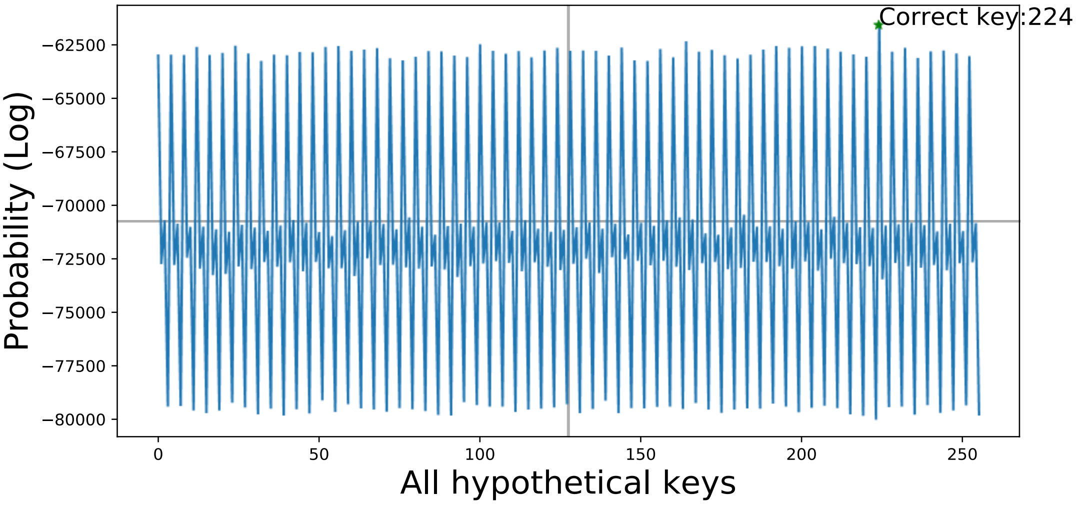

Figure 14. Estimation probability of all hypothetical keys with Dataset3

Table 6. CNNd parameters selected by GWO with Dataset3

Parameter	**Input values of GWO**	**Value after GWO**
CONV 1		
Number of kernels: γ_1	1-10	4
Kernel size: γ_2	1-10	3
Batch Normalization	\	\
POOL 1		
Pooling size	2	2
Stride	2	2
CONV 2		
Number of kernels: γ_3	1-20	8
Kernel size: γ_4	20-100	51
Batch Normalization	\	\
POOL 2		
Pooling size	2	2
Stride	2	2
Flatten	\	\
Fully Connected		
Number of FCs: δ_1	1-3	2
Number of neurons/FC: δ_2	1-50	10
Output	256 neurons Activation function: Softmax	

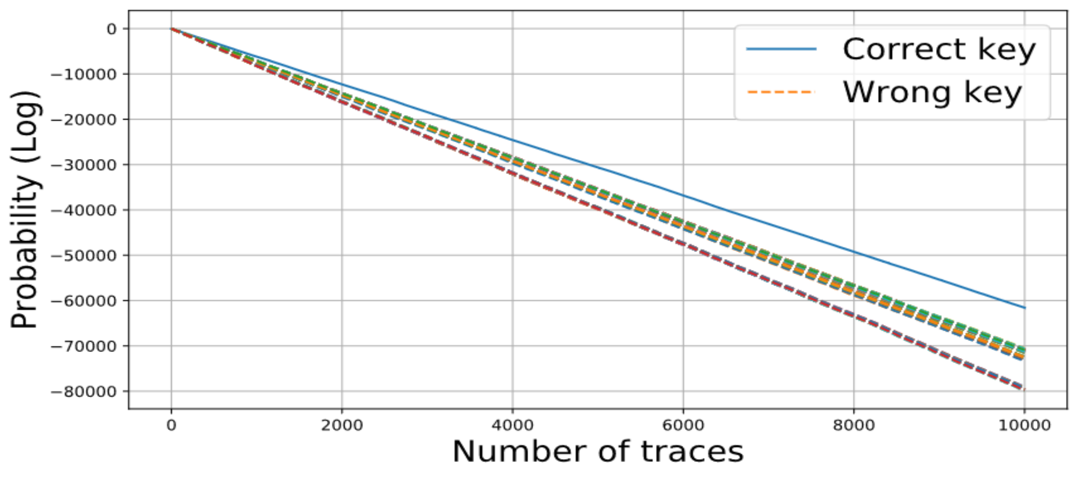

Figure 15. Estimation probability of all hypothetical keys against number of traces with Dataset3

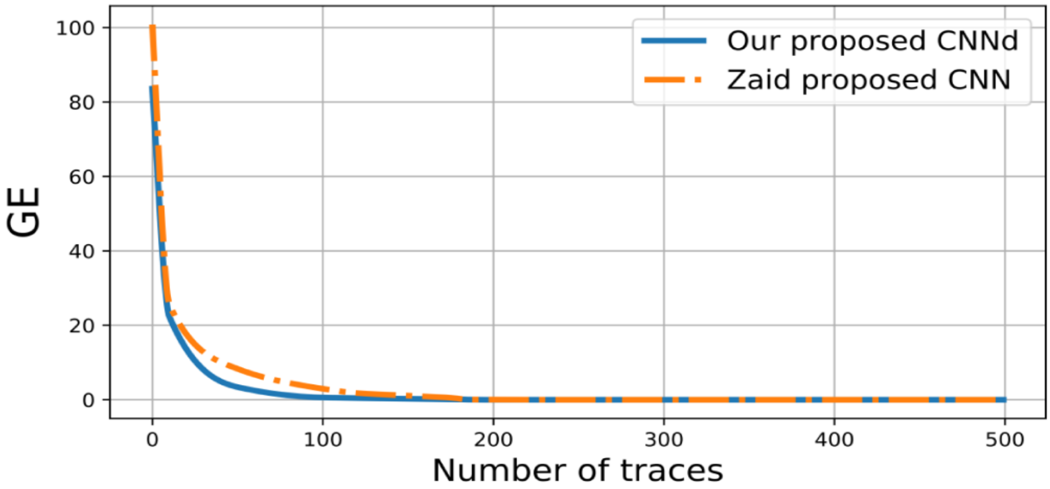

Figure 16. Guessing entropy results for Dataset3

Table 7. Comparison of performance on DataSet3

	Template Attack [12]	**CNN Profiled attack [12]**	**CNN proposed by Zaid [13]**	**CNN proposed by us**
Trainable parameters	\	66.652.444	16.960	26.334
Number of traces for GE=0	450	1.146	195	183
Training time (s)	\	5417	253	790

5. CONCLUSION

In this paper, we have demonstrated that deep learning can be successfully applied to profiled attacks on cryptographic devices. By analyzing the POIs characteristics of power traces and convolution operations, we have proposed two basic CNN architectures, CNNn and CNNd, used for unprotected and masking-protected devices, respectively. The parameters of the proposed basic CNN architecture are optimized by the GWO algorithm. Our CNNn architecture has minimal complexity, and requires only 2 to 4 traces, to reveal the correct key of unprotected devices. After experimenting successfully on both trace datasets, we claim that CNNn should be the first choice when conducting profiled attacks on unprotected devices. Regarding attacking masking-protected devices, although the architecture of CNNd has one more convolution layer than the CNN architecture of Zaid, it gives better results, specifically a 5% decrease in the number of traces required for GE to equal 0. Therefore, both CNN architectures should be used with the protected device. As a final note, CNN can be used to conduct profiled attacks efficiently assuming its architecture and parameters have been carefully selected.

REFERENCES

[1] P. Kocher, J. Jaffe, B. Jun, "Differential Power Analysis," *Advances in Cryptology — CRYPTO' 99. CRYPTO 1999. Lecture Notes in Computer Science*, vol 1666. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-48405-1_25

[2] P .Kocher, "Timing attacks on implementations of DiffieHellman, RSA, DSS, and other systems," in *Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology*, Santa Barbara (USA), 1996.

[3] K. Gandolfi, C. Mourtel, G. Oliver, "Electromagnetic analysis: Concrete results," in *Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems*, Paris, 2001.

[4] F. Standaert, C. Archambeau, "Using subspace-based template attacks to compare and combine power and electromagnetic information leakages," in *Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems - CHES 2008*, Washington, D.C (USA), 2008.

[5] S. Chari, JR. Rao, P. Rohatgi, "Template Attacks," *Cryptographic Hardware and Embedded Systems - CHES 2002. CHES 2002. Lecture Notes in Computer Science*, vol 2523. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-36400-5_3

[6] E. Brier, C. Clavier,F. Olivier, "Correlation power analysis with a leakage model," in *International Workshop on Cryptographic Hardware and Embedded Systems,* Springer, 2004, pp. 16–29. https://doi.org/10.1007/978-3-540-28632-5_2

[7] B. Gierlichs, et al, "Mutual information analysis," *Cryptographic Hardware and Embedded Systems – CHES 2008. CHES 2008. Lecture Notes in Computer Science,* vol 5154. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-85053-3_27

[8] W. Schindler, K. Lemke, C. Paar, "A stochastic model for differential side channel cryptanalysis," *Cryptographic Hardware and Embedded Systems – CHES 2005. CHES 2005. Lecture Notes in Computer Science,* vol 3659. Springer, Berlin, Heidelberg. https://doi.org/10.1007/11545262_3

[9] B. Hettwer, S. Gehrer, T. Güneysu, "Applications of machine learning techniques in side-channel attacks: a survey," *J Cryptogr Eng,* vol. 10, p. 135–162 , 2020.

[10] H. Maghrebi, T. Portigliatti, E. Prouff, "Breaking cryptographic implementations using deep learning techniques," *Security, Privacy, and Applied Cryptography Engineering. SPACE 2016. Lecture Notes in Computer Science,* vol 10076. Springer, Cham. https://doi.org/10.1007/978-3-319-49445-6_1

[11] E. Cagli, C. Dumas, E. Prouff, "Convolutional neural networks with data augmentation against jitter-based countermeasures, *In Wieland Fischer and Naofumi Homma, editors,Cryptographic Hardware and Embedded Systems – CHES 2017*, Cham, Springer International Publishing, 2017, pp. 45–68.

[12] E. Prouff, R. Strullu, R. Benadjila, E. Cagli, C. Dumas, "Study of deep learning techniques for side-channel analysis and introduction to ascad database," *Cryptology ePrint Archive, Report 2018/053, 2018*, https://eprint.iacr.org/2018/053, 2018

[13] G. Zaid, L. Bossuet, A. Habrard, A. Venelli, "Methodology for efficient cnn architectures in profiling attacks," *IACR Transactions on Cryptographic Hardware and Embedded Systems, IACR, 2020,* 2020 (1), pp.1-36. Doi: https://doi.org/10.13154/tches.v2020.i1.1-36

[14] J. Heaton, I. Goodfellow, Y. Bengio, A. Courville, *Deep Learning,* MIT Press, 2016.

[15] S. Ioffe, C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning,* 2015.

[16] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Advances in Engineering Software,* vol. 69, pp. 46–61, 2014.

[17] F. Chollet et al, "Keras," https://keras.io, 2015.

[18] L.N. Smith, N. Topin, "Super-convergence: Very fast training of residual networks using large learning rates," in *ICLR 2018 Conference, CoRR,* 2018.

[19] G. Klambauer, T. Unterthiner, A. Mayr, S. Hochreiter, "Self-normalizing neural networks," *arXiv:1706.02515.*

[20] K. He, X. Zhang, S. Ren, J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), ICCV '15*, Washington, DC, USA, 2015, pages 1026–1034.

[21] D.H. Wolpert, W.G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evolut. Comput,* vol. 1, no. 1, pp. 67–82, 1997.

[22] J. Kim, S. Picek, A. Heuser, S. Bhasin, A. Hanjalic, "Make some noise. Unleashing the power of convolutional neural networks for profiled side-channel analysis", *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2019, no. 3, pp. 148–179. https://doi.org/10.13154/tches.v2019.i3.148-179

[23] S. Mangard, E. Oswald, T. Popp, *Power Analysis Attacks Revealing the Secrets of Smart Cards*, New York: USA: Springer, 2010.

[24] A. Heuser and M. Zohner, "Intelligent machine homicide breaking cryptographic devices using support vector," in *COSADE 2012,* Heidelberg, 2012.

[25] G. Hospodar, B. Gierlichs, E. De Mulder, I. Verbauwhede, J. Vandewalle, "Machine learning in side-channel analysis: A first study," *J Cryptogr Eng,* vol. 1, article number 293, 2011. https://doi.org/10.1007/s13389-011-0023-x

[26] G. Hospodar, E. De Mulder, B. Gierlichs, J. Vandewalle, and I. Verbauwhede, "Least squares support vector machines for side-channel analysis," in *COSADE 2011*, Darmstadt, 2011.

[27] L. Lerman, S. F. Medeiros, G. Bontempi, and O. Markowitch, "A machine learning approach against a masked AES," *J Cryptogr Eng,* vol. 5, pp. 123–139, 2015. https://doi.org/10.1007/s13389-014-0089-3

[28] S. Picek, A. Heuser, A. Jovic, S.A. Ludwig, S.Guilley, D. Jakobovic, N. Mentens, "Side-channel analysis and machine learning: A practical perspective," *2017 International Joint Conference on Neural Networks (IJCNN)*, Anchorage, AK, 2017, pp. 4095-4102. Doi: 10.1109/IJCNN.2017.7966373.

[29] S. Picek, A. Heuser, A. Jovic, L. Batina, and A. Legay, "The secrets of profiling for side-channel analysis: feature selection matters," *IACR Cryptology ePrint Archive*, 2017.

[30] Y. Zheng, Y. Zhou, Z. Yu, C. Hu, H. Zhang, “How to compare selections of points of interest for side-channel distinguishers in practice?,” in *Information and Communications Security. ICICS 2014. Lecture Notes in Computer Science,* vol 8958. Springer, Cham, 2014. https://doi.org/10.1007/978-3-319-21966-0_15

[31] Y. Kong, E. Saeedi, “The investigation of neural networks performance in side channel attacks,” *Artif Intell Rev,* vol. 52, pp. 607–623, 2019. https://doi.org/10.1007/s10462-018-9640-4

[32] F. Standaert, T. Malkin, M. Yung, “A unified framework for the analysis of side-channel key recovery attacks,” *Advances in Cryptology - EUROCRYPT 2009. EUROCRYPT 2009. Lecture Notes in Computer Science,* vol 5479. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-01001-9_26

Received 25 August 2020
Accepted 14 January 2021