Uncertainty Analysis in Cryptographic Key Recovery for Machine Learning-Based Power Measurements Attacks

Pasquale Arpaia, *Senior Member, IEEE*, Francesco Caputo, *Member, IEEE*, Antonella Cioffi, Antonio Esposito, *Member, IEEE*, and Francesco Isgrò, *Member, IEEE*

***Abstract*— The present work concerns side-channel attacks on cryptographic devices protected with the advanced encryption standard (AES). In this regard, the assessment of guessing entropy (GE) and the related uncertainty is proposed for machine-learning-based attacks based on power measurements. For the first time, the GE was assessed on the entire key while uncertainty was introduced in the field of side-channel attacks, thus allowing a more rigorous vulnerability test for a device. Notably, a state-of-the-art attack relying on a multilayer perceptron is exploited for classifying power traces leaked from physically accessible devices. A public database was exploited for the sake of results' reproducibility. Thanks to cross-validation, the uncertainty associated with retrieving a single key byte can be quantified and then propagated to the entire key by means of the Monte Carlo method. It is thus shown that when exploiting about 4000 attack records (traces), there is a 10% probability to retrieve the secret key as a whole with less than ten attempts. This implies that a full cryptographic key can be discovered on average ten times for every 100 similar devices by a side-channel attack. This poses security threats particularly relevant in an Internet-of-Things scenario and addresses the need for improved vulnerability testing and proper countermeasures.**

***Index Terms*— Advanced encryption standard (AES), machine learning, masking countermeasure, multilayer perceptron (MLP), power measurements, profiling attacks, side-channel analysis (SCA), template attacks.**

I. Introduction

In 1990s, Kocher [1] and Kocher et al. [2] demonstrated that measuring execution time and power consumption in cryptographic devices can reveal sensitive information. He pioneered new measurement-based attacks consisting of exploiting unintended computation effects, namely, leakages, to retrieve information like the secret key of a cryptographic algorithm. Such attacks are known in literature as "side-channel attacks" [3].

In side-channel attacks, the physical accessibility of the device is required. This is a common condition for paradigms such as Internet of Things or edge computing, where a distributed network of smart devices is exploited [4], [5], [6], [7]. Other than execution time and power consumption, the exploited leakages include heat dissipation and electromagnetic emission. Power analysis is currently considered as the most powerful side-channel attack [8], and several approaches have been proposed.

Nonprofiled attacks include simple [9], differential [2], and correlation [10] power analysis. They consist of acquiring a set of power traces and then applying statistical analysis to retrieve the secret key. Therefore, the attacker exploits the correlation between power consumption and internal state of the device during cryptographic operations.

Profiled attacks, instead, rely on identifying a statistical model for the target device on the basis of compatible devices [11]. The attack consists of comparing the power traces acquired from the target device with the statistical model, thus allowing to find the secret key with a limited number of traces. A *divide and conquer* strategy is usually applied to break down the secret key recovery into separately recovering single bits or bytes (denoted as "sub-keys").

Since 2011, the possibility to enhance profiled side-channel attacks was explored by means of machine learning [12], [13], [14]. This approach involves the identification of a model using preliminary measures (training data) and then the classification of new data through the identified model. The classification output is a class or the probability of possible classes associated with these data [14].

Hospodar et al. [12] considered the execution of a cryptographic algorithm compliant with the advanced encryption standard (AES) [15] and proposed a support vector machine to classify power consumption. Their results demonstrated that the classification-based approach could outperform template attacks through a fine-tuning of hyperparameters. Then, in [16], the concept of *enhanced brute force* was introduced, namely, attempting all possible keys (brute force) while considering the most probable subkeys first.

Manuscript received 1 May 2023; accepted 30 May 2023. Date of publication 12 June 2023; date of current version 28 June 2023. This work was carried out as part of the "Information Technology and Electrical Engineering" project, which was financially supported by the PON "Ricerca e Innovazione" 2014–2020, Azione IV.5, Ministerial Decree n. 1061 of the Italian Ministry of University and Research (MIUR). The Associate Editor coordinating the review process was Dr. Valentina Bianchi. *(Corresponding author: Pasquale Arpaia.)*

Pasquale Arpaia is with the Department of Electrical Engineering and Information Technology (DIETI), the Augmented Reality for Health Monitoring Laboratory (ARHeMLab), and the Centro Interdipartimentale di Ricerca in Management Sanitario e Innovazione in Sanità (CIRMIS), Università degli Studi di Napoli Federico II, 80125 Naples, Italy (e-mail: pasquale.arpaia@unina.it).

Francesco Caputo is with the Department of Electrical Engineering and Information Technology (DIETI), Università degli Studi di Napoli Federico II, 80125 Naples, Italy.

Antonella Cioffi is with STMicroelectronics, 81025 Marcianise, Italy.

Antonio Esposito and Francesco Isgrò are with the Department of Electrical Engineering and Information Technology (DIETI) and the Augmented Reality for Health Monitoring Laboratory (ARHeMLab), Università degli Studi di Napoli Federico II, 80125 Naples, Italy.

Digital Object Identifier 10.1109/TIM.2023.3284933

Martinasek and Zeman [17] proposed a neural network for classifying the bytes of AES keys. They focused on retrieving a single byte and suggested that in case of byte misclassification, the next most probable bytes could be considered. Such a method could potentially allow retrieving a key. Nonetheless, an analysis considering the entire key is missing. Finally, in recent works, the focus has been on improving the classification of power traces by comparing different machine learning algorithms [18], [19], by exploiting deep learning methods such as convolutional neural networks [20], [21], or by exploring different leakage analyses to prepare the training dataset [22]. These works relied on a more complex model when attempting to improve an attack.

Different metrics have been proposed to assess the performance of a model for side-channel attacks [23], [24]. Among them, a widely used metric is the *guessing entropy* (GE), which quantifies the number of guesses needed on average to recover the right (sub-) key in an enhanced brute-force attack [16]. Thus, the greater the GE, the lower the attack effectiveness.

In calculating the GE, no associated uncertainty was estimated to date. However, evaluating the uncertainty would be desirable, especially in the Internet-of-Things scenario, because, when attacking multiple devices, the key could be retrieved (by chance) before the average number of guesses. Thus, quantifying the uncertainty would allow more rigorous vulnerability tests for a device.

Uncertainty quantification in machine learning has been receiving increasing attention in recent literature [25], [26], [27]. This appears crucial especially when machine learning is applied in healthcare [28], [29]. Widely exploited methods are the "Monte Carlo dropout" [25], or Bayesian neural networks [30]. However, these probabilistic approaches put some constraints on model architectures, e.g., they might require a Bayesian network. On the other hand, misclassification probability could be exploited for any model with probabilistic output to assess the uncertainty of a predictive performance [31].

The present work proposes an assessment of GE and associated uncertainty for machine-learning-based side-channel attacks. Notably, the proposed approach extends uncertainty quantification to profiling attacks by directly exploiting the results of a cross-validation already needed during the model validation. The investigation was carried on as a function of the number of attack traces. Both subkeys and the entire key were considered, and the Monte Carlo method was exploited to propagate the misclassification probability of single bytes to the uncertainty of the entire key.

The approach was applied to a public dataset, and the attack was based on a state-of-the-art multilayer perceptron. In doing so, the present results can be reproduced and extended. In particular, the final proposal allows to quantify the uncertainty of the device's vulnerability before an attack to facilitate the vulnerability assessment. In the remainder of this article, Section II discusses the proposed method, while Section III reports the experimental results.

II. Method

In this section, we introduce basic ideas and highlight the main contributions of this study. Then, an attack relying on measured power traces is presented. Finally, a method is proposed for assessing the uncertainty associated with the guessing of the secret key. This involves propagating the uncertainty of the GE from single bytes to the key as a whole.

A. Basic Ideas

A machine-learning-based side-channel attack was exploited to retrieve the secret key of the cryptographic device from measured power traces. The AES-128 was specifically considered in the present work. Therefore, the secret key $k^\star$ consists of 16 bytes. Indeed, retrieving the entire secret key would require dealing with 2^{128} classes. Meanwhile, when classifying each byte singularly, $2^8 = 256$ classes are tackled at a time. These classes are still many for a clear-cut classification, but the key discovery can be enhanced (see Section II-B).

The state-of-the-art of AES implementations also involve countermeasures. In this article, the *Boolean masking* was considered. This is a popular data obfuscation scheme that conceals a sensitive information value v, e.g., a secret key of cryptography algorithms, through a value m called *mask*, by calculating $v_m = v \oplus m$.

The attack performance is commonly assessed by GE. To obtain this metric, an array with probabilities associated with each possible key value (*guessing vector* $\mathbf{g}$) is needed as classification output. Then, the *guessing vector* is sorted in descending order of probability, and the position of the correct key is identified (key rank). Finally, the GE is the mean key rank over multiple experiments

$$\mathrm{GE} = \frac{1}{N}\sum_{i=1}^{N} \mathrm{rank}_{k^\star}(\mathbf{g}). \tag{1}$$

In the present work, the GE was assessed as a function of the number of attack traces. In principle, the more the attack traces, the easier the secret key retrieving should be. However, no evidence has yet been provided regarding the entire key. Hence, as a first theoretical improvement, both the single key bytes (subkey) and the entire key were investigated.

In addition, when attacking multiple devices, the secret key could be occasionally discovered before or after the average guesses. Thus, as a second theoretical contribution, the tradeoff between the GE and the number of attack traces was also studied in relation with associated uncertainty.

The whole study ultimately investigates whether a cryptographic device can be penetrated by measuring few power traces during the attack and/or trying a reasonable low number of key values. This aims to improve vulnerability tests for characterizing the device's security.

B. Machine-Learning-Based Attack

Machine learning, and more specifically deep learning, is widely used for pattern recognition when recovering the bytes associated with acquired power traces [14], [20]. Recent literature indicates that convolutional neural networks should be preferred in the presence of desyncronization or jitter in power traces [32]. If this is not the case, a simpler network like the multilayer perceptron is similarly effective [14].

In principle, the parameters and hyperparameters of this neural network model would be identified, thanks to a set of

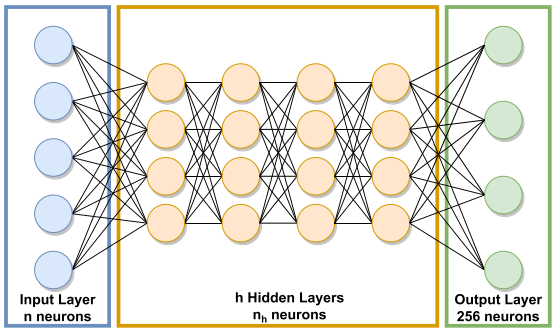

Fig. 1. Architecture of the proposed neural network. The model hyperparameters must be identified for an optimal attack.

power traces and their corresponding byte value. Nonetheless, recent literature suggests that 1) it is not efficient to use entire traces for the training [33] and 2) power traces cannot be analyzed independently of the plaintext [34], namely, the clear message in input to the cryptographic device.

Therefore, the power traces were preprocessed to find the subset of samples with the most significant information for the target byte [points of interest (POIs)] [33] and then an intermediate value for each trace was calculated as

$$v_{p,i} = \text{Sbox}(k_i \oplus p_i) \oplus r_i \tag{2}$$

where k_i $(i = 0, 1, \ldots, 15)$ is the key byte to attack, p_i is the corresponding byte of the plaintext, r_i is the corresponding byte of the mask, $\oplus$ is the XOR operation associated with the *AddRoundKey* step in the first round of the AES algorithm and then with the masking, and Sbox() represents the bytes' substitution (*SubByte*) step [15].

The POIs were identified by the signal-to-noise ratio (SNR) proposed in [32]. In detail, for each byte of the key (target byte), the SNR was estimated by grouping power traces in accordance with the intermediate values. The traces of the same group were averaged to obtain a single mean trace per intermediate value. Instead, multiple noise traces were derived per each intermediate value as a difference between a trace and the mean trace. Finally, for a specific target byte, the SNR was estimated as the ratio between the variances of the mean and noise traces

$$\text{SNR}_j = \frac{\text{var}[S_j]}{\text{var}[N_j]} \tag{3}$$

where j indicates a sample of the trace, var[] indicates the variance among different intermediate values, S_j are the mean traces, and N_j are the noise traces.

Ultimately, a set of POIs with associated intermediate values were exploited to train the multilayer perceptron (Fig. 1). It comprises an input layer with a number of neurons s corresponding to the input samples from a power trace, h hidden layers with the same number of neurons per layer, and an output layer with 256 neurons associated with each possible value of a byte. Each output neuron returns the probability that the input power trace is associated with a specific byte value (i.e., from 0 to 255).

Once the POIs were estimated for each byte, the hyperparameters of the neural network had to be identified. These included, first, the number of layers, the number of neurons per layer, and the activation functions of the neurons. In addition, different numbers of epochs, different loss functions, and different batch sizes were investigated during the following training phase.

After identifying the hyperparameters, the model was trained by exploiting the power traces with associated intermediate values and using the *categorical cross-entropy* cost function for the backpropagation algorithm.

Next, in the attack phase, the model identified for each key byte returns probabilities for all the possible values associated with an input trace. After sorting the guessed byte values by decreasing probability, the number of attempts corresponds to the position of the actual byte value. More than a single trace can also be used to improve the result. In such a case, literature considers joint probabilities of byte values for different traces. Doing so typically decreases the number of attempts to retrieve the correct byte value.

Log-likelihood is commonly used, i.e.,

$$\text{rank}_{k^\star}(\mathbf{g}^*) = \text{rank}_{k^\star}\left[\sum_{t=1}^{T} \log(\mathbf{g}_t)\right] \tag{4}$$

where $\mathbf{g}^*$ is the *guessing vector* with joint probabilities, $\mathbf{g}_t$ is the *guessing vector* associated with a single trace t, and T is the total number of traces used in the attack.

The entire key can be finally retrieved from the probabilities of single-byte values. The idea is that the first key attempt is composed of the 16 most probable byte values. In the following attempts, the less probable values are hence considered in descending order of probability until the secret key is discovered [17]. Hence, the number of attempts to discover the entire key is obtained by combining the attempts for single bytes, as discussed in Section II-C. In doing that, the machine learning approach can enhance the key retrieving process with respect to a brute-force attack.

C. Uncertainty Assessment

Section II-B presented the training of models for guessing the values of single subkeys (bytes). For each model, it is possible to obtain an average GE and the associated uncertainty. This was notably done through cross-validation on training data. Through that, the aim was to approximate the probability distribution of subkey guessing ranks. Hence, the training data were split into n folds, and only $n - 1$ were used for actual training while the remaining one was used for testing.

This training–test step was repeated n times. Per each iteration, the rank of the keys under test was obtained as a function of the number of traces. Then, average μ and standard deviation σ were calculated across the iterations.

The cross-validation allows us to calculate a mean rank (i.e., GE) and an associated uncertainty per each model. Type A evaluation was exploited for the uncertainty, i.e.,

$$u_A = \frac{\sigma}{\sqrt{n}}. \tag{5}$$

This could be repeated for the models associated with different bytes and as a function of the number of attack traces.

The results obtained for a single byte were then combined to achieve the GE of the entire key, along with its uncertainty. The ratio was to multiply the number of guesses for discovering the single byte. However, instead of considering a single GE value per each byte (e.g., the mean), values were selected randomly from the probability distributions associated with each byte. These were assumed Gaussian with mean μ_k equal to the GE of the key byte k and the associated standard deviation σ_k.

The Monte Carlo method was thus exploited to obtain the probability distribution of the GE associated with the entire key. It is worth noting that although assuming Gaussian distributions for the ranks of single bytes, the final distribution is generally non-Gaussian. Therefore, using the standard deviation as a measure of uncertainty may not be appropriate. For this reason, the resulting distributions were analyzed by looking at histograms of occurrences, whose number corresponds to the iterations of the Monte Carlo analysis. Notably, the number of iterations has to properly represents the distribution while minimizing the computation time. Therefore, the analysis was stopped when the variation in the distribution's median and interquartile range was less than 5% between two consecutive iterations.

The proposal as a whole is schematically summarized in Fig. 2. It is worth remarking that for each byte, the SNR analysis allows to select different POIs from the same power traces. Then, power traces are split in different folds F_i. Different models M_j are trained with different data subsets and each model produces a different rank R_k. Hence, the rank uncertainty on a single byte can be determined with cross-validation, while the uncertainty for the entire key needs the Monte Carlo analysis.

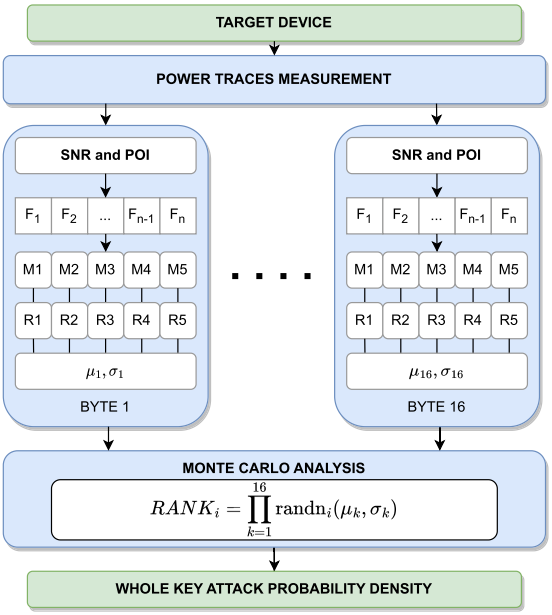

Fig. 2. Schematic illustration of the proposed method. SNR: signal-to-noise ratio. F_i: split of the dataset. M_j: neural network model. R_k: rank value. RANK_i: entire key rank for the ith Monte Carlo iteration. randn: random number from a normal distribution associated with the byte k.

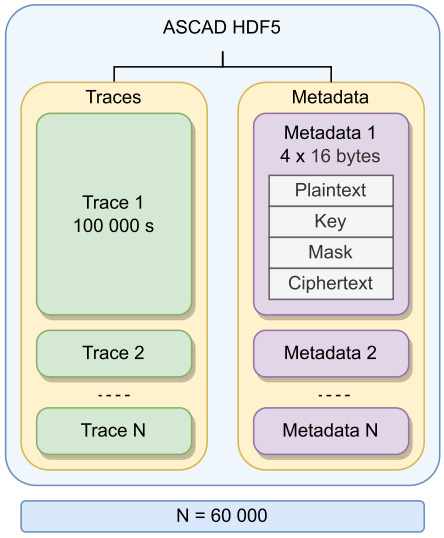

Fig. 3. Data structure of the ASCAD database stored with the HDF5 hierarchical container format.

III. Results

This section first introduces the public data to which the proposed method was applied. The exploited traces resulted preprocessed to avoid jitter and desynchronization. Then, it reports the results in terms of GE and uncertainty of the attacks. Both the single-byte case and the entire key are considered, and the discussion is carried on by considering both the profiling and the attack phases. Moreover, some details on model identification are discussed.

A. Data

The public dataset exploited in this work was the ASCAD database [32], [35]. This constitutes a suitable framework for reproducing and improving the already existing approaches and models. The ASCAD database is composed of power traces. These were measured through the electromagnetic radiation emitted by the ATMega8515 microcontroller during the first round of an AES encryption. The acquisition produced a dataset with 60 000 traces of 100 000 samples. Traces and associated metadata were stored through the Hierarchical Data Format version 5 (HDF5). This is a multipurpose hierarchical container format capable of storing large numerical datasets with their metadata [36].

Fig. 3 shows the hierarchy of the ASCAD database. Notably, this is made of two groups: traces and metadata. The *traces* group contains the raw traces arranged as arrays of samples. The *metadata* group contains, per each trace, information regarding the input plaintext, the secret key, the adopted mask, and the output ciphertext. These are arranged as four vectors of 16 bytes. Indeed, there is a one-to-one correspondence between traces and metadata structures, and the structures are in the same positional order of the traces group.

In accordance with the described method, the raw traces were preprocessed to obtain the actual POIs in input to the model. Meanwhile, the metadata were combined to estimate the intermediate value used as labels for each trace. This was done by considering one byte of the key per time. Hence,

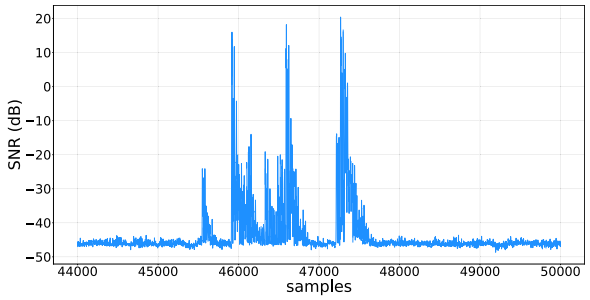

Fig. 4. SNR for the third byte of the AES key under attack.

16 structures similar to the one of Fig. 3 were created. A single structure, associated with a single byte, contained N subsets of s POIs taken from the 100 000 samples of each trace and N intermediate values.

It is worth remarking that the traces and metadata were split in different groups for training, validation, and test. Notably, the first 50 000 traces were used for training and validation, while the remaining 10 000 traces were used as an independent test set. Then, as detailed later, the 50 000 traces were in turn split in fivefolds of 10 000 traces for cross-validation.

B. Models Identification

The classification models for attacking a single byte of the key were identified by searching for optimal hyperparameters. As a first step, the POIs were identified by means of the SNR of (3). Fig. 4 reports an example of SNR estimated for the third byte of the key when using (2) to estimate the intermediate value. The values are reported in decibel (dB) and some peaks can be observed. These are representative of high variance points for the signal.

The specific POIs were identified by considering a window of samples wide enough to contain the peaks. For instance, in the example of Fig. 4, the selected windows was [45 400; 47 600]. Such a window contains all the peaks above the baseline SNR values, which are around −45 dB in the current case. A compatible choice for the POIs can be found in the work introducing the ASCAD database [32]. Nonetheless, instead of selecting a prefixed number of samples, the current work considered time intervals enclosing the peaks with high SNR.

The number s of neurons at the input layer is consequently fixed by choosing the POIs. Then, different combinations were investigated by means of a random search to choose the number of hidden layers and the number of neurons per layer. The search was done by testing a number of hidden layers from 3 to 6 and a number of neurons per layer spanning from 100 to 300 with step 100. The optimal performance in terms of training and validation accuracy was achieved using four hidden layers with 200 neurons each.

Concerning activation functions for the neurons, this work took into account the tanh, the *ReLU*, and the *Softmax*. Empirical evidence showed that the *ReLU* activation function was the optimal choice for input and hidden layers, while the *Softmax* activation function was selected for the output layer. This is in accordance with typical choices for activation functions in artificial neural networks.

TABLE I
ATTEMPTED AND OPTIMAL VALUES FOR MODEL HYPERPARAMETERS

hyperparameter	attempted values	optimal value
hidden layers	3, 4, 5, 6	4
neurons per layer	100, 200, 300	200
number of epochs	100, 200, 500, 1000	200
activation function	*tanh, ReLU, Softmax*	*ReLU* (input/hidden) *Softmax* (output)
loss function	*mean squared error, categorical cross-entropy*	*categorical cross-entropy*
optimizer	*RMSprop*, *Adam*	*RMSprop*

Different training epochs were also evaluated, namely, 100, 200, 400, and 1000. Among them, the evidence showed that there is no significant increase in training accuracy when using more than 200 epochs. Two loss functions and two optimizers were then evaluated. The *categorical cross-entropy* was the best choice for a greater training accuracy reached. The *RMSprop* optimizer was used for a best validation accuracy obtained. Finally, the learning rate was assumed fixed at 10^{-5} as suggested by guides on *RMSprop*.

The attempted model hyperparameters and the respective optimal values for the third byte of the key are summarized in Table I. Note that the third byte was randomly considered among the bytes of the key as one of the 14 bytes where the masking was applied too (worst case). For computational reasons, these same hyperparameters were also used for the other bytes of the key. Nonetheless, a further optimization was attempted whenever the resulting performance was unsatisfactory.

C. Profiling on Single Bytes

Once hyperparameters were identified, a model had to be trained for each byte before carrying out an attack (profiling phase). As discussed above, cross-validation was exploited to train and test the model multiple times and estimate its attack performance in terms of GE. Notably, a fivefold cross-validation was adopted. This section reports three representative cases of attacks on different bytes.

The mean rank assessed on the first byte of the key is flat on the minimum value, i.e., mean 1 and standard deviation 0. Therefore, the attack on this byte always reveals its value in a single attempt. This result is explained by the fact that there is no masking countermeasure in the ASCAD database for the first (and for the second byte too). Indeed, the very same result was also obtained on the second byte of the key. Similar results were observed on unprotected AES [33].

Ultimately, the model identified on a byte with masking effectively attacked bytes without countermeasures. It is worth remarking though that the model had to be trained on the specific byte before performing the attack.

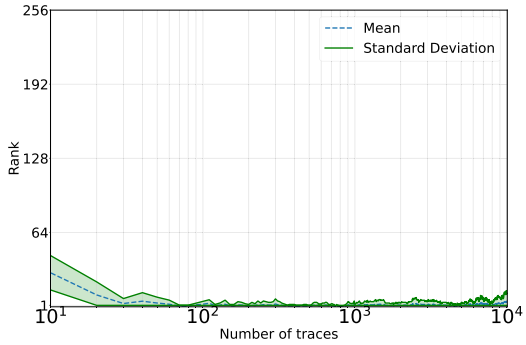

Fig. 5. Mean rank and associated standard deviation resulting from cross-validation considering the third byte of the AES key.

Fig. 5 shows the results of cross-validation applied to the third byte of the AES key. It is worth remembering that the ASCAD database involves a masking countermeasure for this byte. The mean rank (i.e., GE) and associated standard deviation are reported as a function of the number of test traces. This time the results demonstrate that guessing the third byte is almost random when exploiting a few attack traces, while the model can guess the exact byte value in a single attempt when the number of attack traces increases. The result is in accordance with previous works. Indeed, the same trend was observed with optimal convolutional neural networks in [32] and [37]. Moreover, the assessed uncertainty can explain the performance of a multiple-input MLP [38].

An attentive reader should have also noted that the minimum value for a rank is saturated at 1 in the figure because at least one attempt must be made to guess a byte value. Hence, the bottom of Fig. 5 and of following figures was clipped at 1. For a similar reason, the upper part of these figures is saturated at 256, which is the maximum possible number of attempts.

Similar results were obtained for other bytes, i.e., GE was minimized as the number of exploited attack traces increased. The only exception was found for the 13th byte of the AES key. Therefore, as a third representative case, Fig. 6(a) shows the result of cross-validation on byte 13 of the AES key. This byte was masked too. The mean rank and the associated standard deviation demonstrate that the number of guessing attempts remains halfway between the best and worst rank values without any beneficial effects deriving from more attack traces.

As commented in [39], this evidence suggested that the optimal hyperparameters of Table I were a bad choice for such a byte, and therefore an attempt to reoptimize hyperparameters was made. The result after optimization is thus reported in Fig. 6(b). This result appears compatible with the ones exemplified in Fig. 5, and it was achieved by increasing the number of hidden layers from 4 to 6 and the number of neurons per layer from 200 to 300.

Ultimately, the uncertainty assessment of these results allows a better vulnerability analysis with respect to cited literature, where the only mean rank is reported for each attack.

D. Attack on Single Bytes and Entire Key

After assessing the GE of the models by cross-validation, the final models were trained using the whole subset of training traces byte by byte.

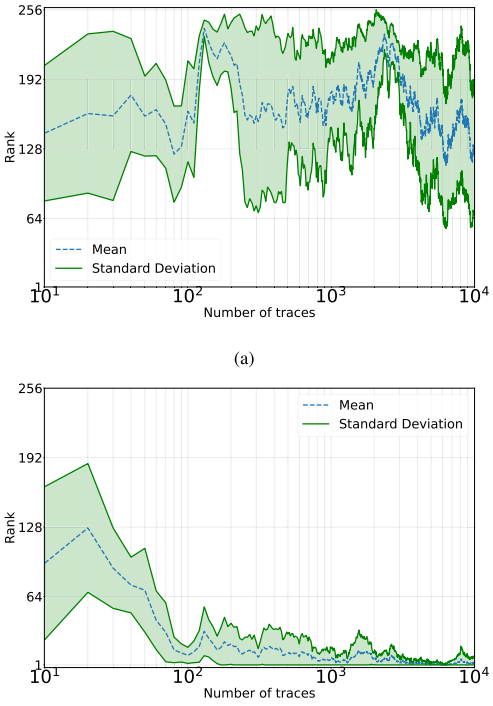

Fig. 6. Mean rank and associated standard deviation resulting from cross-validation considering the 13th byte of the AES key. (a) Nonoptimized model. (b) Optimized model.

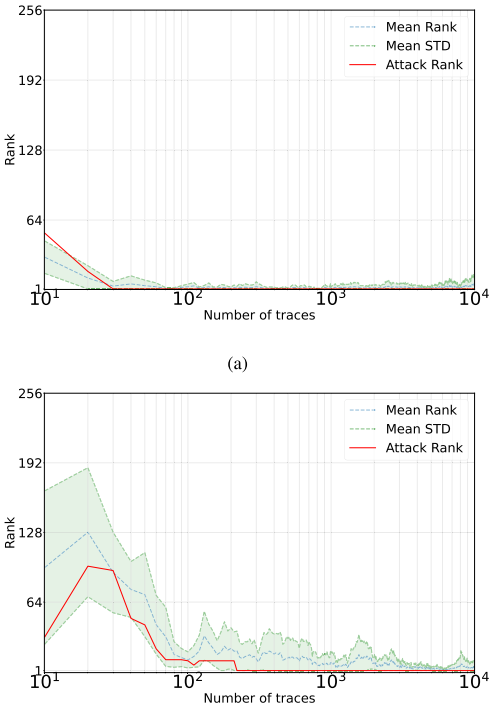

Fig. 7. Validation of the attack rank obtained during the profiling phase for (a) third byte and (b) 13th byte of the AES key.

They could be thus used to attack the target byte by exploiting the remaining independent traces of the test set. The rank as a function of the number of attack traces was obtained by performing such attacks. Each of these curves had to be compared with the band resulting from cross-validation.

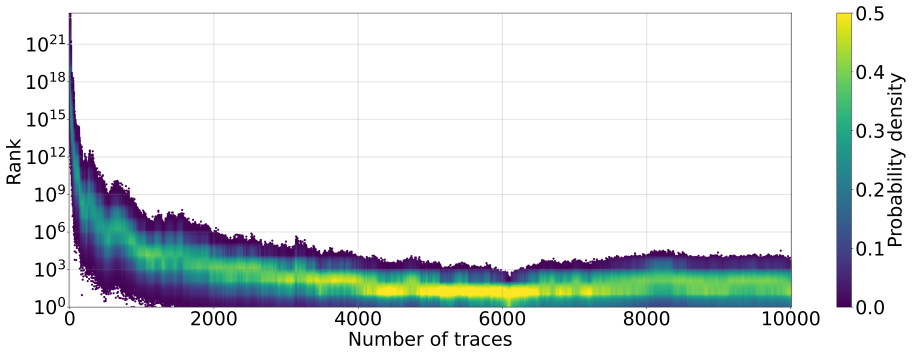

Fig. 8. Scatter plot of occurrences for AES key ranks obtained with the Monte Carlo method as a function of the number of attack traces. The color represents the probability density of rank values.

In Fig. 7(a) and (b), an example of attack performed on byte 3 and byte 13, respectively, is shown.

The mean rank and standard deviation obtained with cross-validation are recalled with a shaded band. Instead, the red line represents the rank of an attack performed using the respective byte model. It can be seen that the attack ranks are basically within the band. Therefore, this validates the model performance estimated during profiling.

Given that, the performance of an attack on the entire key could be investigated with the Monte Carlo method. A general-purpose laptop was exploited and, in accordance with the proposed criterion, the analysis stopped at 100 000 iterations (about 20 min). The results are reported in Fig. 8. This shows that with at least 4000 attack traces, the entire key can be discovered in less than 10^6 attempts. This must be compared with the total number of possible key values, which is in the order of 10^{38}. Moreover, there is a relevant probability of guessing the entire key in few attempts. The last aspect is highlighted in Fig. 8 with a color map representing probability density for rank values.

From the results of the Monte Carlo analysis, the histograms of occurring ranks in the case of 10, 4000, and 10 000 attack traces were particularly focused. These correspond to the values of the scatter plot of Fig. 8 given the respective number of attack traces. As an interesting figure of merit, the resulting probabilities for guessing the entire AES key in less than ten attempts were 0%, 10%, and 19%, respectively.

Overall, the results demonstrate the vulnerability of cryptographic devices to machine-learning-based attacks and quantify the probability of discovering the entire AES key through a side-channel attack. This appears especially relevant for an Internet-of-Things scenario and it suggests a limit for the number of similar devices to be installed when willing to prevent cyber-attacks.

IV. Conclusion

This article has presented a profiled attack employing machine learning for power analysis attacks on cryptographic devices. In particular, its main contribution consisted of estimating an uncertainty for attack ranks associated with the discovery of a single byte and then the entire AES-128 key. Cross-validation allowed such estimation for single bytes, while attacks on independent power traces validated the rank intervals identified by the mean and standard deviation of ranks in the profiling phase. Then, the ranks associated with discovering the entire AES-128 key were calculated with the Monte Carlo method.

The approach of the present work relies on the peculiarities of a side-channel attack, which makes it possible to extend and specifically adapt the already existing literature approaches. The results have demonstrated that there is a 10% probability to retrieve the secret key as a whole with less than ten attempts when exploiting about 4000 attack records (traces). Note that the numeric results necessarily depend on the exploited dataset, which is strictly related to the target devices. However, such a result generally implies that security issues hold in an Internet-of-Things scenario, where multiple similar devices are installed and physically accessible, and that the proposed method allows better vulnerability testing to adopt stronger countermeasures to side-channel attacks.

Future works may also investigate different approaches for uncertainty quantification and compare the proposed method with the advantages and disadvantages of the state-of-the-art approaches.

Acknowledgment

The authors would like to thank Alfonso del Giudice for his preliminary work on the topic.

References

[1] P. C. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," in *Proc. Annu. Int. Cryptol. Conf.* Cham, Switzerland: Springer, 1996, pp. 104–113.

[2] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Proc. Annu. Int. Cryptol. Conf.* Berlin, Germany: Springer, 1999, pp. 388–397.

[3] P. Arpaia, F. Bonavolontá, and A. Cioffi, "Problems of the advanced encryption standard in protecting Internet of Things sensor networks," *Measurement*, vol. 161, Sep. 2020, Art. no. 107853.

[4] O. Choudary and M. G. Kuhn, "Template attacks on different devices," in *Proc. Int. Workshop Constructive Side-Channel Anal. Secure Design*. Cham, Switzerland: Springer, 2014, pp. 179–198.

[5] V. Adat and B. B. Gupta, "Security in Internet of Things: Issues, challenges, taxonomy, and architecture," *Telecommun. Syst.*, vol. 67, no. 3, pp. 423–441, Mar. 2018.

[6] H. A. Abdul-Ghani and D. Konstantas, "A comprehensive study of security and privacy guidelines, threats, and countermeasures: An IoT perspective," *J. Sensor Actuator Netw.*, vol. 8, no. 2, p. 22, Apr. 2019.

[7] Y. Xiao, Y. Jia, C. Liu, X. Cheng, J. Yu, and W. Lv, "Edge computing security: State of the art and challenges," *Proc. IEEE*, vol. 107, no. 8, pp. 1608–1631, Aug. 2019.

[8] H. J. Mahanta, A. K. Azad, and A. K. Khan, "Power analysis attack: A vulnerability to smart card security," in *Proc. Int. Conf. Signal Process. Commun. Eng. Syst.*, Jan. 2015, pp. 506–510.

[9] R. Mayer-Sommer, "Smartly analyzing the simplicity and the power of simple power analysis on smartcards," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst.* Cham, Switzerland: Springer, 2000, pp. 78–92.

[10] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst.* Cham, Switzerland: Springer, 2004, pp. 16–29.

[11] S. Chari, J. R. Rao, and P. Rohatgi, "Template attacks," in *Cryptographic Hardware and Embedded Systems—CHES 2002*, B. S. Kaliski, Ç. K. Koç, and C. Paar, Eds. Berlin, Germany: Springer, 2003, pp. 13–28.

[12] G. Hospodar, B. Gierlichs, E. De Mulder, I. Verbauwhede, and J. Vandewalle, "Machine learning in side-channel analysis: A first study," *J. Cryptograph. Eng.*, vol. 1, no. 4, pp. 293–302, Dec. 2011.

[13] L. Lerman, R. Poussier, O. Markowitch, and F.-X. Standaert, "Template attacks versus machine learning revisited and the curse of dimensionality in side-channel analysis: Extended version," *J. Cryptograph. Eng.*, vol. 8, no. 4, pp. 301–313, Nov. 2018.

[14] B. Hettwer, S. Gehrer, and T. Güneysu, "Applications of machine learning techniques in side-channel attacks: A survey," *J. Cryptograph. Eng.*, vol. 10, no. 2, pp. 135–162, Jun. 2020.

[15] J. Daemen and V. Rijmen, "AES proposal: Rijndael," in *The Rijndael Block Cipher*. Gaithersburg, MD, USA: National Institute of Standards and Technology, 1999.

[16] L. Lerman, G. Bontempi, and O. Markowitch, "Side channel attack: An approach based on machine learning," Center Adv. Secur. Res., Darmstadt, Germany, vol. 29, 2011.

[17] Z. Martinasek and V. Zeman, "Innovative method of the power analysis," *Radioengineering*, vol. 22, no. 2, pp. 586–594, 2013.

[18] Z. Martinasek, V. Zeman, L. Malina, and J. Martinasek, "K-nearest neighbors algorithm in profiling power analysis attacks," *Radioengineering*, vol. 25, no. 2, pp. 365–382, Apr. 2016.

[19] L. Chang, Y. W. Wei, S. He, and X. Pan, "Research on side-channel analysis based on deep learning with different sample data," *Appl. Sci.*, vol. 12, no. 16, p. 8246, 2022.

[20] T. Kubota, K. Yoshida, M. Shiozaki, and T. Fujino, "Deep learning side-channel attack against hardware implementations of AES," *Microprocessors Microsystems*, vol. 87, Nov. 2021, Art. no. 103383.

[21] D. Kwon, S. Hong, and H. Kim, "Optimizing implementations of non-profiled deep learning-based side-channel attacks," *IEEE Access*, vol. 10, pp. 5957–5967, 2022.

[22] F. Hu, H. Wang, and J. Wang, "Multi-leak deep-learning side-channel analysis," *IEEE Access*, vol. 10, pp. 22610–22621, 2022.

[23] L. Masure, C. Dumas, and E. Prouff, "A comprehensive study of deep learning for side-channel analysis," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2020, no. 1, Nov. 2019, pp. 348–375. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/8402

[24] M. Méndez Real and R. Salvador, "Physical side-channel attacks on embedded neural networks: A survey," *Appl. Sci.*, vol. 11, no. 15, p. 6790, Jul. 2021.

[25] M. Abdar et al., "A review of uncertainty quantification in deep learning: Techniques, applications and challenges," *Inf. Fusion*, vol. 76, pp. 243–297, Dec. 2021.

[26] H. Al Osman and S. Shirmohammadi, "Machine learning in measurement—Part 2: Uncertainty quantification," *IEEE Instrum. Meas. Mag.*, vol. 24, no. 3, pp. 23–27, May 2021.

[27] J. Gawlikowski et al., "A survey of uncertainty in deep neural networks," 2021, *arXiv:2107.03342*.

[28] G. Carneiro, L. Z. C. T. Pu, R. Singh, and A. Burt, "Deep learning uncertainty and confidence calibration for the five-class polyp classification from colonoscopy," *Med. Image Anal.*, vol. 62, May 2020, Art. no. 101653.

[29] A. A. Abdullah, M. M. Hassan, and Y. T. Mustafa, "A review on Bayesian deep learning in healthcare: Applications and challenges," *IEEE Access*, vol. 10, pp. 36538–36562, 2022.

[30] Y. Kwon, J.-H. Won, B. J. Kim, and M. C. Paik, "Uncertainty quantification using Bayesian neural networks in classification: Application to biomedical image segmentation," *Comput. Statist. Data Anal.*, vol. 142, Feb. 2020, Art. no. 106816.

[31] C. L. M. Morais, K. M. G. Lima, and F. L. Martin, "Uncertainty estimation and misclassification probability for classification models based on discriminant analysis and support vector machines," *Analytica Chim. Acta*, vol. 1063, pp. 40–46, Jul. 2019.

[32] R. Benadjila, E. Prouff, R. Strullu, E. Cagli, and C. Dumas, "Deep learning for side-channel analysis and introduction to ASCAD database," *J. Cryptograph. Eng.*, vol. 10, no. 2, pp. 163–188, Jun. 2020.

[33] U. Rioja, L. Batina, J. L. Flores, and I. Armendariz, "Auto-tune POIs: Estimation of distribution algorithms for efficient side-channel analysis," *Comput. Netw.*, vol. 198, Oct. 2021, Art. no. 108405.

[34] A.-T. Hoang, N. Hanley, and M. O'Neill, "Plaintext: A missing feature for enhancing the power of deep learning in side-channel analysis? Breaking multiple layers of side-channel countermeasures," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2020, no. 4, pp. 49–85, Aug. 2020. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/8677

[35] *ASCAD (ANSSI SCA Database)*. Access: Sep. 26, 2022. [Online]. Available: https://github.com/ANSSI-FR/ASCAD

[36] *The HDF5 Library and File Format*. Access: Sep. 26, 2022. [Online]. Available: https://www.hdfgroup.org/solutions/hdf5/

[37] G. Zaid, L. Bossuet, A. Habrard, and A. Venelli, "Methodology for efficient CNN architectures in profiling attacks," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2020, no. 1, pp. 1–36, Nov. 2019. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/8391

[38] H. Feng, J. Zhou, W. Lin, Y. Zhang, and Z. Qu, "Multiple-input, multilayer-perception-based classification of traces from side-channel attacks," *Computer*, vol. 53, no. 8, pp. 40–48, Aug. 2020.

[39] L. Wu et al., "On the attack evaluation and the generalization ability in profiling side-channel analysis," Cryptol. ePrint Arch., Digit. Secur. Group, Delft Univ. Technol., Radboud Univ., Nijmegen, The Netherlands, Tech. Rep., 2020.

Pasquale Arpaia (Senior Member, IEEE) received the M.S. and Ph.D. degrees in electrical engineering from the University of Naples Federico II, Naples, Italy, in 1988 and 1992, respectively.

He is a Full Professor of instrumentation and measurements with the University of Naples Federico II.

Francesco Caputo (Member, IEEE) received the M.S. degree in electronic engineering from the University of Naples Federico II, Naples, Italy, in 2021.

His main research activities focus on cybersecurity for the IoT-embedded devices.

Antonella Cioffi received the M.S. degree in electronic engineering and the Ph.D. degree from the University of Naples Federico II, Naples, Italy, in 2018 and 2022, respectively.

Her research activities focus on side-channel attacks and vulnerability tests.

Antonio Esposito (Member, IEEE) received the M.S. degree in electronic engineering from the University of Naples Federico II, Naples, Italy, in 2017, and the Ph.D. degree in metrology from the Politecnico di Torino, Turin, Italy, in 2022.

His research activities involve metrological characterization and machine-learning-based analysis.

Francesco Isgrò (Member, IEEE) received the M.S. degree in mathematics from the University of Palermo, Palermo, Italy, in 1994, and the Ph.D. degree in computer science from Heriot-Watt University, Edinburgh, U.K., in 2001.

He is an Associate Professor with the University of Naples Federico II, Naples, Italy.

Open Access funding provided by 'Università degli Studi di Napoli "Federico II"' within the CRUI CARE Agreement