Comparison of Profiling Power Analysis Attacks Using Templates and Multi-Layer Perceptron Network

Zdenek Martinasek and Lukas Malina

Abstract—In recent years, the cryptographic community has explored new approaches of power analysis based on machine learning models such as Support Vector Machine (SVM), Multi-Layer Perceptron (MLP) or Random Forest (RF). Realized experiments proved that the method based on MLP can provide almost 100% success rate after optimization. Nevertheless, this description of results is based on the first order success rate that is not enough satisfactory because this value can be deceiving. Moreover, the power analysis method based on MLP has not been compared with other well-known approaches such as template attacks or stochastic attacks yet. In this paper, we introduce the first fair comparison of power analysis attacks based on MLP and templates. The comparison is accomplished by using the identical data set and number of interesting points in power traces. We follow the unified framework for implemented side-channel attacks therefore we use guessing entropy as a metric of comparison.

Keywords—Power Analysis, Neural Network, Template Attack, Comparison.

I. INTRODUCTION

Power analysis (PA) measures and analyzes the power consumption of cryptographic devices depending on their activity. It was introduced by Kocher in [1]. The goal of PA is to determine the sensitive information of cryptographic devices from the measured power consumption and to apply the obtained information in order to abuse the cryptographic device. A detailed description of power analysis including side-channel sources, testbeds, statistical tests and countermeasures is summarized in the book [2].

A. Related Work

Application of neural networks in the field of power analysis was first published in [3]. Naturally, this work was followed by other authors, e.g. [4], [5], who dealt with the classification of individual power prints. These works are mostly oriented towards reverse engineering. Yang et al. [6] proposed MLP in order to create a power consumption model of a cryptographic device in DPA based on correlation coefficient. In recent years, the cryptographic community has explored new approaches based on machine learning models. Lerman et al. [7], [8] compared a template attack (TA) with a binary machine learning approach based on non-parametric methods. Hospodar et al. [9], [10] analysed the SVM on a software implementation of a block cipher. Heuser et al. [11] created the general description of the SVM attack and compared this approach with the template attack. In 2013, Bartkewitz [12] applied a multi-class machine learning model that improves the attack success rate with respect to the binary approach. Recently, Lerman et al. [13] proposed a machine learning approach that takes into account the temporal dependencies between power values. This method improves the success rate of an attack in a low signal-to-noise ratio with respect to classification methods. Lerman et al. [14] presented a machine learning attack against a masking countermeasure, using the dataset of the DPA Contest v4. Interesting method of power analysis based on a multi-layer perceptron was first presented in [15]. In this work, the authors used a neural network directly for the classification of the AES secret key. In [16], this MLP approach was optimized by using the preprocessing of the power traces measured.

B. Contribution

In [15], [16], the authors used the first order success rate for efficiency description of the proposed MLP power analysis method. This is not sufficiently reliable because this value can be deceiving [17]. According the framework, the guessing entropy represents an appropriate metric of two side analysis attack implementation [17]. The metric measures the average number of key candidates to test after the side-channel attack.

Other important fact is that both methods based on MLP (original implementation and optimized one) have not been yet compared with other well-known approaches such as the template attack or the stochastic attack. In this paper, we introduce the first fair comparison of power analysis attacks based on the MLP and templates. The comparison is accomplished by using the identical data set including a number of interesting points. In previous researches described in [15], [16], the adversary uses 1200 interesting points to realize the attack. This large number of interesting points is not practically applicable to TA because of possible numerical problems connected with a covariance matrix. Moreover, we create a general description of the MLP aimed for byte classification including the structure, setting and training algorithm, because this information was also missing in previous research.
II. GENERAL DESCRIPTION OF THE MLP

This section provides only a basic information about the neural networks that we used during the attack (the basic structure and the training algorithm of the MLP). We refer the work [18], [19] for more specific information. The main goal of this section is to show how to use MLP to realize the side channel attack.

The basic element of an artificial neural network is a formal neuron, often called as a perceptron in the literature. The basic model of the neuron is shown on the left side in Fig. 1. The neuron contains \( x_i \) inputs that are multiplied by the weights \( w_i \), where \( i = 1 \) to \( n \). Input \( x_0 \) multiplied by the weight \( w_0 = -\theta \) determines the threshold of the neuron (bias). During the training of the neuron, weights are updated to achieve a desired output value. Firstly, a post-synaptic potential is calculated. It is defined as the internal function of the neuron:

\[
\xi = \sum_{i=1}^{n} x_i w_i - \theta. \tag{1}
\]

Subsequently, the output value of the neuron is calculated as \( y = f(\xi) \) where \( f \) represents a non-linear function, mostly a sigmoid. Naturally, one formal neuron is not able to solve complex problems, therefore we use neurons (perceptrons) connected into a network. The multilayer perceptron consists of two or more layers of neurons that are denoted as an output layer and a hidden layer. Each neuron in one layer is connected with a certain weight \( w_{ij} \) to every neuron in the following layer. Frequently, the input layer is not included when one is counting the number of layers because the input layer is not composed of neurons. We follow this notation in this article. An example of the two-layer neural network is shown in Fig. 1 (on the right side).

These networks are modifications of the standard linear perceptron and can distinguish data that are not linearly separable [19]. These networks are widely used for a pattern classification, recognition, prediction and approximation and utilize mostly a supervised learning method called backpropagation [20]. The backpropagation (BPG) algorithm is an iterative gradient learning algorithm which minimizes squares of a cost function using the adaptation of the synaptic weights. This method is described with the following steps (the following equations are valid for the two-layer neural network which is shown in Fig. 1):

- **Step 1:** Weights \( w_{ij} \) and thresholds \( \theta \) of each neuron are initialized with random values.
- **Step 2:** An input vector \( \mathbf{X} = [x_1, \ldots, x_N]^T \) and a desired output vector \( \mathbf{D} = [d_1, \ldots, d_M]^T \) are applied to the neural network. In other words, one creates a training set containing pairs of \( \mathbf{T} = \{[\mathbf{X}_1, \mathbf{D}_1], [\mathbf{X}_2, \mathbf{D}_2], \ldots, [\mathbf{X}_n, \mathbf{D}_n]\} \), where \( n \) denotes the number of training set patterns and the training set prepared is applied to the neural network. Provided, that NN represents an ordinary classifier which classifies input data to the desired output groups, the \( \mathbf{D} \) represents mostly a classification matrix where the desired outputs are labeled by value 1 and other outputs 0.
- **Step 3:** The current output of each neuron is calculated by the following equations:

\[
y_k(t) = f_s(\sum_{k=1}^{N_1} w_{jk}(t)x_j(t) - \theta_k), \tag{2}
\]

\[
x_j(t) = f_s(\sum_{i=1}^{N} w_{ij}(t)x_i(t) - \theta_j), \tag{3}
\]

where \( 1 \leq k \leq M \) denoted output layer and \( 1 \leq j \leq N_1 \) hidden layer.
- **Step 4:** Weights and thresholds are applied according to the following equation:

\[
w_{ij}(t + 1) = w_{ij}(t) + \eta \delta_j x_i. \tag{4}
\]

Adaptation of weight values starts at the output neurons and proceeds recursively back to the input neurons. In this equation, \( w_{ij} \) denotes weights between the \( i \)-th hidden or input neuron and the neuron \( j \)-th at time \( t \). Output of the \( i \)-th neuron is denoted as \( x_i \), \( \eta \) represents the learning coefficient and \( \delta_j \) is an error of neuron which is calculated as follows:

\[
\delta_j = y_j(1 - y_j)(d_j - y_j), \quad \text{(output layer)}, \tag{5}
\]

\[
\delta_j = x_j'(1 - x_j')(\sum_{k=1}^{M} \delta_k w_{jk}), \quad \text{(hidden layer)}, \tag{6}
\]

where \( \delta_k \) represents all neurons in the output layer.
- **Step 5:** Steps from 3 to 5 are repeated until the error value is less than the predetermined value.

During the training of NN which is based on the BPG algorithm, some problems may occur. These problems are caused by inappropriate setting of training parameters or the improper initialization of weights and thresholds. These difficulties can be reduced by using a modification of the basic algorithm such as Back-Propagation with Momentum or Conjugate Gradient Backpropagation.

III. GENERAL DESCRIPTION OF MLP ATTACK

In this section, we describe the general usage of the MLP in power analysis attack. Machine learning algorithms are mostly used in profiled attacks where an adversary needs a physical access to a pair of identical devices, which we call a profiling device and a target device. Basically, these attacks consist of two phases. In the first phase, the adversary analyzes the
profiling device and then, in the second phase, the adversary attacks the target device. Typical examples are template-based attacks [21], [2], [22]. By contrast, non-profiled attacks are one-phase attacks that perform the attack directly on the target device such as DPA based on the correlation coefficient [23].

A. Profiling Phase of MLP Attack

In the attack based on the MLP, we assume that we can characterize the profiling device using a well trained neural network. We assume that desired value by adversary is the secret key stored in cryptographic device. This means that one can create and train a NN for a certain part of a cryptographic algorithm. We execute this sequence of instructions on the profiling device with the same data \( d \) and different key values \( k_j \) to record the power consumption. After measuring \( n \) power traces, it is possible to create the matrix \( X_n \) that contains power traces corresponding to a pair of \( (d, k_i) \). These pairs represent a training set \( T \) of the neural network. Input values are power traces measured and values of secret key \( k_i \) represent the desired output of the neural network. In this case, secret key values \( k_i \) can be easily represented using the \( n \times 256 \) classification matrix \( D \).

After the measurement phase, an adversary creates a neural network. The number of input neurons has to be equal to the numbers of chosen interesting points. We use only interesting points because memory limitation and time-consuming training process (similar situation like in classical Template attack). Generally, the setting of the hidden layer depends on problem solving and the training set, therefore the adversary has to set the number of hidden layers and neurons experimentally. The output layer should contain the desired number of neurons corresponding to the aim of the attack (output byte of S-Box, byte of the secret key, Hamming weight etc.). In our example, the NN is aimed on byte classification, therefore the output layer contains 256 neurons. In the last step of the profiling phase, the adversary trains the neural network created by the prepared training set and the chosen training algorithm.

B. Attack Phase of MLP Attack

During the attack phase, the adversary uses a well-trained NN together with a measured power trace from the target device (denoted as \( t \)) to determine the secret key value. The adversary puts the \( t = [x_1, \ldots, x_N]^T \) as an input to NN and it classifies the output values using the calculation:

\[
y_k = f_s(\sum_{k=1}^{N_1} w_{ik}'x_j - \theta_k'), 1 \leq k \leq M,
\]

where \( w_{ik}' \) denotes weights between \( i \)-th hidden neuron (or the input neuron) and the neuron \( j \)-th and \( x_i \) denotes the output of hidden neurons:

\[
x_j' = f_s\left(\sum_{i=1}^{N} w_{ij}x_i - \theta_j\right), 1 \leq j \leq N_1.
\]

The result of this classification is a vector \( g = [g_1, g_2, \ldots, g_M] \) which contains the probability value 0 to 1 for every output value. The probabilities show how well the measured trace \( t \) corresponds to the training patterns. Intuitively, the highest probability should indicate the correct training pattern in the training set \( T \) and because each training pattern \( X_n \) is associated with a desire value (in our case secret key), the adversary obtains the information about secret key stored in the target device.

IV. TESTBED AND IMPLEMENTATION DESCRIPTION

This section summarizes the most important facts about the experimental setup and the implementation of the attacks. A complete AES algorithm with a key length of 128 bits was implemented into the cryptographic module and the synchronization was performed only for the AddRoundKey and SubBytes operations in the initialization phase of the algorithm. The stored secret key can be expressed in bytes as \( K_{sec} = \{k_1, k_2, \ldots, k_{16}\} \) where \( k_i \) represents individual bytes of the key. The program allowed setting of the secret key and plain text value indicated this operation by sending the respective value via a serial port to a computer. The synchronization signal and the communication with the computer did not affect the power consumption of the cryptographic module. The cryptographic module was represented by PIC 8-bit microcontroller, and for the power consumption measurement we used CT-6 current probe and Tektronix DPO-4032 digital oscilloscope. We used standard operating conditions with 5 V power supply.

Because our implementation was realized in the assembly language and the executed instructions of examined operations (AddRoundKey and SubBytes) were exactly the same for every key byte \( k_i \), we assume that it is possible to use parts of power traces where first byte is processed (see Fig. 2) to build template and train the neural network to determine the whole secret key byte by byte. In the first step, we determine the value of \( k_1 \), and in the second step, byte \( k_2 \) and so on. The difference between these steps is in the division of the power traces into parts corresponding to the time intervals in which the cryptographic device works with the respective bytes of the secret key. The division of power traces is indicated in Fig. 2 by numbers and every part of a power trace contained 1,200 samples. We verify this assumption experimentally and
it is naturally conditioned by the excellent synchronization of measured power traces.

We measured a set of 2,560 power traces where ten power traces were independently stored for each value of the first secret key byte. This number of power traces was chosen because we wanted to compare both implementation of the attack (MLP approach and template) using the typical 10-fold cross-validation. In data mining and machine learning, the 10-fold cross-validation is the most common method of model verification. Cross-validation (CV) is a statistical method of evaluating and comparing learning algorithms by dividing data into two segments: one is used for learning a model and the other one is used for the model validation. In typical cross-validation, the training and validation sets must cross-over in successive rounds that each data point has a chance of being validated against. Therefore, we used 9 power traces in profiling phase of the attack and one power trace in attack phase in every step of validation.

We chose five interesting points according to the information provided in [24]. Our algorithm searched for the maximum differences of an average power consummation and power consumption corresponding to key value 1. The algorithm accepted only the maximums that had a distance of at least one clock cycle from each other. This restriction for having interesting points not too close from each other avoids numerical problems during the covariance matrix inverting. Measured power traces were properly synchronized and our device leaks Hamming weight (HW) of processed data. These facts confirm the plots shown in Fig. 3 and Fig. 4. Figure 3 shows the detail of power traces that correspond to MOV instruction where data values 0 to 255 were processed. Figure 4 shows plot of these measured power traces for one point \( t = 4, 086 \). Each of our chosen interesting points leaked HW of processed data. Same chosen points were used for the template creation and the neural network model.

A well-known fact is that noise always poses the problem during the power consumption measurement. We performed the experimental measurements of a test bed that were made according to the information provided in [2] and we established that the noise level was distributed according to the normal distribution with the parameters \( \mu = 0 \text{mA} \) and \( \sigma = 5 \text{mA} \). Every stored power trace was calculated as an average power trace from ten power traces measured using the digital oscilloscope to reduce the electronic noise.

### A. Template Attack Implementation

We implemented the classical template attack and reduced template attack to compare the classification results with MLP attack. We were interested in effective template attack based on pooled covariance matrix [22], therefore we calculated the pool covariance matrix as an average value of all covariance matrices and we calculated the probability density function (Eq. 9) with this matrix. Implementations of template attacks were done according to the Eq. 9:

\[
p(t; (m, C)_{d, k}) = \frac{\exp\left(-\frac{1}{2} \cdot (t - m)\cdot C^{-1}\cdot (t - m)\right)}{\sqrt{(2 \cdot \pi)^{NP} \cdot \det(C)}}
\]

where \((m, C)\) represents templates prepared in profiling phase based on multivariate normal distribution that is fully defined by a mean vector and a covariance matrix. Measured power trace from the target device is denoted as \( t \) and \( NI \) is the number of interesting points. In following text, classical template, reduced template and template attack based on the pooled covariance matrix are denoted as \( T_{cls}, T_{red} \) and \( T_{pool} \) sequentially. All template attack implementations were made in the Matlab environment.

### B. MLP Attack Implementation

We created and trained the neural network in Matlab using the Netlab neural network toolbox [18]. Ian Nabney and Christopher Bishop from Aston University in Birmingham are the authors of this toolbox and it is available for downloading. We created a typical two layer perception network and we used optimized learning based on the scaled conjugate gradient algorithm (see Sec. II). A standard sigmoid was chosen as an activation function. The created NN is shown in Fig. 5. The input layer contained 5 inputs corresponding with interesting
V. Obtained Results

The measured set of 2,560 power traces was used for the comparison of implemented methods. We realized a typical 10-fold cross-validation, where nine power traces were used for the template preparation and neural network training in the profiling phase and one power trace was used in the attack phase in every step of the cross-validation. We used the guessing entropy to compare our implemented attacks.

The guessing entropy is defined as follows: let \( g = [p_1,p_2,\ldots,p_N] \) contains the probability such as \( p_1 \geq p_2 \geq \ldots \geq p_N \) of all possible key candidates after \( N \) iterations of Eq. 9 or Eq. 7. Indices \( i \) correspond with the correct key in \( g \). After the realization of \( S \) experiments, one obtains a matrix \( G = [g_1,\ldots,g_S] \) and a corresponding vector \( i = [i_1,\ldots,i_S] \). Then the guessing entropy determines the average position of the correct key:

\[
GE = \frac{1}{S} \sum_{x=1}^{S} i_x. \tag{10}
\]

In other words, the guessing entropy describes the average number of guesses, required for recovering the secret key [17], [11].

In the first experiment, we determined the value of one byte of the secret key from one measured power trace. We tried this for all 256 power traces measured corresponding to every key values from 0 to 255. In other words, we determined the value of 256 individual bytes in every step of the cross-validation. After the realization, we calculated the \( GE \) according to the Eq. 10. Obtained results are summarized in Tab. I, where \( \phi \) denotes an average value calculated from every cross-validations realized. The template attack based on the pooled covariance matrix \( T_{pol} \) achieved the best result in one byte guessing but it is important that the classification based on NN was not much worse. The original implementation of the neural network \( NN_{org} \) was the worst of all implemented attacks and achieved \( GE = 1.18 \) in average. The optimized method achieved \( GE = 1.04 \) that was almost identical with template attacks.

In the second experiment, we determined the whole 128 bit secret key by using the 16 power traces measured. The secret key stored had value \( K = [29, 245, 48, 93, 215, 65, 139, 198, 5, 232, 81, 107, 173, 243, 24, 151] \). Obtained results are written in Tab. II. The second experiment confirmed the previous results. The adversary needs about 18 guesses to determine the correct secret key after the side-channel attack based on the original implementation of neural network \( NN_{org} \). The results of the optimized method were almost identical with template attacks. Potential adversary would need in average about 4 guesses to determine the secret key value after the side-channel attack. Our experiments confirm that success revelation of secret key is comparable for MLP and template based attacks (identical number of interesting points, number of power traces and so on). MLP is able to be trained only for a few interesting points of power traces. In order to complete the comparison of implemented attacks, Tab. III provides the information about the time complexity of attack phase \( \tau \) and memory complexity \( m \).

<table>
<thead>
<tr>
<th>Step of CV</th>
<th>( NN_{org} )</th>
<th>( NN_{opt} )</th>
<th>( T_{cls} )</th>
<th>( T_{red} )</th>
<th>( T_{pol} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.16</td>
<td>1.02</td>
<td>1.07</td>
<td>1.04</td>
<td>1.02</td>
</tr>
<tr>
<td>2</td>
<td>1.18</td>
<td>1.04</td>
<td>1.07</td>
<td>1.06</td>
<td>1.02</td>
</tr>
<tr>
<td>3</td>
<td>1.32</td>
<td>1.03</td>
<td>1.04</td>
<td>1.04</td>
<td>1.03</td>
</tr>
<tr>
<td>4</td>
<td>1.16</td>
<td>1.05</td>
<td>1.04</td>
<td>1.04</td>
<td>1.02</td>
</tr>
<tr>
<td>5</td>
<td>1.16</td>
<td>1.05</td>
<td>1.07</td>
<td>1.05</td>
<td>1.02</td>
</tr>
<tr>
<td>6</td>
<td>1.23</td>
<td>1.04</td>
<td>1.04</td>
<td>1.04</td>
<td>1.02</td>
</tr>
<tr>
<td>7</td>
<td>1.15</td>
<td>1.03</td>
<td>1.08</td>
<td>1.03</td>
<td>1.02</td>
</tr>
<tr>
<td>8</td>
<td>1.11</td>
<td>1.05</td>
<td>1.07</td>
<td>1.02</td>
<td>1.02</td>
</tr>
<tr>
<td>9</td>
<td>1.18</td>
<td>1.06</td>
<td>1.08</td>
<td>1.02</td>
<td>1.00</td>
</tr>
<tr>
<td>10</td>
<td>1.17</td>
<td>1.03</td>
<td>1.03</td>
<td>1.04</td>
<td>1.01</td>
</tr>
</tbody>
</table>

| \( \phi \) | 1.18           | 1.04           | 1.06        | 1.04        | 1.02        |

<table>
<thead>
<tr>
<th>Step of CV</th>
<th>( NN_{org} )</th>
<th>( NN_{opt} )</th>
<th>( T_{cls} )</th>
<th>( T_{red} )</th>
<th>( T_{pol} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4.00</td>
<td>2.00</td>
<td>4.00</td>
<td>4.00</td>
<td>2.00</td>
</tr>
<tr>
<td>2</td>
<td>24.00</td>
<td>4.00</td>
<td>4.00</td>
<td>1.00</td>
<td>2.00</td>
</tr>
<tr>
<td>3</td>
<td>32.00</td>
<td>2.00</td>
<td>4.00</td>
<td>4.00</td>
<td>8.00</td>
</tr>
<tr>
<td>4</td>
<td>24.00</td>
<td>8.00</td>
<td>2.00</td>
<td>4.00</td>
<td>4.00</td>
</tr>
<tr>
<td>5</td>
<td>4.00</td>
<td>2.00</td>
<td>4.00</td>
<td>4.00</td>
<td>4.00</td>
</tr>
<tr>
<td>6</td>
<td>30.00</td>
<td>4.00</td>
<td>16.00</td>
<td>4.00</td>
<td>4.00</td>
</tr>
<tr>
<td>7</td>
<td>8.00</td>
<td>2.00</td>
<td>4.00</td>
<td>2.00</td>
<td>2.00</td>
</tr>
<tr>
<td>8</td>
<td>16.00</td>
<td>6.00</td>
<td>8.00</td>
<td>2.00</td>
<td>2.00</td>
</tr>
<tr>
<td>9</td>
<td>32.00</td>
<td>2.00</td>
<td>4.00</td>
<td>4.00</td>
<td>2.00</td>
</tr>
<tr>
<td>10</td>
<td>2.00</td>
<td>1.00</td>
<td>2.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
</tbody>
</table>

| \( \phi \) | 17.60| 3.30 | 5.20 | 3.00 | 3.10 |

| \( \tau \) [\( ms \)] | 1.59 | 1.11 | 174.89 | 149.85 | 221.66 |
| \( m \) [\( kB \)] | 1,920.00 | 1,920.00 | 94.20 | 22.30 | 22.60 |
VI. Conclusion

In this paper, we made the first fair comparison of power analysis using the MLP with well-known template attacks. We followed the unified framework for two implementations of the side-channel attack, therefore we used a guessing entropy as a metric of comparison. The comparison was made by using the same data set and same number of interesting points for all implementation. We described the usage of MLP in power analysis attack including the structure, setting and training algorithm because these information were missing in the previous research.

The experiment realized, that determined the whole secret key of the AES algorithm, confirmed that the efficiency of the power analysis attack based on MLP and the template attack is comparable. By contrast, the adversary needs about 18 guesses to determine the correct secret key using the original implementation of the MLP attack. This result is three times worse in comparison with the classical template attack that needs about 5.2 guesses to reveal the whole secret key. For these reasons, we do not recommend a usage of original implementation of MLP attack. The results of optimized method were almost identical with template attacks. Potential adversary would need in average about 4 guesses after the side-channel attack to determine the secret key value of AES algorithm.

REFERENCES


