Side-Channel Analysis of Integrate-and-Fire Neurons within Spiking Neural Networks

Matthias Probst*, Manuel Brosch*, Georg Sigl*†
*School of Computation, Information and Technology, Technical University of Munich, Munich, Germany
†Fraunhofer Institute for Applied and Integrated Security (AISEC), Munich, Germany
{matthias.probst, manuel.brosch, sigl}@tum.de

***Abstract*—Spiking neural networks gain attention due to low power properties and event-based operation, making them suitable for usage in resource constrained embedded devices. Such edge devices allow physical access opening the door for side-channel analysis. In this work, we reverse engineer the parameters of a feed-forward spiking neural network implementation with correlation power analysis. Localized measurements of electro-magnetic emanations enable our attack, despite inherent parallelism and the resulting algorithmic noise of the network. We provide a methodology to extract valuable parameters of integrate-and-fire neurons in all layers, as well as the layer sizes.**

***Index Terms*—spiking neural networks, side-channel analysis, integrate-and-fire neuron**

I. Introduction

Implementations of Neural Networks (NNs) on edge devices become increasingly popular. Compression techniques enable NNs on resource constraint devices [1], since they improve both performance and power efficiency.

Parallel to the optimization of NNs, a new type of NN inspired by the human brain is under development and testing: Spiking Neural Networks (SNNs). Human neurons interchange information via synapses in form of electric pulses, so-called spikes. This behavior is mimicked within SNNs and leads to an event based processing, where information is encoded as spikes. A schematic of a layer within such a SNN can be seen in fig. 1. SNNs are even more power efficient compared to classical NNs and, thus, ideal for edge devices.

The protection of Intellectual Property (IP) stored in the trained NN is crucial and possible information leaks about the internal parameters or structure of the NN should be avoided. Since classical NNs are more common today, they are already analyzed from a security point of view. Hardware attacks like Side-Channel Analysis (SCA) must be considered for edge devices, since an adversary can gain physical access to the device. SCA utilizes physical quantities, such as power consumption, Electro-Magnetic (EM) emanations or execution time, to extract secret information about the NN.

However, in the domain of SNNs there exists little work in regard of security analysis. Within this paper, we close this gap and employ Correlation Power Analysis (CPA) to reverse engineer synchronous feed-forward SNNs implementing Integrate-and-Fire (IF) neurons and, thereby, stealing the stored IP.

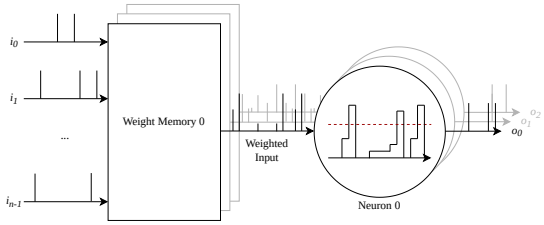


Fig. 1: Schematic of a simple SNN layer of three neurons reacting on n inputs.

A. Related Work

Recent publications focus on SCA to reveal information about NNs. Prominent works exploit the timing behavior [2] or power side-channel information [3], in order to reverse engineer the architecture of a NN. In addition to the network architecture, its parameters, such as weights, are of particular interest. Batina et al. utilize EM measurements to recover a complete NN from a microcontroller implementation [4]. Considering hardware implementations of NNs, Hua et al. extract a Convolutional Neural Network (CNN), executed on a hardware accelerator [5]. Dubey et al. reverse engineer the weights of a hardware implementation of a quantized NN by performing a Differential Power Analysis (DPA) [6]. The collection of side-channel attacks on NNs indicate the need for effective countermeasures such as shuffling [7] or masking [8], [9].

The related work we covered so far, focuses merely on implementations of classical NNs, but not SCA of SNNs. Similar attacks on implementations of SNNs do not exist to the best of our knowledge. Nevertheless, Yang et al. propose an algorithm to effectively reconstruct the neural network topology from the firing activity of a biological neural network [10]. This can be transferred to an attack, if an attacker has access to the communication between neurons. However, such an attack implies a very skillful and well-equipped attacker.

The closest to our work is a SCA by Garaffa et al. [11]. They recover the weights that preprocessing their associated input, before it is fed to the first hidden layer, on the basis of timing information, as well as information depicted in power traces.

B. Contribution and Organization

In this paper, we employ CPA to reverse engineer synchronous feed-forward SNNs implementing IF neurons. We purposely choose this particular implementation in order to introduce side-channel analysis of SNNs and its limitations. SNNs exhibit high parallelism and, thus, high algorithmic noise, which can be potentially counterproductive for an attacker. However, we show that this effect can be avoided by means of localized EM measurements, since individual neurons can be resolved. The acquired EM traces build the base of our CPA, with which we extract both - the weights and threshold parameters - of IF neurons.

As our results indicate, the attack is possible without any misclassification with about 2,000 EM-traces for each neuron. Hence, an attacker can extract a full SNN using IF neurons with our methods whilst maintaining the accuracy of the attacked network living the attacker with a white box model.

Our work is structured as follows: First we give an overview of the theoretical background of SNNs and CPA in section II. Section III explains the principle of our attack and the results are analyzed in section IV. Finally, we discuss the results and their implications in section V and conclude in section VI.

II. Background

A. Spiking Neural Networks

SNNs are inspired by the biological neural system and how information is transmitted and processed by the brain [12], [13]. Such a network can be seen as dynamical system, where individual neurons interact in an event-driven manner with each other [14].

The function of neurons within a SNN differs significantly from neurons in formal NNs. Similar to the classical neuron model, spiking neurons have weighted inputs to receive information from other preceding neurons or from some external instance. However, these inputs encode the information as so-called spikes and not as numerical values, as it is the case for formal NNs. As soon as a neuron recognizes a spike on one of its inputs, it accumulates the corresponding weight w to its internal membrane potential v. Depending on the used neuron model, the internal membrane potential v keeps its value until a new spike is received (IF), or it is reduced by some time constant as long as no spikes appear (Leaky Integrate-and-Fire (LIF)). At some point in time, when enough spikes have emerged, the membrane potential v of the neuron crosses a predefined threshold t. When exceeding the threshold the neuron emits a spike at its output o and resets v to the initial value. A representation of a layer can be found in fig. 1, where all neurons of this layer react on the inputs and generate as many outputs as neurons are in the layer. [12]

Due to their simple operation, i.e., multiplications and accumulations of formal NNs are reduced to accumulations only, and their event driven sparsity, SNNs are believed to reduce the energy consumption significantly [12], [15].

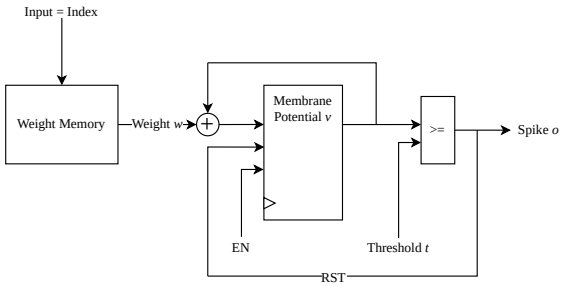


Fig. 2: Schematic of our IF neuron implementation.

B. Side-Channel Analysis

Side-Channel Analysis (SCA) was introduced in 1996 by Kocher [16]. It originates from the cryptographic sector and exploits information leakage about secrets through physical quantities, such as timing [16], power consumption [17] or EM [18]. Differential attacks, like DPA, employ statistical methods to utilize data dependencies within the power consumption for different input data to retrieve secret information. Pearson's correlation coefficient is a common statistical metric, in order to compare the collected traces with hypothetical values from a power model and distinguish the correct hypothesis from wrong ones. Such a correlation-based DPA is also referred to as CPA [19]. The advantage of CPA is, that no detailed knowledge about the implementation is required and it is robust against noise [20].

III. Attack Methodology

The goal of this work is to show the extraction of the valuable parameters of IF neurons implemented in SNNs putting the contained IP at risk. For the theoretical analysis, as well as the experiments, we focus on a synchronous hardware implementation on a Field Programmable Gate Array (FPGA).[1]

A. SNN Under Attack

Our realization of the IF neuron is depicted in fig. 2. Within a layer of a SNN, multiple of such neurons are implemented in parallel. Each neuron in one layer reacts on the same input sequence. Whenever an input arrives in form of a spike, the corresponding weight w is loaded from the weight memory and added to the current membrane potential v. The resulting sum is the new membrane potential v' a clock cycle later. The comparator in fig. 2 compares v against a threshold t. Whenever v exceeds t the neuron generates a spike at the output o and resets v to zero within the same clock cycle. While the neurons from the input layer react on the input provided to the SNN, subsequent layers react on the output of the previous layers neurons.

To interconnect inputs and neurons, we utilize a version of the Address Event Representation (AER) protocol [21]. Therefore, each input and neuron broadcasts its address over

[1] Nevertheless, our attack principle can be transferred to other platforms with minor modifications.

the bus. For example, whenever the neuron in fig. 2 generates a spike, its address is put on the AER bus. For each neuron, which is reacting on this event, the weight corresponding to this address is loaded from the weight memory and processed as described above.

Weights w in our case are 4-bit wide, whereas the membrane potential register has a width of 13-bit. Please note, that the small bitwidth of the weights is chosen in order to keep the computational effort for the CPA low. Nevertheless, our simulations prove, that the choice of bitwidth has no influence on our attack, apart from increasing the degree of computational effort. Furthermore, the attack is independent of the interconnection protocol, since we do not rely on any properties of it.

B. Threat Model

For our experiments we assume an attacker, who has physical access to the device executing the target SNN. Moreover, the attacker is passive, i.e., can observe the device during its normal operation, but cannot interfere the computations by inducing faults, for example. Since we use a decapped chip, we are in the semi-invasive scenario, where modifications to the hardware are allowed as long as they do not alter the functionality of the device. As a result, active attacks, like fault attacks or control flow hijacking, are out of scope here. Furthermore, no access to the memory or bus system is possible for the attacker, meaning that he/she cannot monitor memory accesses or retrieve data from memory. We assume, that the network parameters are securely loaded to the neurons. Thus, this poses no potential attack vector.

However, the attacker has full control over the input[2]. The attacker can observe side-channel information, as he/she has physical access to the targeted device, e.g., by measuring the power consumption or EM radiation. He/she is interested in reverse engineering the IP stored within the SNN, since training a network successfully is depending on a magnitude of design parameters and requires high computational effort. This can be of special interest, as well, when effective adversarial examples to a competitors model should be crafted [22], since a white box model is necessary there.

C. Weight Retrieval

In order to extract the weights of a neuron w, a point, where they are processed together with known information is necessary. A suitable target is the addition of the weights w to the membrane potential v of a neuron in fig. 2, since an attacker can influence v by applying spikes to the inputs.

Specifically, an attacker chooses the index i of the secret weight w_i that will be added to the membrane potential v of a neuron, by applying a spike to the corresponding input. Please note, the index i is equivalent to the input i, where a spike is applied to. In general, the idea of the attack is to apply different combinations of spike sequences on the inputs, to achieve high variance in the membrane potential v.

[2]If an attacker can only observe, but not control the input, he/she can also apply the attack by only utilizing input patterns suiting the attack requirements.

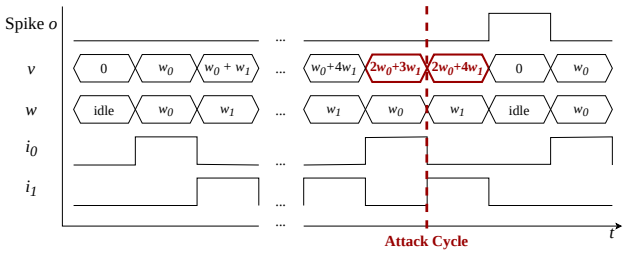


Fig. 3: Attack point for weight retrieval.

For our attack, we apply different sequences to inputs i_0 and i_1 and guess the associated weight values w_0 and w_1, as shown in fig. 3. The change in membrane potential v at a targeted point in time is modeled for all possible weights and all of those guesses are stored in the hypothesis matrix h_d. Those hypothetical intermediate results for v are then transformed into theoretical power consumption values by means of a power model. In order to model the power consumption of the value transistion, we are using the Hamming Distance (HD) power model, where the amount of changing bits between two values defines the power consumption. If the power model matches the implementation, the correct hypothesis of the weights should achieve a significantly higher correlation as all other hypotheses. The correlation is calculated utilizing Pearson's correlation coefficient, where the maximum value of

$$\rho_{d,c} = \frac{\sum_{m=0}^{N-1}((h_{m,d} - \overline{h}_d) \cdot (t_{m,c} - \overline{t}_c))}{\sqrt{\sum_{m=0}^{N-1}(h_{m,d} - \overline{h}_d)^2 \cdot \sum_{m=0}^{N-1}(t_{m,c} - \overline{t}_c)^2}}, \quad (1)$$

will most likely belong to the correct hypothesis. Thereby, N corresponds to the number of measurements, $h_{m,d}$ represents a theoretical power value for measurement m and the hypothesis d, and $t_{m,c}$ is the m-th power trace at sample point c.

However, there are a few restrictions for our attack. First, the number of different input indices must be limited, i.e., we cannot attack all weights of a neuron in one step. If we do not limit the amount of inputs and weights involved in the hypotheses d, our hypothesis space would be no longer manageable due to memory size and computation time. For our targeted implementation, we have 4-bit weights and if we apply spikes on n different inputs, we have $2^{n \cdot 4}$ different hypotheses. The choice of $n = 2$ results in a manageable hypothesis space of 2^8, while at the same time the correct weights can be identified accurately in our experiments. Choosing n is a trade-off between being able to retrieve more weights at once more robustly and having a large hypothesis space which is computational challenging.

Additionally, we have to assume that the neuron does not emit a spike during our attack, i.e., v crosses the threshold t and the neuron emits a spike at the output after the targeted attack cycle, as shown in fig. 3. That implies, however, that a late attack cycle increases the chance that the neuron itself emits a spike and resets its membrane potential v. Such a behavior cannot easily be implemented in our hypothesis, since

we require knowledge about the threshold in addition, which we can only extract after knowing the weights.

However, a late point in time also means a higher possible variance of the membrane potential of a neuron, which is beneficial to distinguish the correct hypothesis from incorrect ones. Therefore, the targeted point in time on that the hypotheses are build on, fundamentally contributes to the distinctness of single hypotheses. The choice of attack cycle again is a trade-off: A later cycle makes the attack more robust, since there is a higher variance in v that has a positive impact to the correlation coefficient in eq. (1).

We choose to attack at 20 clock cycles after the reset of v, since this produces sufficient variance and keeps the computational overhead low. Nevertheless, the attack works as well with even less cycles. Furthermore, the neurons in our network never reach their thresholds at this point.

D. Threshold Retrieval

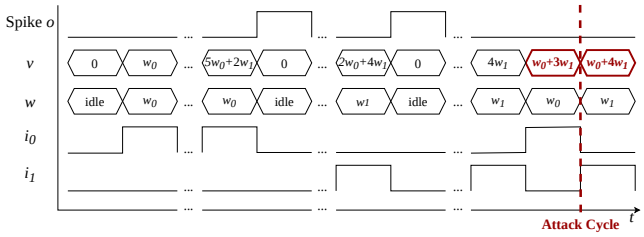


Fig. 4: Attack point for threshold retrieval.

Next, we can extract the neuron's threshold potential as follows. An attacker applies input combinations to the neuron, where for each combination he/she knows the corresponding weights added to the potential v. In fig. 4 w_0 and w_1 are known to the attacker. As a result, he/she can again build hypotheses on the corresponding intermediate values of v, however now, by taking a guessed threshold into account. The point of interest in this case is later than before in the weight attack scenario. In contrast to the weight attack, an attacker does not want to avoid exceeding the threshold, but requires the neuron to spike at least once. This is indicated in fig. 4, where the depicted neuron emits two spikes before our chosen attack cycle. Therefore, a possible attack point is after many inputs, when the neuron has emitted one or more spikes.

As before, the correlation between the hypotheses and the measured traces is calculated according to eq. (1). Again the highest correlating value most likely belongs to the correct threshold.

Please note that the threshold can only be determined as granular as the smallest difference of two weights occurring in the particular neuron. For example, a neuron featuring two weights with value 20 and 30, the threshold can only be determined to a precision of ten. If in the example 45 would be the true threshold value, the attack can only determine that it is between 40 and 49. Nevertheless, this limitation does not change the behavior of the reverse engineered neuron.

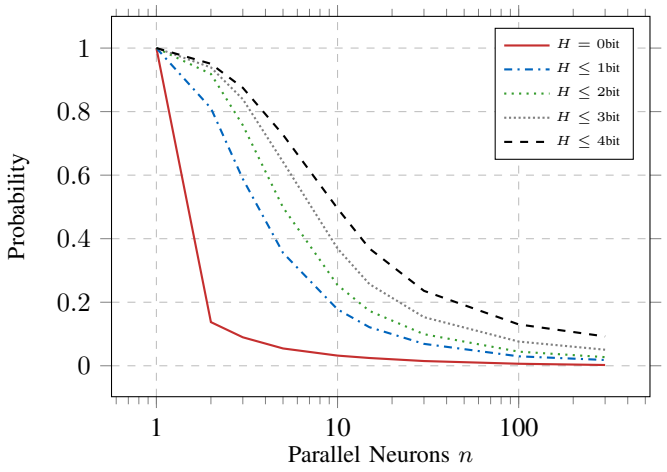


Fig. 5: Probability of a guessing entropy H after attacking a single neuron with the presence of n parallel neurons.

E. Extraction Limitations

SNNs are implemented in a highly parallel manner and, thus, offer a large degree of algorithmic noise. This can influence the performance of our attack.

We chose simulations as best case scenario for an attacker, where no measurement noise is present. Within our simulations, we consider, that all n parallel operating neurons update their membrane potential at the same point in time. For the attacker we assume, that he/she can only measure the overall power consumption during the operation of the device, which gives us the scenario that he/she acquires power measurements during the operation of the device. To transfer processed values to a virtual power consumption we use the HD model, which also fits the afterwards attacked FPGA implementation.

We performed 100,000 experiments for each number of parallel neurons. We attack at clock cycle 20, which gives us 420 distinct input combinations consisting of two inputs and idle cycles.[3] For each run the weights to be attacked, as well as the weights of all other neurons are chosen randomly. During each simulation, we try to attack a single neuron. The simulation results are depicted in fig. 5, where the probability of a guessing entropy H after applying our attack are plotted over an increasing amount n of parallel neurons. The probabilities are calculated by counting the occurrences of attacks with the depicted guessing entropy H and dividing them by the number of all experiments. The simulated neurons utilize 4-bit weights, and we employ hypotheses of two weights resulting in an initial guessing entropy of 8-bit. These are the same parameters, we use in our later attacked FPGA implementation. Please note, that the depicted trend does not change with increasing width of weights, i.e., experiments with using 8-bit weights result in similar behavior.

[3]The number of input combinations is equal to the maximum amount of different value transitions from clock cycle 19 to 20, which are possible when we apply spikes to two inputs.

Results in fig. 5 indicate, that an increase in the number of parallel neurons decreases the success of an attack. For example, a guessing entropy of 2-bit or less is only reached in about 4.4 % of the cases, when 100 neurons are implemented in parallel after performing the CPA. Consequently, extracting the neuron parameters by observing the overall power consumption is not possible. In fig. 5 one can identify, that an attacker must reduce or eliminate the influence of other neuron(s) while taking measurements. Thus, localized measurements are necessary to extract the valuable information.

F. Layer Size and Subsequent Layers

In addition to the neuron parameters, the network topology is of interest to an attacker. The layer size of the targeted feed-forward network can be identified, since an attacker needs to attack each neuron reacting to the input spikes he/she is applying. Therefore, he/she needs to take measurements of each neuron or a few neurons separately, due to the effects described in section III-E. Hence, an attacker can identify, how many neurons react to the applied inputs directly and assign them to the first layer. A reaction to the input pattern is noticed by performing an attack according section III-C and analyzing the magnitude of correlation coefficients. In other words: an attack on the weights of a neuron with hypotheses h_d originating from a wrong assumption of the input will not yield any distinct correlation coefficients.

After attacking the first layer, an attacker must also recover subsequent layers to extract the full IP of the implemented network. According to our methods from section III-C and section III-D, the attacker completely reverse engineers the first layer. Knowing the first layer, enables the attacker to model its behavior. Hence, known input combinations can be applied to the next hidden layer. As a result, an attacker can successively reverse engineer the whole network layer by layer.

IV. Experimental Results

In the following the experimental results of our CPA from Section III are analyzed. We utilize the NewAE Technology CW305 FPGA board featuring a Xilinx Artix-7 FPGA, which is decapped to increase the Signal-to-Noise Ratio (SNR) of EM radiation. The device runs at a frequency of $1\,\mathrm{MHz}$ to simultaneously obtain exploitable power[4] and EM traces. We sample with $625\,\mathrm{MHz}$ using a Picoscope 6402D and a near field EM probe (Langer ICR HH250-75). Both, power and EM signals are amplified with a Langer PA 303 by $30\,\mathrm{dB}$. Our measurement setup is depicted in fig. 6.

We implement one layer of a SNN with 100 neurons operating in parallel, where the 4-bit weights are chosen randomly[5], and the threshold is set to 1,000 so we do not reach it. We acquire EM and power measurements at the same time,

[4]Due to capacitive effects a lower clock frequency is necessary for power measurements. However, EM traces can be attacked by higher frequencies, as well.

[5]The attack is independent of the chosen weight and threshold parameters, i.e. any weight and any threshold can be reverse engineered the same way.

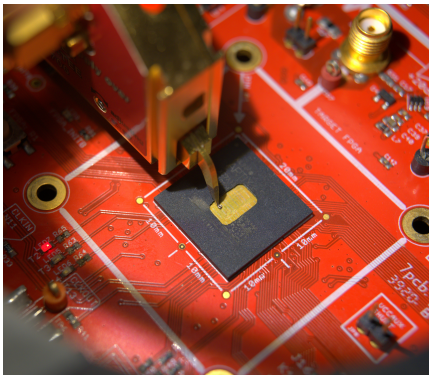


Fig. 6: EM near field probe over the decapped FPGA.

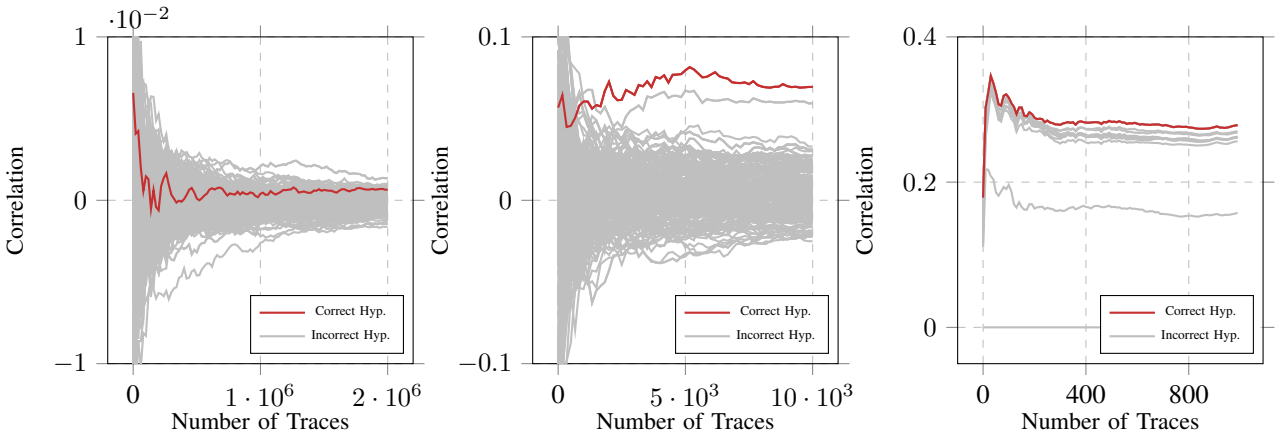
meaning that the results of fig. 7a and fig. 7b are from the same measurement run. The EM measurements from fig. 7b depict a single neuron operating. The plots are originating from measurements, where random patterns stimulating two inputs are applied to the network under attack. The measurements used to extract the threshold, shown in fig. 7c, are acquired separately, since we reduced the threshold there to ensure that it is exceeded several times.

From fig. 7a we can see that an attack is not possible when the power consumption of 100 parallel neurons is observed, even with 2 million traces. This goes in hand with our findings from section III-E. However, we also stated, that one to a few neurons are attackable. This is confirmed by the results of the localized EM measurement in fig. 7b, where the correct weights can be identified with about 2,000 traces. Please note, due to our probe, we succeeded in resolving a single neuron. The correct threshold can also be distinguished from wrong candidates with about 200 traces, which is shown in fig. 7c. The significantly lower number of traces for attacking the threshold compared to the weights can be explained due to two effects. On the one hand, the granularity of weights only allows a few threshold values. On the other hand, a small change in threshold influences the membrane potential drastically at a late attack cycle.

Overall, the theoretical simulations from section III-E are confirmed by our experimental results. Localized measurements lead to a successful retrieval of weights and thresholds. Thus, an attacker is able to reverse engineer all neurons by localizing them and applying our attack methodology from section III. Please note, that results of neurons in further layers are not depicted here. Although the attack strategy according to section III-F is different from first layer neurons, the resulting plots do not differ.

V. Discussion

Section IV indicates, that reverse engineering SNNs is successful as long as an attacker is able to avoid the influence of parallel neurons in his/her measurements. Compared to [4], where a sequential implementation of a formal NN is attacked, the algorithmic noise increases due to the parallelism in our implementation. Thus, several attack coordinates on the chip,



(a) Weight SCA on power traces (b) Weight SCA on local. EM traces (c) Result for Threshold attack

Fig. 7: CPA result for weights retrieval based on power (a) and EM measurements (b). The results for the threshold (c) are from separate EM traces.

i.e. positions of the probe, must be measured separately. This means, that the effort of our SCA scales directly with the number of positions, and thus, with the size of the implemented SNN. However, parallel hardware implementations of formal NNs also suffer from the problem of algorithmic noise influencing the attack success. Attacks such as [6], however, do not go into detail of this effect. This can be due to the fact, it is less significant in classical NNs compared to SNNs. Further investigation can be necessary to give more insights in algorithmic noise due to parallelism, in the context of NN hardware implementations.

Nevertheless, we performed a successful attack on integrate-and-fire neurons within a SNN. Our findings highlight the need for countermeasures against SCA of SNNs such as masking or shuffling.

VI. Conclusion

Due to their low-power properties, as well as their event based operation, spiking neural networks are suitable for edge devices. However, edge devices enable side-channel analysis with aim to reverse engineer the network. In this work, we outlined the threat of CPA to reverse engineer the parameters of feed-forward SNNs utilizing IF neurons. High parallelism and the resulting algorithmic noise within SNNs hinders the extraction of information from the overall power consumption of the chip. We circumvented this effect by utilizing localized EM measurements. Furthermore, this behavior is additionally confirmed by analyzing virtual results under ideal conditions by means of simulation.

References

[1] Y. Chen, Y. Xie, L. Song, F. Chen, and T. Tang, "A survey of accelerator architectures for deep neural networks," *Engineering*, 2020.

[2] V. Duddu, D. Samanta, D. V. Rao, and V. E. Balas, "Stealing neural networks via timing side channels," *arXiv preprint arXiv:1812.11720*, 2018.

[3] Y. Xiang, Z. Chen, Z. Chen, Z. Fang, H. Hao, J. Chen, Y. Liu, Z. Wu, Q. Xuan, and X. Yang, "Open dnn box by power side-channel attack," *IEEE Transactions on Circuits and Systems II: Express Briefs*, 2020.

[4] L. Batina, S. Bhasin, D. Jap, and S. Picek, "{CSI}{NN}: Reverse engineering of neural network architectures through electromagnetic side channel," in {*USENIX*} *Security Symposium* ({*USENIX*} *Security)*, pp. 515–532, 2019.

[5] W. Hua, Z. Zhang, and G. E. Suh, "Reverse engineering convolutional neural networks through side-channel information leaks," in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, pp. 1–6, IEEE, 2018.

[6] A. Dubey, R. Cammarota, and A. Aysu, "Maskednet: A pathway for secure inference against power side-channel attacks," *arXiv preprint arXiv:1910.13063*, 2019.

[7] M. Brosch, M. Probst, and G. Sigl, "Counteract side-channel analysis of neural networks by shuffling," in *2022 Design, Automation and Test in Europe Conference and Exhibition (DATE)*, IEEE, Mar 2022.

[8] K. Athanasiou, T. Wahl, A. A. Ding, and Y. Fei, "Masking feedforward neural networks against power analysis attacks," *Proceedings on Privacy Enhancing Technologies*, vol. 2022, no. 1, pp. 501–521, 2022.

[9] A. Dubey, A. Ahmad, M. A. Pasha, R. Cammarota, and A. Aysu, "Modulonet: Neural networks meet modular arithmetic for efficient hardware masking," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 506–556, 2022.

[10] X. Yang, J. Wang, and C. Liu, "Reconstructing neural network topology from firing activity," in *2020 39th Chinese Control Conference (CCC)*, pp. 7106–7111, IEEE, 2020.

[11] L. C. Garaffa, A. Aljuffri, C. Reinbrecht, S. Hamdioui, M. Taouil, and J. Sepulveda, "Revealing the secrets of spiking neural networks: The case of izhikevich neuron," in *2021 24th Euromicro Conference on Digital System Design (DSD)*, pp. 514–518, IEEE, 2021.

[12] M. Bouvier, A. Valentian, T. Mesquida, F. Rummens, M. Reyboz, E. Vianello, and E. Beigne, "Spiking neural networks hardware implementations and challenges: A survey," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 15, no. 2, pp. 1–35, 2019.

[13] A. Rosado-Muñoz, M. Bataller-Mompeán, and J. Guerrero-Martínez, "Fpga implementation of spiking neural networks," *IFAC Proceedings Volumes*, vol. 45, no. 4, pp. 139–144, 2012.

[14] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018.

[15] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, B. Brezzo, I. Vo, S. K. Esser, R. Appuswamy, B. Taba, A. Amir, M. D. Flickner, W. P. Risk, R. Manohar, and D. S. Modha, "A million spiking-neuron

integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, 2014.

[16] P. C. Kocher, "Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems," in *Advances in Cryptology — CRYPTO '96* (N. Koblitz, ed.), (Berlin, Heidelberg), pp. 104–113, Springer Berlin Heidelberg, 1996.

[17] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology — CRYPTO' 99*, pp. 388–397, Springer Berlin Heidelberg, 1999.

[18] D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi, "The em side—channel (s)," in *International workshop on cryptographic hardware and embedded systems*, pp. 29–45, Springer, 2002.

[19] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *International workshop on cryptographic hardware and embedded systems*, pp. 16–29, Springer, 2004.

[20] S. Mangard, E. Oswald, and T. Popp, *Power analysis attacks: Revealing the secrets of smart cards*, vol. 31. Springer Science & Business Media, 2008.

[21] C. S. Thakur, J. L. Molin, G. Cauwenberghs, G. Indiveri, K. Kumar, N. Qiao, J. Schemmel, R. Wang, E. Chicca, J. O. Hasler, J. sun Seo, S. Yu, Y. Cao, A. van Schaik, and R. Etienne-Cummings, "Large-scale neuromorphic spiking array processors: A quest to mimic the brain," *Frontiers in Neuroscience*, vol. 12, 2018.

[22] A. Bagheri, O. Simeone, and B. Rajendran, "Adversarial training for probabilistic spiking neural networks," in *2018 IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pp. 1–5, IEEE, 2018.