Optimization of Power Analysis Using Neural Network

Zdenek Martinasek, Jan Hajny, and Lukas Malina

Brno University of Technology, Department of Telecommunications
Technicka 12, 612 00 Brno, Czech Republic

Abstract. In power analysis, many different statistical methods and power consumption models are used to obtain the value of a secret key from the power traces measured. An interesting method of power analysis based on multi-layer perceptron was presented in [1] claiming a 90% success rate. The theoretical and empirical success rates were determined to be 80% and 85%, respectively, which is not sufficient enough. In the paper, we propose and realize an optimization of this power analysis method which improves the success rate to almost 100%. The optimization is based on preprocessing the measured power traces using the calculation of the average trace and the subsequent calculation of the difference power traces. In this way, the prepared power patterns were used for neural network training and of course during the attack. This optimization is computationally undemanding compared to other methods of preprocessing usually applied in power analysis, and has a great impact on classification results. In the paper, we compare the results of the optimized method with the original implementation. We highlight positive and also some negative impacts of the optimization on classification results.

Keywords: Power analysis, neural network, optimization, preprocessing

1 Introduction

Power analysis (PA) measures and analyzes the power consumption of cryptographic devices depending on their activity. It was introduced by Kocher in [2]. 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. There are two basic methods of power analysis: simple PA and differential PA. The attacker tries to determine the secret key directly from the traces measured in the simple power analysis (SPA). In the most extreme case, this means that the attacker attempts to reveal the key based on one single power trace. The goal of the differential power analysis (DPA) attacks is to reveal the secret key of the cryptographic module by using a large number of power traces that have been recorded while the device was encrypting or decrypting various input data. Power analysis is widely discussed

and belongs to the most popular types of side channel analysis methods because the attacker does not need any expensive special equipment. A detailed description of power analysis including side-channel sources, testbed, statistical tests and countermeasures is summarized in the book [3].

1.1 Related Work

Simple power analysis attacks were described by Kocher in [2]. A typical example of SPA is the attack on the implementation of the RSA (Rivest Shamir Adleman) asymmetric cryptographic algorithm, where the difference in power consumption between the operations of multiplication and squaring can be observed [4]. Template based attacks are another type of SPA attack, which were introduced in [5]. Practical aspects of template attacks have been discussed in [6, 7].

The concept of the DPA attack was first described also in [2] and the basic principle was introduced on a DES algorithm using the statistical method based on the Difference of Means. Subsequently, applicable statistical tests were discussed in [8]. An important question of the impact of preprocessing the measured data on the effectiveness of DPA was presented in [9, 10]. The application of the correlation coefficient as a statistical method in DPA was described in [11] and nowadays, this method is one of the most widely used. A detailed description of the general schema on which all power analyses are based and the best known statistical tests including the basic power simulation models are given in [3, 12].

One of the first examples of digital signal processing applied to side channel analysis can be found in [13]. Digital filtering is used to facilitate attacks based on side channel analysis for devices such as Xilinx Field Programmable Gate Arrays (FPGAs) [14], Radio Frequency Identification (RFID) devices [15–17] and Cortex-M3 SoC [18].

Neural networks (NN) are used mostly in the cryptography branch to realize key distribution [19], hash functions [20], random number generators [21], in public-key cryptography [22], and in the exchange protocols [23] (similar to the Diffie-Hellman protocol). The publications [24, 25] dealing with the use of NN in the side channel cryptanalysis are mostly focused on acoustic side channels, where NN are used for the classification of captured records of buttons pressed on a keyboard.

In the field of power analysis, the possibility of using neural networks was first published in [26]. Naturally, this work was followed by other authors, e.g. [27–29], who dealt with the classification of individual power prints. These works are mostly oriented towards reverse engineering based on power print classification. The usage of neural networks for the classification of a secret key value has been sparsely published and tested yet. Works [30–33] dealing with this issue are based on machine learning algorithm such as support vector machines (SVM).

An interesting method based on typical multi-layer perceptron (MLP) was demonstrated in [1]. In this work, a neural network was used for the classification of the AES secret key. This power analysis method uses a typical two-layer perceptron (three-layer neural network if we take into consideration the input

layer) to determine the secret key value only from one power trace measured. First classification results were really promising and this method achieved a successful classification of 90% for the first byte of the secret key. The method was thoroughly tested using 2560 power traces and an empirical success rate of around 85% was determined. The theoretical success rate determined from the results was only about 80%. Other negative characteristics were revealed during the subsequent testing, e.g. the distribution of the maximum probability values or the low probability value of selected key estimation.

1.2 Our Contribution

Our contribution lies in the optimization of the power analysis method described in [1]. We minimize the above-mentioned negative characteristics of the method implementation to increase the success rate of classification. The optimization is based on preprocessing the power traces measured, using the calculation of the average trace and the subsequent calculation of the difference power traces. Preprocessed power patterns are used for the neural network training and, naturally, during the attack phase, in the same way as described in [1]. This optimization is computationally undemanding compared to other methods of preprocessing usually applied in power analysis (e.g. filtering [34]) and has a great impact on the classification results. In the paper, we compare the results of the optimized method with the original implementation. We highlight the positive and also some small negative impacts of the optimization on the classification results. Both methods were verified using $2,560$ power traces corresponding to all the values of the secret key to analyze the repeatability and feasibility of the method. In the original paper, the cross-validation was not used to verify the neural network, we decided to compare the original method implementation with the optimized method, using the typical 10-fold cross-validation. In data mining and machine learning, the 10-fold cross-validation is the most common way to verify the model. Our contribution can be summarized in the following main points:

- optimization proposal,
- implementation of optimization,
- comparison of results,
- cross-validation of both implementations.

2 Method and Testbed Description

The following text summarizes the most important facts about the original implementation of the power analysis method and the experimental setup. The fundamental goal of the method is to obtain from the power trace measured the secret key value, which is stored in the cryptographic module. In the following text, we denote the secret key stored in the attacked cryptographic module as K_{sec}, and the estimated value of secret key, which was determined using a neural

network, as K_{est}. Naturally, if the method works correctly, the values K_{est} and K_{sec} are equal at the end of the classification process. Assume that the secret key can be expressed in bytes as $K_{sec} = \{k_1, k_2, \ldots, k_N\}$ for $0 \leq k_i \leq 255$, where N represents the secret key length and i each step of the method. The method assumes sequential classification as most DPA attacks do, which means that the classification is realized byte by byte. This power analysis determines the first byte k_1 of the secret key in the first step and the second byte k_2 in the second step, and so on. The difference between individual steps is in the division of the power traces measured into parts corresponding to the time intervals in which the cryptographic device works with the respective bytes of the secret key. The method is divided into three phases:

- The first phase is the preparation of power consumption patterns, where the attacker has to prepare the training set to train the neural network. The attacker must know the type of the cryptographic module on which he wants to realize the attack.
- The second phase is the preparation and training of the neural network using the power patterns measured in the first phase.
- The third phase is the attack. The attacker measures the power consumption of the device under attack and inserts the measured power trace to the input of the trained neural network. The neural network assigns the probability vector to the power consumption that contains probabilities for all key estimates. The estimate key with the highest probability should be equal to the secret key stored in the device under attack.

It is clear that it is not suitable to measure and classify the power trace corresponding to the whole cryptographic algorithm but it is better to locate some important operations where the cryptographic module works with intermediate result and the secret key. The `AddRoundKey` and `SubBytes` operations represent a suitable place in the power trace of the AES algorithm.

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 `AddRoudnKey` and `SubBytes` operations in the initialization phase of the algorithm. The program allowed incrementing and decrementing the first byte of the secret key (k_1) and indicated this operation by sending the respective value via a serial port to a computer. In [1] and in our experiment, the measurements were focused on the first byte of the secret key but we claim that this power analysis method is able to classify the whole AES secret key from only one measured power trace. Therefore, the term secret key denotes the first byte of the secret key in the following text. 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 the PIC 8-bit microcontroller, and for the power consumption measurement we used the CT-6 current probe and the Tektronix DPO-4032 digital oscilloscope. We used standard operating conditions with 5 V power supply.

A well known fact is that noise always poses the problem during the power consumption measurement. We performed the experimental measurements of a

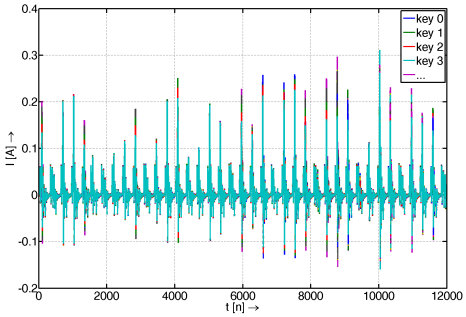

Fig. 1: Original power patterns.

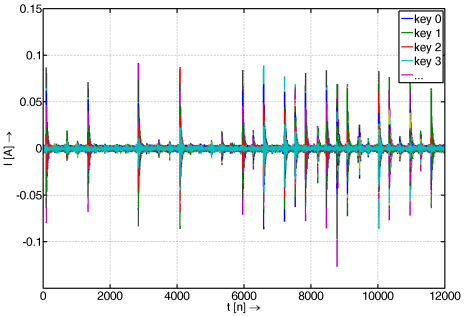

Fig. 2: Preprocessed power patterns.

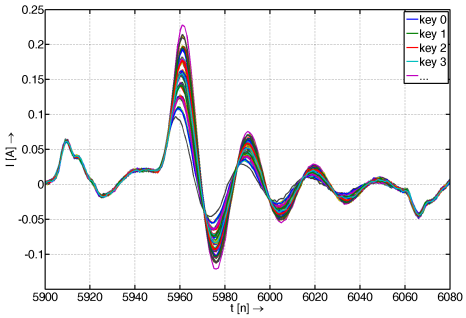

Fig. 3: Detail of original patterns.

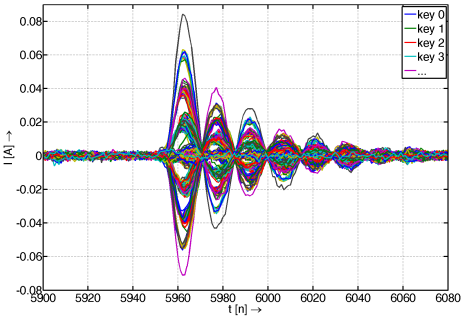

Fig. 4: Detail of preprocessed patterns.

test bed made according to the information provided in [3] and we established that the noise level was distributed according to the normal distribution with the parameters $\mu = 0$mA and $\sigma = 5$mA. Every stored power trace was calculated as an average power trace from ten power traces measured using the digital oscilloscope to reduce electronic noise. More information about the testbed is given in [12, 35]. Our other experiments with power analysis and implementation, for example power consumption measurement of smart phone encrypted data, are reported in [36, 37].

3 Proposed Optimization

The optimization is based on the preprocessing of power traces measured during the first phase of the method, where the training patterns are prepared. During this phase, the attacker tries to obtain the training patterns of power consumption for the `AddRoundKey` and `SubBytes` operations for all variants of the secret key byte k_1 (256 possible variants). Figure 1 shows the power patterns for all values of the secret key cut out from the whole power trace for the first byte, and Fig. 3 shows a detail of the power peak at time $t[n] = 6,000$. From these figures, it is clear that the measured power traces are greatly synchronized and divided into several groups. These power patterns were stored and used for the neural

network training in the original method proposal and implementation [1] (it was used three times 256 power traces for neural network training). We magnified the differences in the power traces measured to improve the classification results. Increased differences were achieved by employing a preprocessing process based on the calculation of average power traces corresponding to every key value. The main principle of preprocessing is described in the following text.

The measured power traces are functions with discrete time. We denote the measured power traces corresponding to every secret key value as $P[i, n]$, where n represents the discrete time $n = \{0, \ldots, 12000\}$, and i represents all possible secret key byte values from 0 to 255. Subsequently, we can calculate an average trace $\bar{A}$ using the following equation:

$$\bar{A}[n] = \frac{1}{256} \sum_{i=0}^{255} P[i, n]. \tag{1}$$

The training patterns for the optimized implementation are calculated as a subtraction of measured traces from the average trace and are denoted as P_D:

$$P_D[i, n] = \bar{A}[n] - P[i, n] = \frac{1}{256} \sum_{i=0}^{255} P[i, n] - P[i, n]. \tag{2}$$

Figure 2 shows the resulting power patterns after preprocessing and Fig. 4 shows the corresponding power peak detail at time $t[n] = 6,000$. The resulting patterns were stored and used for the neural network training in the optimized implementation. If we compare these two sets of patterns, it is clear that after preprocessing the patterns show the places where the power traces are different.

4 Comparison of Classification Results

The neural network was implemented and trained in Matlab using the Netlab neural network toolbox in the same way as described in [1]. The implementation differs only in the preprocessing of power pattern according to the optimization proposal described in Sec. 3. To compare the suitability of optimization, we measured once again the whole set of power traces corresponding to all the secret key values and this set was subsequently analyzed using the created and trained neural network. The measured traces were stored in the matrix and all matrix rows (all power traces) were classified using the neural network. In this manner, we obtained classification results for all possible key values and the first notion of how successful the optimized method is when compared to the original implementation.

The classification of all power traces gave the matrices $\mathbf{R_D}$ of dimension 255×255. The row index corresponds to the value of a secret key K_{sec} and the column index corresponds to the value of a key estimate K_{est}. In other words, the neural network assigned to every measured power trace a probability vector for individual key estimates. Table 1 shows a really small part of the

Table 1: Part of the resulting matrices.

	Original implementation $\mathbf{R}$					Optimized implementation $\mathbf{R_D}$				
⋮	...	...	...	...	...	...	...	...	...	...
2	0.00%	0.00%	6.46%	0.00%	...	0.00%	0.00%	92.86%	0.00%	...
1	0.00%	66.42%	0.00%	0.00%	...	0.00%	99.87%	0.00%	0.00%	...
0	36.77%	0.00%	0.00%	0.00%	...	98.23%	0.00%	0.00%	0.00%	...
K_{sec}/K_{est}	0	1	2	3	...	0	1	2	3	...

resulting matrix $\mathbf{R_D}$ together with the original results matrix $\mathbf{R}$. From Tab. 1 it can be seen that the neural network classified the power trace corresponding to $K_{sec} = 0$ with a probability of 98.23% for the key estimate $K_{est} = 0$ and other estimates with zero probability in the optimized implementation (we do not take into consideration the whole output vector in this demonstration). The neural networks classified the power trace corresponding to $K_{sec} = 0$ with a probability of 36.77% for the key estimate $K_{est} = 0$ in the original implementation. From this small comparison of the results obtained, we can confirm the increase in the probability of correct key estimates. For example, probability estimates for correct key 0 and 1 increased from 36.77% and 66.42% to 98.23% and 99.87% respectively.

The whole matrix $\mathbf{R}$ of classification is shown graphically in Fig. 5 and matrix $\mathbf{R_D}$ is shown in Fig. 6. Each row of the matrix corresponds to the output probability vector, which is the result of power trace classification. Each column contains the probability of an individual key estimate. The main goal of the method is to have the estimate key value equal to the secret key value after classification. In other words, the function $K_{est} = K_{sec}$ is true. The function $K_{est} = K_{sec}$ is visible in both matrices but in $\mathbf{R_D}$ it is much more distinguishable because the correct classified probabilities consist of values between 90% and 100% and thus the line is darker. The graphs also show the reduction of alternative variants of classification and thus the absence of parallel lines with the function $K_{est} = K_{sec}$ in Fig. 6. The graphs displayed in Fig.7 and Fig. 8 confirm this desired property. These graphs show the classification results (output probability vectors) for five chosen secret keys for both implementations. Appropriate probability vectors for the chosen $K_{sec} = 5, 41, 81, 129, 248$ values are distinguished by color and the optimized implementation is shown in Fig. 8 and original implementation in Fig. 7. If we compare the results, for example, for the power trace $K_{sec} = 5$ of the optimized implementation, the increase in the correct key estimate from 35% to 96% is clearly visible while other possible key estimates were fully suppressed. This desired property, i.e. suppressing potential key estimates was confirmed for the other three chosen secret keys $(41, 81, 248)$. For the last chosen power trace corresponding to the secret key 129, alternative key estimates were also suppressed, except one, but the probability of correct key estimate increased from 70% to 90%. From these results, we conclude that

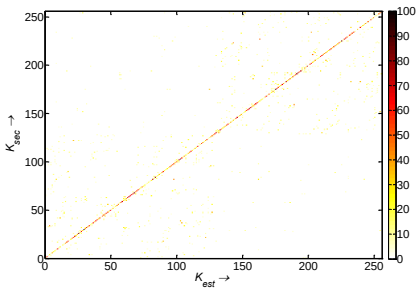

Fig. 5: Graphically depicted matrix $\mathbf{R}$.

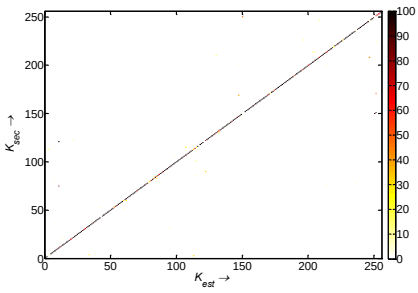

Fig. 6: Graphically depicted matrix $\mathbf{R_D}$.

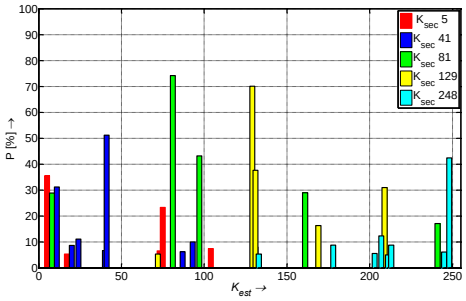

Fig. 7: Probability vector for five secret keys of the original method.

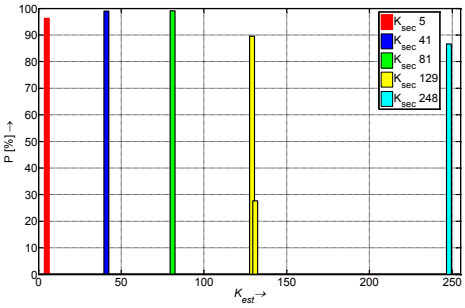

Fig. 8: Probability vector for five secret keys of the optimized method.

proposed optimization allows a significant increase in the classification results because the probability of correct key estimates is increased and the other possible key estimates are suppressed.

On the other hand, a complete suppression of alternative probabilities can be negative, because the probability of a correct key estimate was always the second highest probability for all erroneously classified keys in the original implementation. The attacker would use this feature if it happened that the key was badly classified at the end of the attack. If the optimization suppressed all alternative possibilities of key estimates, similarly like in Fig. 8, the attacker would not be able to try a second key estimate.

However, it is necessary to investigate all selected key estimates from the tested set because during this investigation, the theoretical success rate about 80% was calculated in the original implementation. The main problem of wrongly classified key estimations was the low value of the selected highest probability. Figure 9 shows these selected highest probabilities of power traces corresponding to all the values of the secret key for the original implementation. In other words, it shows which key estimate was classified with the highest probability for a specific power trace. The graph is displayed with two Y-axes for better clarity. The X-axis represents the secret key values and the blue Y-axis shows the probability of the highest selected probability while the red Y-axis corresponds

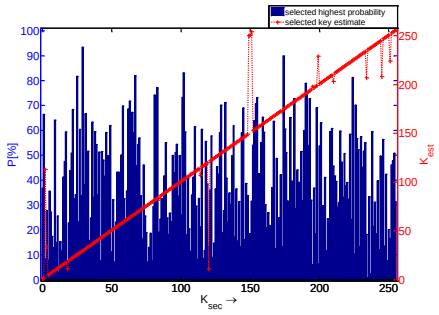

Fig. 9: The highest selected probabilities of original implementation.

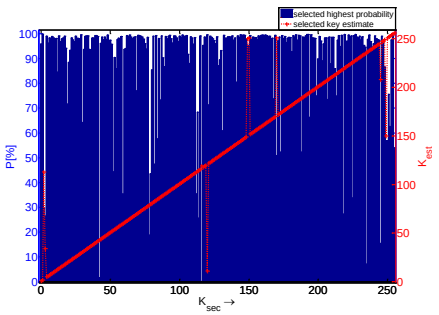

Fig. 10: The highest selected probabilities of optimized implementation.

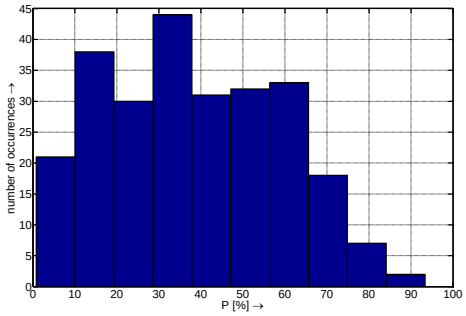

Fig. 11: Histogram of highest probabilities of original implementation.

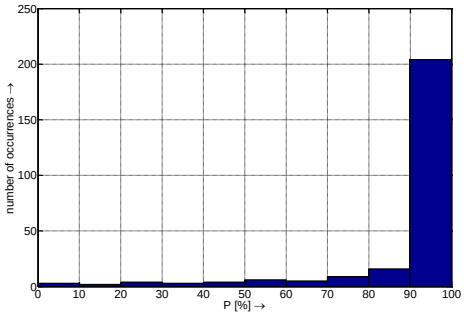

Fig. 12: Histogram of highest probabilities of optimized implementation.

to the chosen key estimate. The shape of the function $K_{est} = K_{sec}$ is again clearly visible and only a few points interrupt the linear progression. From the whole proofing set (256 power traces measured), the neural network classified a wrong key estimate sixteen times in the original implementation. Classification errors occurred for key estimates with low values of the highest probability. The average value of the highest probability which led to the wrong classification was 17%. From these results, the theoretical border of correct classification was established as 20%. For key estimates with a selected highest probability lower than 20%, the probability of wrong clasification is higher. Figure 9 shows that the occurrence of 14%, 18% and 20% probabilities is no exception. Figure 11 displays a histogram of selected highest probabilities for the original implementation. From the histogram, it can be seen that probabilities of up to 10% occurred twenty-one times while the probabilities of $10-20\%$ occurred thirty-eight times, which makes a total of 59 occurrences of all 256 values. The total number of keys potentially predisposed to incorrect classification is about 23%, which means that the original method theoretically works with a success rate of about 80%.

These results obtained from the first implementation were promising but the success rate was not sufficient. This was the main reason why we tried

an optimized implementation to increase the selected highest probabilities and thus reduce the wrongly classified keys. Figure 10 shows the selected highest probabilities for the optimized implementation with the course of the function $K_{est} = K_{sec}$ is almost smooth and containing only nine wrongly classified keys. If we compare these results with the first implementation results, we achieve a decrease in wrong classification from 16 to 9, which corresponds to an improvement of 43%. These results clearly demonstrate the functionality and suitability of preprocessing the power traces measured for classification using a neural network. Nine wrongly classified key estimates correspond to 3.5% of the power traces measured. Therefore, we can declare that the optimized method identified the correct value of the secret key in 96.5% of cases. During repeated tests (another training of the neural network with the identical training set), the optimized method achieved a correct classification of $95 - 98\%$.

Figure 12 displays a histogram of the selected highest probabilities for the optimized implementation. From the histogram, it can be seen that probabilities of 10% to 70% occurred only five times on average. Probabilities of 70% and 80% occurred ten times and fifteen times, respectively. The largest representation in the selected maximum probability is that of the 90% to 100% probabilities, which occurred two hundred and five times. The histogram confirmed the increase of the maximum probabilities, thus increasing the occurrence of the 90% probability. The total number of keys potentially predisposed to wrong classification is reduced from 20% to 5% after optimization.

5 Cross Validation

A ten-times larger set of power consumptions was measured and used for a detailed comparison of the original and optimized method in the same manner as described in [1]. Ten power traces were independently stored for each value of the secret key. The set composed of $2,560$ power traces was classified using neural networks in the same manner as described in the previous sections. The number of wrongly classified key estimates and the overall success rate are given in Tab. 2. The original method achieved a correct secret key classification in 85% and the optimized method achieved a correct secret key classification in 94%. These results confirm the previous results including the correct calculation of the theoretical success rate and the necessity of optimization. We can state that the optimized method achieved results that were better by 10%.

Table 2: Classification results for 2560 power traces.

Method	Number of errors [-]	Success rate [%]
Original implementation	378	**85.23**
Optimized implementation	139	**94.57**

In the original paper, the cross-validation was not used for the verification of the neural network. We decided to compare the original method with the optimized method 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 is a statistical method of evaluating and comparing learning algorithms by dividing data into two segments: one is used to learn or train a model and the other is used to validate the model. In typical cross validation, the training and validation sets must cross-over in successive rounds such that each data point has a chance of being validated against. Our set of 2560 measured power traces consisted of 10 power traces corresponding to every secret key value, therefore we used 9 power traces for neural network training and one for testing in every step of validation. The results of 10-fold cross-validation are summarized in Tab. 3, where err denotes the number of wrongly classified key estimates and $\overline{err}$ denotes the average value of wrong estimates calculated from every step of the cross-validation.

Table 3: Number of errors for 10-fold cross-validation.

Step of cross-validation	1	2	3	4	5	6	7	8	9	10	$\overline{err}$	Success rate [%]
Original implementation $err[-]$	10	5	12	17	8	17	13	14	7	12	11.5	95.71
Optimized implementation $err[-]$	0	0	0	0	1	0	1	0	0	0	0.2	**99.92**

The results obtained reveal that the original implementation is able to classify the secret key with a success rate of around 95%. It is better than the assumption stated above. This difference is caused by the size of the training set. In the original implementation, 3 power traces for every secret key value for neural network training and one for testing were used. In comparison with the cross-validation, 9 power traces for neural network training and one for testing were used. The results of cross-validation confirm the positive impact of optimization on classification results. The optimized method is able to classify the secret key value with almost 100% success rate.

6 Conclusion

In the paper, we presented and realized an optimization method of the power analysis based on multi-layer perceptron. The optimization was based on preprocessing the measured power traces using the calculation of the average trace and the subsequent calculation of the difference power traces. These power patterns were used for neural network training and, naturally, during the attack phase. We compared the classification results of the optimized method with the original implementation and evaluated the positive and negative impact of optimization on classification results.

The proposed optimization allowed a significant improvement in the classification results because the probability of correct key estimates was increased and the other possible key estimates were suppressed. On the other hand, a complete suppression of alternative probabilities can be negative because the attacker is not able to try a second key estimate if the key estimate with the highest probability is wrong.

In the original paper, cross-validation was not used to verify the neural network and thus we compare the original method with the optimized method, using the typical 10-fold cross-validation. The result of cross-validation confirm the positive impact of optimization on classification result. The optimized method is able to classify the secret key value with almost a 100% success rate.

The features of the optimized method can be summarized in the following points:

- optimization is computationally undemanding,
- places where power traces differ can be highlighted,
- probability corresponding to correct key estimations is increased,
- probability corresponding to incorrect key estimations is suppressed,
- number of keys potentially predisposed to wrong classification is reduced,
- negative impact consists in a complete suppression of alternative probabilities.

Acknowledgments. This research work is funded by the Ministry of Industry and Trade of the Czech Republic, project FR-TI4/647. Measurements were run on computational facilities of the SIX Research Center, registration number CZ.1.05/2.1.00/03.0072.

References

1. Martinasek, Z., Zeman, V.: Innovative method of the power analysis. Radioengineering 22(2), 586–594, (2013)
2. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: CRYPTO '99: Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology, London, UK, Springer-Verlag (1999) 388–397
3. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks: Revealing the Secrets of Smart Cards (Advances in Information Security). Springer-Verlag New York, Inc., Secaucus, NJ, USA (2007)
4. Joye, M., Olivier, F.: Side-channel analysis. In: Encyclopedia of Cryptography and Security (2nd Ed.). (2011) 1198–1204
5. Chari, S., Rao, J.R., Rohatgi, P.: Template attacks. In: CHES. (2002) 13–28
6. Rechberger, C., Oswald, E.: Practical template attacks. In: Information Security Applications, 5th International Workshop, WISA 2004, Jeju Island, Korea, August 23-25, 2004, Revised Selected Papers, volume 3325 of Lecture Notes in Computer Science, Springer (2004) 443–457
7. Hanley, N., Tunstall, M., Marnane, W.P.: Using templates to distinguish multiplications from squaring operations. Int. J. Inf. Sec. 10(4) (2011) 255–266

8. Coron, J.S., Naccache, D., Kocher, P.: Statistics and secret leakage. ACM Trans. Embed. Comput. Syst. **3**(3) (August 2004) 492–508
9. Joye, M., Paillier, P., Schoenmakers, B.: On second-order differential power analysis. In: Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Springer (2005) 293–308
10. Herbst, C., Oswald, E., Mangard, S.: An aes smart card implementation resistant to power analysis attacks. In: Applied Cryptography and Network Security, Second International Conference, ACNS 2006, volume 3989 of Lecture Notes in Computer Science, Springer (2006) 239–252
11. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: CHES. (2004) 16–29
12. Martinasek, Z., Clupek, V., Krisztina, T.: General scheme of differential power analysis. In: Telecommunications and Signal Processing (TSP), 2013 36th International Conference on. (2013) 358–362
13. Messerges, T.S., Dabbish, E.A., Sloan, R.H., Messerges, T.S., Dabbish, E.A., Sloan, R.H.: Investigations of power analysis attacks on smartcards. In: In USENIX Workshop on Smartcard Technology. (1999) 151–162
14. Moradi, A., Barenghi, A., Kasper, T., Paar, C.: On the vulnerability of fpga bitstream encryption against power analysis attacks: extracting keys from xilinx virtex-ii fpgas. In: Proceedings of the 18th ACM conference on Computer and communications security. CCS '11, New York, NY, USA, ACM (2011) 111–124
15. Plos, T., Hutter, M., Feldhofer, M.: Evaluation of side-channel preprocessing techniques on cryptographic-enabled hf and uhf rfid-tag prototypes. In Dominikus, S., ed.: Workshop on RFID Security 2008, Budapest, Hungary, July 9-11, 2008. (2008) 114 – 127
16. Kasper, T., Oswald, D., Paar, C.: Side-channel analysis of cryptographic rfids with analog demodulation. In Juels, A., Paar, C., eds.: RFIDSec. Volume 7055 of Lecture Notes in Computer Science., Springer (2011) 61–77
17. Oswald, D., Paar, C.: Breaking mifare desfire mf3icd40: Power analysis and templates in the real world. In Preneel, B., Takagi, T., eds.: CHES. Volume 6917 of Lecture Notes in Computer Science., Springer (2011) 207–222
18. Barenghi, A., Pelosi, G., Teglia, Y.: Improving first order differential power attacks through digital signal processing. In: Proceedings of the 3rd international conference on Security of information and networks. SIN '10, ACM (2010) 124–133
19. Kim, H.M., Kang, D.J., Kim, T.H.: Flexible key distribution for scada network using multi-agent system. Bio-inspired, Learning, and Intelligent Systems for Security, ECSIS Symposium on (2007) 29–34
20. Lian, S., Sun, J., Wang, Z.: One-way hash function based on neural network. CoRR abs/0707.4032 (2007)
21. Wang, Y.h., Shen, Z.d., Zhang, H.g.: Pseudo random number generator based on hopfield neural network. (August 2006) 2810–2813
22. Liu, N., Guo, D.: Security analysis of public-key encryption scheme based on neural networks and its implementing. In: Computational Intelligence and Security, 2006 International Conference on. Volume 2. (nov. 2006) 1327 –1330
23. Mislovaty, R., Perchenok, Y., Kanter, I., Kinzel, W.: Secure key-exchange protocol with an absence of injective functions. Phys. Rev. E 66 (Dec 2002) 066102
24. Fiona, A.H.Y.: ERG4920CM Thesis II Keyboard Acoustic Triangulation Attack. PhD thesis, Department of Information Engineering the Chinese University of Hong Kong (2006)

25. Zhuang, L., Zhou, F., Tygar, J.D.: Keyboard acoustic emanations revisited. In: Proceedings of the 12th ACM conference on Computer and communications security. CCS '05, New York, NY, USA, ACM (2005) 373–382
26. Quisquater, J.J., Samyde, D.: Automatic code recognition for smart cards using a kohonen neural network. In: Proceedings of the 5th conference on Smart Card Research and Advanced Application Conference - Volume 5. CARDIS'02, Berkeley, CA, USA (2002) 6–6
27. Kur, J., Smolka, T., Svenda, P.: Improving resiliency of java card code against power analysis. In: Mikulaska kryptobesidka, Sbornik prispevku. (2009) 29–39
28. Martinasek, Z., Macha, T., Zeman, V.: Classifier of power side channel. In: Proceedings of NIMT2010. (September 2010)
29. Yang, S., Zhou, Y., Liu, J., Chen, D.: Back propagation neural network based leakage characterization for practical security analysis of cryptographic implementations. In: Proceedings of the 14th international conference on Information Security and Cryptology. ICISC'11, Springer-Verlag (2012) 169–185
30. Heuser, A., Zohner, M.: Intelligent machine homicide - breaking cryptographic devices using support vector machines. In: COSADE. (2012) 249–264
31. Bartkewitz, T., Lemke-Rust, K.: Efficient template attacks based on probabilistic multi-class support vector machines. In: Proceedings of the 11th international conference on Smart Card Research and Advanced Applications. CARDIS'12, Springer-Verlag (2013) 263–276
32. Hospodar, G., Gierlichs, B., Mulder, E.D., Verbauwhede, I., Vandewalle, J.: Machine learning in side-channel analysis: a first study. J. Cryptographic Engineering 1(4) (2011) 293–302
33. Lerman, L., Bontempi, G., Markowitch, O.: Side channel attack: an approach based on machine learningn. In: COSADE 2011 - Second International Workshop on Constructive Side-Channel Analysis and Secure Design. (2011) 29–41
34. Oswald, D., Paar, C.: Improving side-channel analysis with optimal linear transforms. In: Proceedings of the 11th international conference on Smart Card Research and Advanced Applications. CARDIS'12, Berlin, Heidelberg, Springer-Verlag (2013) 219–233
35. Martinasek, Z., Zeman, V., Sysel, P., Trasy, K.: Near electromagnetic field measurement of microprocessor. PRZEGLAD ELEKTROTECHNICZNY 89(2a) (2013) 203 – 207
36. Malina, L., Clupek, V., Martinasek, Z., Hajny, J., Oguchi, K., Zeman, V.: Evaluation of software-oriented block ciphers on smartphones. In: Foundations and Practice of Security. Springer (2013)
37. Hajny, J., Malina, L., Martinasek, Z., Tethal, O.: Performance evaluation of primitives for privacy-enhancing cryptography on current smart-cards and smartphones. In: Data Privacy Management and Autonomous Spontaneous Security. Lecture Notes in Computer Science 8247, Springer Berlin Heidelberg (2013)