# Give Me 5 Minutes

## Attacking ASCAD with a Single Side-Channel Trace

Olivier Bronchain, Gaëtan Cassiers and François-Xavier Standaert

ICTEAM Institute, UCLouvain, Louvain-la-Neuve, Belgium.
{olivier.bronchain,gaetan.cassiers,fstandae}@uclouvain.be

**Abstract.** In this paper, we describe a profiled attack against the ANSSI Side-Channel Analysis Database (ASCAD), which recovers the full key using the leakage of a single masked AES execution. The attack uses a new open-source Side-Channel Analysis Library (SCALib), which allows running the leakage profiling and attacking in less than 5 minutes. It exploits well-known techniques, yet improves significantly over the best known attacks against ASCAD. We conclude by questioning the impact of these experimental findings for side-channel security evaluations.

**Keywords:** Profiled Side-Channel Analysis · Soft Analytical Side-Channel Attacks

## 1 Introduction

Since 2018, ASCAD has become one of the most used datasets for side-channel analysis benchmarking [BPS+20]. It contains power measurements of a two-share masked AES implementation running on an 8-bit micro-controller. As intended by the authors, most of the published attacks against ASCAD use Deep Learning (DL). We refer to the publications using this dataset at CHES 2020 for illustration [ZBHV20, MDP20, WAGP20, WP20, PCP20, HHO20, ZZN+20, ZDF20]. Most of these works focus on recovering a single key byte by looking at a small (possibly desynchronized) part of the leakage traces, and do not exploit the knowledge of the masking randomness during their profiling stage. An exception is the recent work of Xiangjun Lu et al. which we discuss in conclusion [LZC+21].

In this paper, we take another approach and use the recently introduced SCALib[1] to analyze ASCAD. The proposed attack efficiently takes advantage of all the time samples in the leakage traces (next called raw traces) and leverages the knowledge of the masking randomness during the profiling phase. It mixes Signal to Noise Ratio (SNR) computations, Linear Discriminant Analysis (LDA) [SA08], pooled Gaussian templates [CK13] and Soft Analytical Side-Channel Attacks (SASCA) [VGS14]. As a result, we are able to efficiently recover a full encryption key by exploiting the leakage of a single masked block cipher execution.[2]

Our attack methodology is not novel: it is based on well-known algorithmic tools and resembles closely methodologies used in previous works [BS20, BS21]. However, it results in a very efficient and straightforward attack which outperforms the state-of-the-art attacks against this dataset and therefore raises interesting questions for side-channel security evaluations.

[1] https://github.com/simple-crypto/SCALib

[2] The full key is 14 bytes since the 2 first bytes of the AES key are not masked in ASCAD.

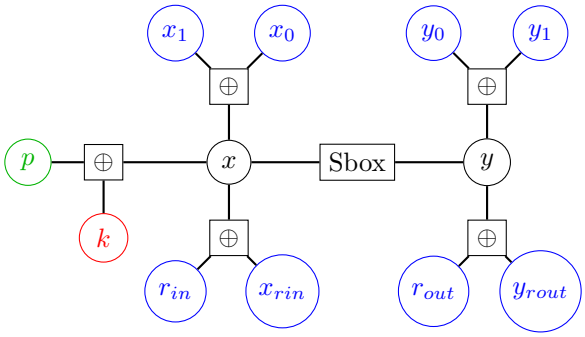



Figure 1: Single byte graph.

## 2 ASCAD Dataset

The ASCAD dataset [BPS+20] is a set of power consumption traces of an ATMega8515 microcontroller (MCU) computing the AES with a first-order masked implementation. In this paper, we use the *raw traces* of the *variable key* dataset[3] with no pre-processing. Each trace contains 250 000 8-bit power consumption measurements, and is labeled with the following metadata: the key, the plaintext and all the masks used in the masking scheme.

We next describe (and illustrate in Figure 1) the part of the AES implementation that is relevant for our attack, namely the beginning of the first round. The implementation randomly generates the masks $x_1^i$, $y_1^i$ (for $i = 1, \ldots, 16$), $r_{in}$ and $r_{out}$ (a mask is an independent and uniform random byte). For each key byte $k^i$ and corresponding plaintext byte $p^i$, the sharing $(x_0^i, x_1^i)$ such that $x_0^i \oplus x_1^i = x^i$ is computed, where $x^i = p^i \oplus k^i$. Then, this sharing is converted to another sharing $(x_{rin}^i, r_{in})$ such that $x_{rin}^i \oplus r_{in} = x^i$. Using lookup tables, the implementation computes the sharing $(y_{out}^i, r_{out})$ such that $y_{rout}^i \oplus r_{out} = y^i$, where $y^i = \text{Sbox}(x^i)$. Finally, this sharing is converted to a last sharing $(y_0^i, y_1^i)$ such that $y_0^i \oplus y_1^i = y^i$.

## 3 Attack methodology

The methodology used to mount the attack is similar to the one proposed by Bronchain and Standaert in [BS20, BS21].

**Profiling phase.** The profiling exploits the knowledge of the value of all the sensitive variables (i.e., the shares) manipulated by the implementation (thanks to the knowledge of the key, the plaintext and the masks). It works in two steps:

1. The **SNR** is computed for all the shares within the implementation [Man04]. The samples with largest SNR are taken as $p$ Points-Of-Interest (POIs) for each share, out of the 250 000 traces' samples.

2. **Models** are then built for each share, which gives us an estimate of the probability distribution of each share conditioned on the leakage trace. Each model is built by first applying a LDA dimensionality reduction from the $p$-dimensional space (POIs) to a $d$-dimensional feature space. Next, we build pooled Gaussian templates for the reduced-dimensionality features.

[3] More precisely, we use the traces from the database located at https://static.data.gouv.fr/resources/ascad-atmega-8515-variable-key/20190730-071646/atmega8515-raw-traces.h5.

**Attack phase.** The attack phase uses SASCA and exploits only the trace and the knowledge of the plaintext. For each key byte, it goes in three sequential steps:

1. **Graph description** consists in representing the implementation with a factor graph including operations and variables. The graph used in this paper is represented in Figure 1 and follows the guidelines of [BS21]. More precisely, each shared intermediate variables is unmasked, and the relationships between the unshared variables are encoded. As an example, the Sbox output $y$ is derived from its shares $y_i$ such that $y = y_0 \oplus y_1$. Then the circuit is represented with operations on the unshared variables. In Figure 1, the shares are in blue, the plaintext is in green and the key byte to recover is in red. A rectangle stands for an operation and a circle for a variable. If multiple traces are used for the attack, the graph is replicated once for each trace, with the $k$ node being common across all replicas. This construction is effective at recovering information on the key, while also being computationally lightweight (the factor graph is relatively small) and avoiding cycles (which may cause convergence issues [GRO18]).
2. **Probabilities extraction** consists in using the models to obtain the distribution $P_x = (\Pr[x = \alpha|\mathbf{l}])_{\alpha \in \mathbb{F}_{256}}$ of the shares $x$ given the leakage $\mathbf{l}$. Then, this probability is inserted in the factor graph as the initial distribution of these shares.
3. **Belief propagation** is a message passing rule between the nodes of the factor graph [VGS14]. It allows us to estimate the probability of a key byte (i.e., $k$ in Figure 1) based on the initial distribution of the shares. For our simple factor graph, the belief propagation algorithm reduces to the following computation when there is only on attack trace:

$$P_k = P_p \otimes \left[ (P_{x_0} \otimes P_{x_1}) * (P_{r_{in}} \otimes P_{x_{rin}}) \right. \\ \left. * \text{Sbox}^{-1} \left( (P_{y_0} \otimes P_{y_1}) * (P_{r_{out}} \otimes P_{y_{rout}}) \right) \right] , \quad (1)$$

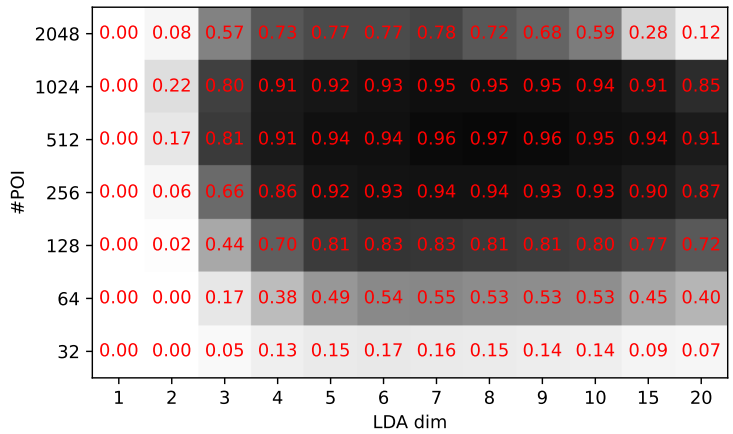
where $P_\alpha$ denotes the distribution of a variable $\alpha$ (extracted from leakage or prior knowledge), $*$ represents a multiplication (followed by a normalization) of the distribution's probabilities, while $\otimes$ represents to the convolution of the histograms (which corresponds to the $\oplus$ operation on the random variables). When there are multiple traces, Equation 1 is computed independently for each trace, and the resulting $P_k$ values are combined together using the $*$ operation.

**Evaluation.** To evaluate the attack, we compute the rank of the correct key based on the probabilities retrieved by the attack for each key byte. SCALib provides an implementation of the histogram-based rank estimation from CHES 2016 [PSG16]. We estimate the success rate (i.e., the probability that the full key is ranked first) by running the attack on different sets of attack traces sets.

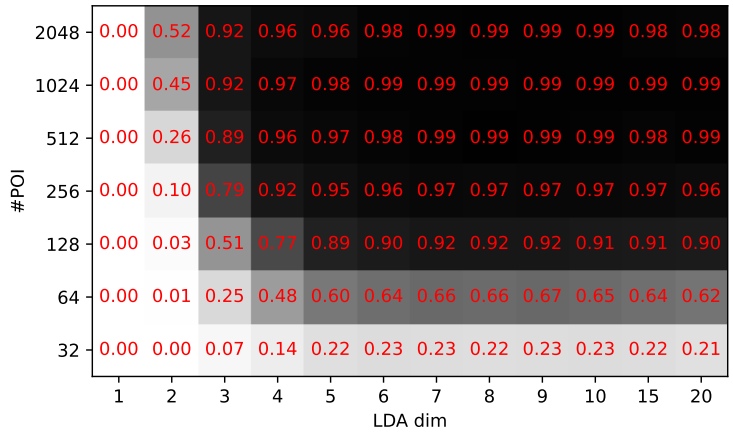
**Implementation.** The implementation of the attack and its evaluation are based on the SCALib library. The attack and evaluation scripts are both available at `https://github.com/cassiersg/ASCAD-5minutes`. Thanks to the simplicity of the algorithms involved and to the efficient implementations in SCALib, the attack runs very quickly: at most in a few minutes for the full profiling phase, and milliseconds for the online attack. Performance measurements are reported in Appendix B.

## 4 Results

We ran our attack on the full raw traces (i.e., 250.000 time samples) with a single attack trace (i.e., an attack set of size 1) and estimated the success rate for the 14 byte masked




(a) $N = 5000$ profiling traces.




(b) $N = 20\,000$ profiling traces.

Figure 2: Attacks against the full key (14 bytes) using the full ASCAD traces: success rate for attacks with a single trace (i.e., probability that the rank of the correct key is 1). The axes are the number of POIs selected and the number of dimensions kept after the LDA. Attacks are repeated 1000 times to estimate the success rate.

key. The main result is the following: **with a single attack trace, we achieve a 99 % the success rate (i.e., correct key at rank 1)**.

### 4.1 Meta-parameters exploration

Beyond the number of attack traces, the main parameters of our methodology are related to the profiling phase, namely:

- $N$: the number of profiling traces (used for SNR, LDA and Gaussian templates computations),
- $d$: the number of dimensions at the output of the LDA,
- $p$: the number of POIs.

For completeness, we investigate the impact of these three parameters on the success rate (with a single attack trace). Regarding the number of profiling traces, we considered the cases $N = 5000$ and $N = 20\,000$.[4] The number of LDA dimensions ranges from 1 to 20 (at which point the success rate saturates), and the number of POIs ranges from 32 (as it must be above the number of LDA dimensions) to 2048 (where success rate saturates).

The results are shown in Figure 2. When $N = 20\,000$, we first remark that our attack works for most sets of meta-parameters considered: it succeeds with significant likelihood as long as $d \geq 3$. Next, the success rate is above 95 % for a large set of meta-parameter values: if $d$ and $p$ are above 5 and 256 respectively. For the smaller $N = 5000$, the success rate drops when $d$ and $p$ grow above some threshold. This is likely due to the lack of stability of the LDA and Gaussian templates when the number of parameters to fit becomes too large compared to the number of profiling traces.

**Short-trace attack.** For the sake of completeness, we also report in Appendix A an attack on the third byte exploiting only a few samples (1400) of the traces, as considered in many previous works.

## 5 Conclusion: so what?

The full key recovery attack described in this paper significantly improves the state-of-the-art, both in terms of profiling cost and in terms of online attack complexity. At high level, efficient profiling is obtained thanks to a good exploitation of the masking randomness while the improved online attack is primarily due to an analytical strategy that takes advantage of the complete (raw) traces. These results naturally raise the question of what is their impact for security evaluations. That is, should one consider that the security level of the ASCAD implementation corresponds to worst-case attacks with strong capabilities or to more relaxed attacks (e.g., without masking randomness during profiling)?

Our main conclusions in this respect are twofold. On the one hand, worst-case evaluations are significantly faster to perform and are also a useful tool to anticipate the risk of improved attacks due to continuous research developments. So following the

[4] We took $N = 5000$ to ensure that our attack can be run without error. Indeed, the LDA and pooled Gaussian templates can only be computed if each of the 86 profiled intermediate variables takes each of the 256 possible values at least once in the profiling dataset. This occurs with probability

$$\left(1 - \sum_{k=1}^{256} (-1)^{k-1} \binom{256}{k} \left(1 - \frac{k}{256}\right)^N\right)^{86},$$

which is more than 99.9 % for $N = 5000$, but is below this threshold for $N = 4000$.

backwards evaluation approach outlined in [ABB+20], we believe they are anyway an interesting asset before launching more expensive attacks in less permissive contexts.[5] On the other hand, the gap between worst-case attacks and relaxed ones remains to be thoroughly investigated. Understanding whether it affects the profiling complexity, the online attack complexity or both (and to what extent) would help to better clarify the conservativeness/practicality of these two evaluation settings. Without such a clearly established and quantified gap, considering worst-case attacks as unpractical appears as an undesirable risk of security overstatement, especially in the long term.

We finally remark that the recent work in [LZC+21] provides an interesting counterpart to our results and does a first step in quantifying the gap between worst-case and relaxed adversaries. It shows attacks against a single byte of the ASCAD implementation that succeed with 3 to 10 raw traces. Their profiling does not leverage the known masking randomness of the ASCAD traces and is therefore significantly more expensive, confirming our worst-case approach as a useful shortcut for evaluators in this case.

## References

[ABB+20] Melissa Azouaoui, Davide Bellizia, Ileana Buhan, Nicolas Debande, Sébastien Duval, Christophe Giraud, Éliane Jaulmes, François Koeune, Elisabeth Oswald, François-Xavier Standaert, and Carolyn Whitnall. A systematic appraisal of side channel evaluation strategies. In *SSR*, volume 12529 of *Lecture Notes in Computer Science*, pages 46–66. Springer, 2020.

[BPS+20] Ryad Benadjila, Emmanuel Prouff, Rémi Strullu, Eleonora Cagli, and Cécile Dumas. Deep learning for side-channel analysis and introduction to ASCAD database. *J. Cryptogr. Eng.*, 10(2):163–188, 2020.

[BS20] Olivier Bronchain and François-Xavier Standaert. Side-channel countermeasures' dissection and the limits of closed source security evaluations. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(2):1–25, 2020.

[BS21] Olivier Bronchain and François-Xavier Standaert. Breaking masked implementations with many shares on 32-bit software platforms or when the security order does not matter. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(3):202–234, 2021.

[CK13] Omar Choudary and Markus G. Kuhn. Efficient template attacks. In *CARDIS*, volume 8419 of *Lecture Notes in Computer Science*, pages 253–270. Springer, 2013.

[GRO18] Joey Green, Arnab Roy, and Elisabeth Oswald. A systematic study of the impact of graphical models on inference-based attacks on AES. In *CARDIS*, volume 11389 of *Lecture Notes in Computer Science*, pages 18–34. Springer, 2018.

[HHO20] Anh-Tuan Hoang, Neil Hanley, and Máire 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. Cryptogr. Hardw. Embed. Syst.*, 2020(4):49–85, 2020.

[5] Note that this conclusion is not specific to the choice of statistical tools we made in our experiments. For example, deep learning can be used in a worst-case context too and could possibly lead to even more powerful attacks at the cost of a more expensive profiling. The winning (divide-and-conquer) attack of the CHES 2020 CTF is an example in this direction: https://ctf.spook.dev/submissions/. Its combination with an analytical strategy is an interesting scope for further investigations.

[LZC+21] Xiangjun Lu, Chi Zhang, Pei Cao, Dawu Gu, and Haining Lu. Pay attention to raw traces: A deep learning architecture for end-to-end profiling attacks. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(3):235–274, 2021.

[Man04] Stefan Mangard. Hardware countermeasures against DPA ? A statistical analysis of their effectiveness. In *CT-RSA*, volume 2964 of *Lecture Notes in Computer Science*, pages 222–235. Springer, 2004.

[MDP20] Loïc Masure, Cécile Dumas, and Emmanuel Prouff. A comprehensive study of deep learning for side-channel analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(1):348–375, 2020.

[PCP20] Guilherme Perin, Lukasz Chmielewski, and Stjepan Picek. Strength in numbers: Improving generalization with ensembles in machine learning-based profiled side-channel analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(4):337–364, 2020.

[PSG16] Romain Poussier, François-Xavier Standaert, and Vincent Grosso. Simple key enumeration (and rank estimation) using histograms: An integrated approach. In *CHES*, volume 9813 of *Lecture Notes in Computer Science*, pages 61–81. Springer, 2016.

[SA08] François-Xavier Standaert and Cédric Archambeau. Using subspace-based template attacks to compare and combine power and electromagnetic information leakages. In *CHES*, volume 5154 of *Lecture Notes in Computer Science*, pages 411–425. Springer, 2008.

[VGS14] Nicolas Veyrat-Charvillon, Benoît Gérard, and François-Xavier Standaert. Soft analytical side-channel attacks. In *ASIACRYPT (1)*, volume 8873 of *Lecture Notes in Computer Science*, pages 282–296. Springer, 2014.

[WAGP20] Lennert Wouters, Victor Arribas, Benedikt Gierlichs, and Bart Preneel. Revisiting a methodology for efficient CNN architectures in profiling attacks. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(3):147–168, 2020.

[WP20] Lichao Wu and Stjepan Picek. Remove some noise: On pre-processing of side-channel measurements with autoencoders. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(4):389–415, 2020.

[ZBHV20] Gabriel Zaid, Lilian Bossuet, Amaury Habrard, and Alexandre Venelli. Methodology for efficient CNN architectures in profiling attacks. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(1):1–36, 2020.

[ZDF20] Ziyue Zhang, A. Adam Ding, and Yunsi Fei. A fast and accurate guessing entropy estimation algorithm for full-key recovery. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(2):26–48, 2020.

[ZZN+20] Jiajia Zhang, Mengce Zheng, Jiehui Nan, Honggang Hu, and Nenghai Yu. A novel evaluation metric for deep learning-based side channel analysis and its extended application to imbalanced data. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(3):73–96, 2020.

# A Attack with partial traces

Figure 3 shows the result of an attack on the third byte exploiting only a few samples (1400) of the traces, as considered in many previous works. We run the attack 100 times

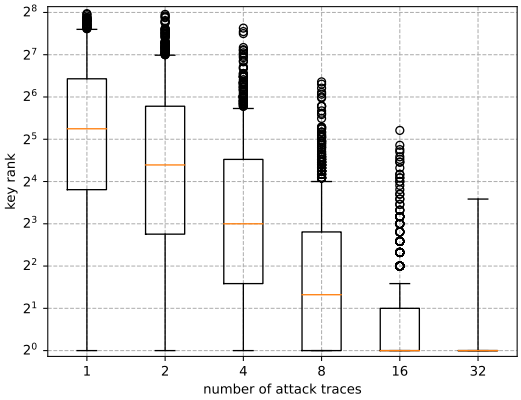



Figure 3: Attack against partial ASCAD traces for one key byte: boxplot of the rank of the correct key. Attacks with 32 traces almost always result in finding the correct key at rank 1. With only 8 traces, the correct key is at rank 1 more than half of the times.

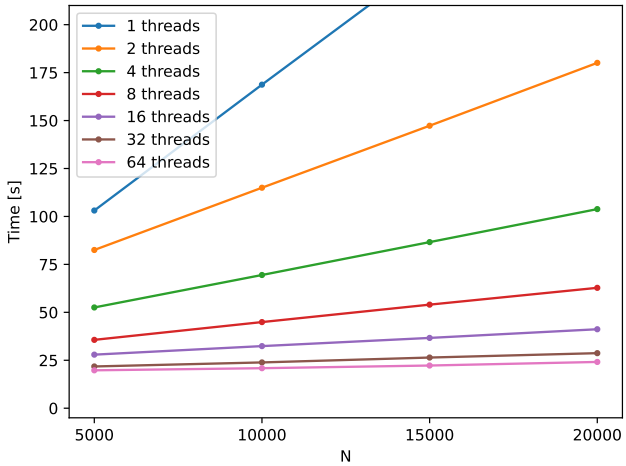



Figure 4: Duration of the SNR computation on the full ASCAD trace for the 86 profiled variables as a function of the number of profiling traces ($N$) and the execution parallelism.

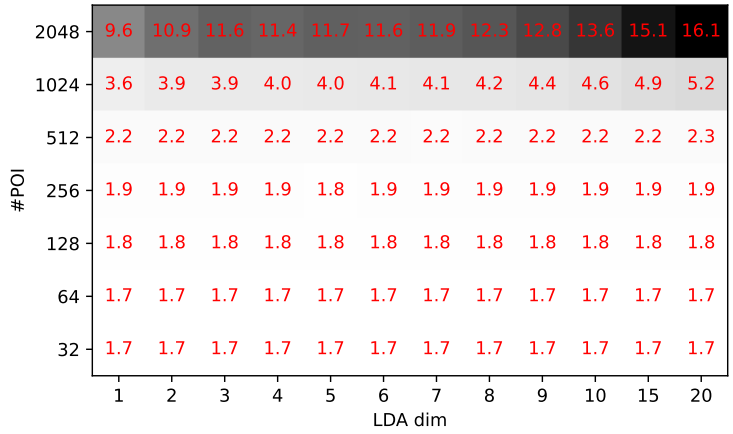
for each number of attack traces. This attack parameters are $N = 20\,000$, $D = 8$ and $P = 512$, the total execution time (profiling and attacks) is 12 s. We can see that if all bytes were giving similar results, 8 attack traces would be enough to perform a full key recovery attack. As expected, due to the reduction in the number of leakage points, this attack is less powerful than the one exploiting the full trace.

For attacks with multiple traces, we choose a random key $k$, then we label each trace with the plaintext $p_i = k \oplus k'_i \oplus p'_i$, where $k'_i$ is the true key and $p'_i$ is the true plaintext. This trick has no impact on the attack result since our attack only depends on $p \oplus k$, which is left invariant by the transformation.

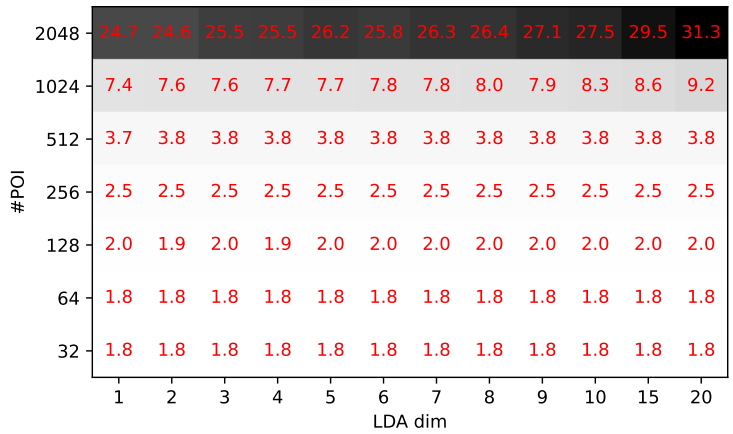
## B Attack execution time

We ran the attack on workstation with a AMD Ryzen Threadripper 3990X processor (at 2.90 GHz clock frequency) and with 256 GB RAM.

We next analyze the sensitivity of the execution time of the profiling phase to the meta-parameters. We give in Figure 4 the computation time for the SNR and in Figure 5 the




(a) $N = 5000$ profiling traces.




(b) $N = 20\,000$ profiling traces.

Figure 5: Duration in seconds of model computation for all of the 86 shares (LDA and Gaussian templates), running with a single thread.

computation time for the modeling (LDA and Gaussian templates). The computation time for the attack phase (probabilities extraction from the templates and belief propagation) takes about 20 ms in all considered cases.

We can observe that the SNR computation is the most computationally expensive stage of the attack: on a single core and for $N = 5000$, it takes 103 s, whereas the modeling all the shares takes at most 16 s. The SNR computation can however be accelerated by an order of magnitude when using a CPU with many cores.

**Human time complexity.** The methodology from [BS20, BS21] has been straightforwardly applied/adapted to the ASCAD dataset. Indeed, this methodology essentially requires to know the operations performed by the implementation (i.e., the source code) in order to adapt the factor graph used by the SASCA. The other steps do not require much work, since the algorithms are already implemented in SCALib and very little to no parameter tuning is needed.