Power leakage of a masked AES hardware implementation

The SMAesH dataset

Gaëtan Cassiers[1,2] and Charles Momin[1]

[1] UCLouvain, Louvain-la-Neuve, Belgium
[2] CryptoExperts, Paris, France

Abstract. Datasets of side-channel leakage measurements are widely used in research to develop and benchmark side-channel attack and evaluation methodologies. Compared to using custom and/or one-off datasets, widely-used and publicly available datasets improve research reproducibility and comparability. Further, performing high-quality measurements requires specific equipment and skills, while also taking a significant amount of time. Therefore, using publicly available datasets lowers the barriers to entry into side-channel research. This paper introduces the SMAesH dataset. SMAesH is an optimized masked hardware implementation of the AES with a provably secure arbitrary-order masking scheme. The SMAesH dataset contains power traces of the first-order protected version of SMAesH acquired on two FPGAs from different generations, along with key, plaintext and masking randomness. A part of the dataset use uniformly random key and plaintext to enable leakage profiling, while another part uses a fixed key (still with uniformly random plaintext) to enable attack validation or leakage assessment in a fixed-versus-random setting. We document the experimental setup used to acquire the dataset and also discuss particular methods employed to maximize the information content in the leakage traces, such as power supply selection, fine-grained trace alignment and resolution optimization. We perfom an in-depth leakage analysis for the two targets. Finally, we briefly discuss the known attacks against the dataset.

Keywords: Side-channel · Power leakage · Dataset · Masking · FPGA

1 Introduction

Physical side-channel attacks have been a threat to the security of embedded devices since their introduction [KJJ99]. Protecting a device against such attacks, as well as evaluating its security, is a challenging task, which led to the continuous improvement of countermeasures and attacks. In order to evaluate and compare attacks, the side-channel research community has been using public datasets of side-channel traces. Besides avoiding duplication of the measurement work, the usage of such datasets presents multiple advantages. First, this improves replicability of results, and therefore eases subsequent improvements of attacks. Further, the widespread use of public datasets in works designing new attacks makes their result much easier to compare. This also lowers the barrier to entry for the design of state-of-the-art attacks, sidestepping the need for equipment and skills to perform the measurements.

Many side-channel datasets have been introduced over the years, often in the context of side-channel attacks contests or challenges. Among these datasets, most of them

E-mail: gaetan.cassiers@uclouvain.be (Gaëtan Cassiers), charles.momin@uclouvain.be (Charles Momin)

correspond to fairly simple leakage structures such as non-protected implementations (such as AES_HD [BJP20]), or present very strong leakage due to being software implementations (e.g., DPA contest v4[1], ASCAD v1 [BPS+20] and v2 [MS21]). However, none of the widely used datasets covers a protected hardware implementations, with the only ones covering this design corner being (to the best of our knowledge) the AES_HD_MM [Fei14] dataset and the Spook CTF [BBC+21] (CHES 2020 challenge) datasets. These datasets are not very widely used, and it has been hypothesized that this is due to AES_HD_MM not containing random key traces (preventing profiled attacks) [PPM+23] and being broken within a low number of traces [WHJ+21], while the Spook dataset contains traces of an uncommon cipher (Clyde-128) [BBC+20, BBB+20].

Contributions We propose a new power leakage dataset of a masked hardware implementation of the AES: the SMAesH dataset. This dataset aims at being useful to the research community by being a realistic benchmark for attacks, while being not excessively difficult to attack. While the latter goal may seem surprising, we believe that very hard datasets have a limited interest to the research community: attacking such dataset requires many traces and large computational resources, which increases attack development iteration time and raises the barrier of entrance. In particular, care has been taken to ensure that, beyond the intrinsic security brought by masking, the dataset is as easy as possible to attack. Indeed, attacks requiring fewer traces are faster to run and require less computational resources (all other parameters being equal), and are therefore easier to work on.

These design goals are translated into the following dataset characteristics. First, the target is SMAesH, an open-source masked hardware implementation of the AES which is thoroughly documented. The implementation is reasonably simple to understand, but not artificially simple or unoptimized. Second, the security is concentrated on the masking countermeasure: we ensured that the masking has no big flaw, but did not add any other countermeasure such as added noise or clock jitter. Third, we optimized the acquisition setup parameters to maximize the amount of leakage, including power supply, clock signal, measurement device. Fourth, the acquisition setup uses devices that are easy to procure and are fully documented, with the aim of enabling its reproduction.

The SMAesH dataset contains measurements for two devices: a Xilinx Artix-7 FPGA (XC7A100T) on a Chipwhisperer CW305 board (next denoted A7), and a Xilinx Spartan-6 (C6SLX75) on a Sakura-G board (next denoted S6), a more leaky (thus easier to attack) target than A7. The dataset contains a total of 6×2^{24} traces of respectively 4450 samples for A7 and 4400 samples for S6, covering the execution of the first round of the AES.

The challenge associated to CHES 2023 was a side-channel analysis contest based on the SMAesH dataset. The attacks against the A7 target were performed in a worst-case setting (i.e., the design was fully public, including masking randomness for the profiling and validation data), and improved down to 2.9×10^5 traces. On the other hand, for the S6 target, the attack setting was more restricted (only the source code was public and not the bitstream, while the masking randomness was not public), which led to the best attack requiring 9.0×10^5 traces.

Organization We first introduce the SMAesH implementation used as the target for the collection of our dataset. We then discuss the high-level dataset structure, the configuration of the target chips, and the acquisition setup. We then perform an analysis of the structure of the leakage and of its information content. Finally, we report and discuss the attacks submitted during the CHES 2023 challenge.

[1] https://dpacontest.telecom-paris.fr/home/

2 SMAesH

SMAesH[2] is an open-source masked hardware implementation of the AES. In version 1.0.0 (the one used for this dataset), the core supports the encryption for the 128-bit key variant of the AES algorithm. The implementation is based on the Hardware Private Circuits (HPC2) masking scheme, which provides state-of-the-art guarantees in terms of resistance against physical defaults (e.g., glitches) and composability. It is therefore provably secure at arbitrary masking order and has been formally verified by the fullVerif tool [CGLS21].

As depicted in Figure 1, SMAesH relies on a pipeline 32-bit architecture. It instantiates 4 masked S-boxes that are shared between the round operations (namely `AddRoundKey`, `SubBytes` and `MixColumns`) and the key scheduling computations. The module `MSKaes_32bits_state_datapath` is a 4 layers shift-register used to store the masked 128-bits state across the different rounds executions. It embeds the combinational logic (i.e., 32 XORs) performing the `AddRoundKey` layers sequentially in 4 cycles, and forwarding the resulting masked 32-bit values through the bus `sh_4bytes_to_SB`. Additionally, it implements the `ShiftRows` layers solely relying on routing (i.e., without dedicated logic). Similarly, the module `MSKaes_32bits_key_datapath` is also a 4-stages pipeline used to handle the key scheduling operations. In particular, it stores the masked 128-bit key and forward masked 32-bit key material to the `AddRoundKey` logic through the bus `sh_4bytes_to_AK`. During the key scheduling operation, the key material are sent to the S-boxes through the bus `sh_4bytes_rot_to_SB`. Finally, the module `MixColumns` is a combinational logic block operating on a single masked column of 32 bits. It takes as input the bytes coming from the S-boxes and forwards the results back to the state holder (the module can be bypassed using a dedicated MUX during the computation of the last round).

An single execution of SMAesH is performed by compute each round sequentially, with the operands looping across the pipeline. As depicted for the first round of the execution in Figure 2, a round starts (i.e., corresponding to the cycle where `cnt_round` $= 0$ and `cnt_fsm` $= 0$) by sequentially computing the `AddRoundKey` operation, operating column per column, and sending the results (denoted `pAK`) to the S-boxes. This process is done in 4 cycles, and each S-boxes execution has a latency of 6 cycles. Therefore, the resulting values are valid at the S-boxes' outputs spanning from the cycles 6 to 9. Eventually, the `MixColumns` layers is applied to every column coming from the S-boxes before being forwarded back to the state holder. In the same time, the key scheduling operation is performed in parallel to the round computation. This process starts one cycle before starting a new round (or during the very first cycle of the execution) by sending the key material to the Sbox. The resulting values are then forwarded back to the key holder using the dedicated port `sh_4bytes_rot_from_SB`. Overall, the latency of an execution equals $1 + 10 * 10 + 4 = 105$ cycles, where the term 1 is caused by the very first cycle required to start the key scheduling and the 4 comes from the final key addition performed internally to the state holder. As a final note, the randomness needed for masking is generated on the fly by an embedded PRNG based on an instance of Trivium as supported by recent work reported in [CMM+24].

The synthesis parameters of SMAesH are `D`, the number of shares and `PRNG_MAX_UNROLL`, a limit on the unrolling of the Trivium PRNG (which allows adjusting the critical path vs area trade-off), as proposed in [CMM+24]. We used `D=2` and `PRNG_MAX_UNROLL=128` (leading to the instantiation of two Trivium cores).

3 Dataset

Two datasets have been acquired for each target:

[2] https://github.com/simple-crypto/SMAesH

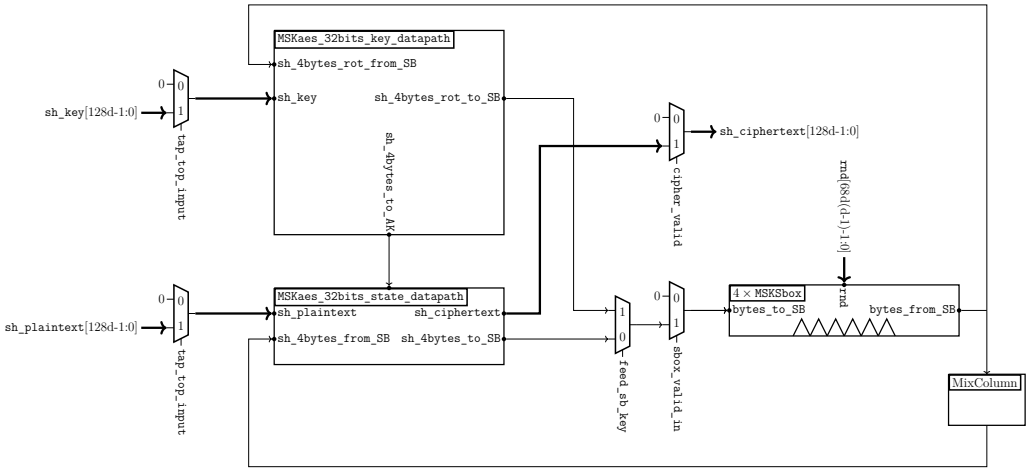

Figure 1: Top level architecture of the SMAesH IP. Bold wires represent masked 128-bit buses, and remaining wires (other than Muxes control, and randomness buses) depict masked 32-bit buses.

Table 1: SMAesH v2 datasets

Name[a]	Target	Original Role	Number of traces	Trace length (n_s)
`SMAesH-A7_d2-vk0`	Artix-7 ($d = 2$)	Profiling	2^{24}	4250
`SMAesH-A7_d2-fk0`		Validation		
`SMAesH-A7_d2-fk1`		Test		
`SMAesH-S6_d2-vk0`	Spartan-6 ($d = 2$)	Profiling	2^{24}	4400
`SMAesH-S6_d2-fk0`		Validation		
`SMAesH-S6_d2-fk1`		Test		

[a] The names are intended as unique dataset identifiers. Future dataset versions will not reuse the same dataset names (except when re-packaging the exact same data).

- A training dataset that uses a fresh random key for each trace.

- A validation dataset that uses a single key for the whole dataset.

All datasets use a fresh random plaintext for each trace and make a correct use of the SMAesH core: for each trace, the sharing of the key and of the plaintext is fresh. Moreover, although not necessary for security, we reseed the core before each trace with a fresh seed (the reseeding is not included in the trace), in order to ease the simulation of the computations performed by the target. The dataset characteristics as summarized in Table 1, The dataset is available at [CM23].

For each trace, the datasets contain the power leakage trace, as well as all the data required to exactly replicate the measured execution, as shown in Table 2 (the `umsk_plaintext` and `umsk_key` can be derived from `msk_plaintext` and `msk_key` respectively, they are provided for convenience only).

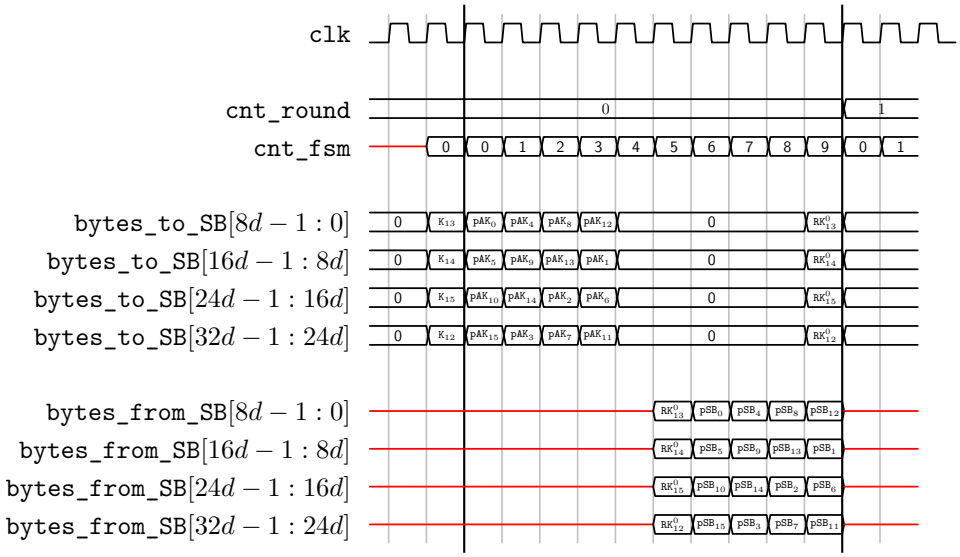

Figure 2: Timing diagram relative the S-boxes input/output behavior during the first round of an execution of SMAesH.

Table 2: Dataset fields

Label	Type	Length	Description
`traces`	`int16`	n_s	Power trace.
`umsk_plaintext`	`uint8`	16	Non-masked plaintext.
`umsk_key`	`uint8`	16	Non-masked key.
`msk_plaintext`	`uint8`	$16d$	Plaintext shares (each 16-byte chunk is a share).
`msk_key`	`uint8`	$16d$	Key shares (each 16-byte chunk is a share).
`seed`	`uint8`	10	PRNG seed.

4 Target designs

4.1 Artix-7

The FPGA bitstream used to perform the acquisitions has been generated using the Xilinx Vivado Toolset (v2022.1 64-bit) and the following modifications have been applied compared to the default toolflow parameters:[3]

- HDL annotation:
  - attribute `DONT_TOUCH` set for every module.
  - attribute `KEEP_HIERARCHY` set for every module.
- Synthesis parameters:
  - `flatten_hierarchy` set to `none`
  - `gated_clock_conversion` set to `off`
  - `bufg` set to 12
  - `directive` set to `Default`
  - `no_retiming` `checked`
  - `fsm_extraction` set to `auto`
  - `keep_equivalent_registers` checked
  - `ressource_sharing` set to `off`
  - `no_lc` checked
  - `no_srlextract` checked
- Implementation parameters:
  - `opt_design` related: `is_enabled` unchecked
  - `phys_opt_design` related: `is_enabled` unchecked

The acquisition setup programs the microcontroller of the CW305 board with a tweaked version of the newAE-provided firmware. The changes increase the flexibility in data sent to the FPGA (such as seeds, shared values, etc.) and add the ability to perform quick interleaved acquisition of multiple datasets. In more details, the microcontroller can run a full "batch" of traces without interaction with the control computer. At each trace, the microcontroller selects randomly a dataset to which the trace belongs (in our case, `-fk0` or `-vk0`), generates the required inputs and sends them to the FPGA. All the operations are performed in constant time, such that the FPGA traces are as similar to each other as possible. The randomized interleaving ensures that there is no systematic bias in the datasets, which allows using them for a fixed-vs-random TVLA. All the randomness of the microcontroller is drawn from a PRNG seeded by the control computer, which allows it to reconstitute the data sent to the FPGA.

4.2 Spartan-6

The FPGA bitstream used to perform the acquisitions has been generated using the Xilinx ISE Toolset (v14.7 lin64) and the following modification have been applied to the default toolflow parameter:

- HDL annotation:
  - attribute `DONT_TOUCH` set for every module.
  - attribute `KEEP_HIERARCHY` set for every module.
- Synthesis parameters:

[3]The Vivado project used to generate the bitstream is available at ANONYMIZED .

  - `Optimization Effort` set to `Fast`
  - `Keep Hierarchy` set to `Yes`

- Map Properties:
  - `Placer Effort Level` set to `Standard`
  - `Generate Detailed MAP report` set to `checked`
  - `Use RLOC Constraints` set to `No`
  - `Enable Multi-Threading` set to `2`

- Place & Route Properties:
  - `Place & Route Effort Level (Overall)` set to `Standard`
  - `Enable Multi-Threading` set to `4`

The controller FPGA of the Sakura-G board has been designed in order to handle the same interleaving feature as the one implemented on the CW305 controller used for the Artix7 target.

5 Experimental setup

5.1 Artix-7

The power traces have been acquired by measuring the signal at the `X4` point (directly connected to the oscilloscope with a SMA cable), which corresponds to the voltage drop across the 100 mΩ shunt resistor `R27` amplified with a low-noise amplifier. The target FPGA is powered through the dedicated banana connectors (with the `SW1` accordingly set) by an external low noise power supply Keysight E36102B set to a 1 V DC voltage. This setup reduces the noise level compared to the on-board power supply derived from the 5 V USB supply by a switching voltage converter.

The leakage is measured by a PicoScope 6242E digital oscilloscope. The target FPGA and oscilloscope clocks are synchronized in order to reduce the level of noise induced by clock jitter. This is achieved by configuring the CDCE906 PLL of the CW305 board to generate two clocks signals based (derived from the 12 MHz crystal of the CW305). The first is the FPGA clock, running at 1.5625 MHz generated by the `PLL1` and fed to the port `N13` on the FPGA. The second is a 10 MHz signal generated by the `PLL0` and fed routed to the `X6` SMA connector. It is then forwarded to the PicoScope 10 MHz clock reference input port. A single measurement channel (channel A) is used to perform the measurement and the trigger signal is fed from the onboard test point `TP1` to the oscilloscope AUX trigger port with a PicoScope probe (Picotech TA386, 1:1 ratio, 200MHz of bandwidth). The power traces are sampled at 5 GS/s using a vertical resolution of 10 bits.

The clock configuration aims at minimizing the jitter between the target FPGA clock and the oscilloscope sampling clock. In particular, the configured frequencies result in exactly 3200 samples per target clock cycle, and the relative phase of the clocks are fixed by the PLLs. While this setup has been observed to give a low clock jitter and therefore an excellent alignment of consecutive traces, the relative phase of the clocks may drift over time, resulting in slight misaligments of the traces when measured over a long time span. This drift has been mitigated by increasing the sampling frequency to 5 GS/s, which is beyond the useful signal bandwidth, but limits the possible clock drift to 200 ps. The signal is then down-sampled with a moving average filter, with a decimation factor of 16, leading to a final sampling frequency of 312.5 MS/s, an a vertical resolution of 14 bits.

The target clock frequency has been selected as the largest sub-multiple of the oscilloscope sample rate for which the leakage caused by both edges of the clock can be clearly identified in the leakage trace, as recommended by [BUS21]. Then, the oscilloscope sample rate is selected based on the perceived information [BHM+19, MCHS23] on the shares: in

order to minimize the dataset size, we selected the lowest sample rate that does not result in significantly degraded information content.

The acquisition has been performed in a room without accurate temperature control, but with relatively stable temperature (air conditioning) and no direct sunlight. The total duration for the acquisition of the Artix-7 datasets is 21 hours each (224 traces/s). In order to minimize the dataset size, the samples selected in the final dataset cover only the leakage of the first round of the AES (this has been determined by means of SNR computation on all the variables of the first round: all the SNR peaks are kept in the trace).

5.2 Spartan-6

The acquisition setup for the Spartan-6 target is similar to the one for the Artix-7 target. We discuss the few differences below.

The datasets contain power traces that have been acquired by measuring the voltage drop across a $2\,\Omega$ shunt resistor placed at on `JP2`. This voltage drop is amplified by the on-board amplifier, and measured at the `J3` point through a SMA cable. The whole Sakura-G board is powered by an external low noise power supply Keysight E36102B set to a 5 V DC voltage.

Regarding the oscilloscope configuration, the external trigger signal is connected to a GPIO header of the board. Further, the clock synchronization is achieved by generating the clock signal with the signal generator of the oscilloscope (1.5625 MHz square wave, 2V peak-to-peak amplitude, 1V offset) and feeding it directly to the SMA connector J6P on the board (i.e., clock-capable SMA for controller FPGA). This signal is directly used as a clock by the target (clock buffers are used in the design, but no PLL is used). Finally, the oscilloscope performs acquisition at 1.25 GS/s (resulting in 800 samples per target clock cycle) using a vertical resolution of 12 bits, and the post-processing uses a decimation factor of 4 to reduce this to 312.5 MS/s with 14-bit vertical resolution.

The clock generation method used for the Spartan-6 eliminates the drift observed on the Artix-7 acquisitions, which can be explained by the absence of any PLL between the oscilloscope acquisition clock and the target clock. Therefore, a very high sample rate was not needed, and a lower sample rate was used (along with a higher vertical resolution) in the interest of a faster acquisition. After downsampling, both targets' datasets have the same sample rate and resolution.

6 Worst-case security analysis

In this section, we peform a worst-case analysis (i.e., assuming complete knowledge of the design) for both datasets. We begin with qualitative analysis of the collected traces, by computing the Signal-to-Noise ratio for different variables of interest inside the first round of the AES and complement our analysis with more quantitavie analysis by estimation the Mutual Information for these variables, which leads us to approximate the attack complexity of state-of-the art attack strategy.

6.1 Qualitative leakage analysis

Additive noise assumption and Signal-to-Noise Ratio A side-channel leakage trace $L = [t_0, t_1, ..., t_{n_s}]$ is a vector containing the n_s power/EM measurements acquired with the measurement aparatus during a single execution (denoted next the time samplesa), representing the instantanous current consumption of the device. Under the classical additive noise assumption, every time sample can be modelled as

$$t_i(X) = \delta(X) + N$$

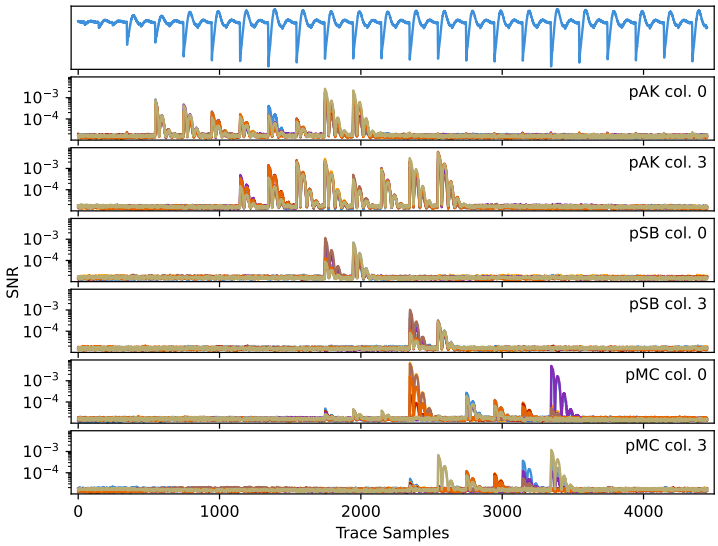

Figure 3: SNR for the bytes of the first and last column processed by SMAesH, after the AddRoundKey, SubBytes and MixColumn layers for the A7_d2 dataset.

where δ is a deterministic function used to model the impact of the target variables X on the global consumptions and N is the random noise. The latter typically models the impact on the overal device consumption of the components not modelled by X as well as the intrinsic physical noise. Next, we denote a set of q traces as $\mathbf{L} = [L_0, L_1, \ldots, L_q]$ where $L_{j\in[0,q-1]} = [t_0^j, t_1^j, \ldots, t_{n_s}^j]$ is the j-th trace.

Under this additive assumption, the Signal-to-Noise Ratio (SNR) is a common tool used to identify Point-of-Interest (POI) in side-channel traces [Man04]. Informally, it consists in evaluating the ratio between the variation of the current consumpion relative to the target variable and the random noise. It is computed as

$$\mathsf{SNR}_i = \frac{\mathsf{Var}_x\left(\mathsf{E}\left[\mathbf{t}_i(X=x)\right]\right)}{\mathsf{E}_x\left[\mathsf{Var}\left(\mathbf{t}_i(X=x)\right)\right]}$$

where Var denotes the variance, E the mean and $\mathbf{t}_i(X=x)$ is the vector containing the i-th time sample of every possible traces under the assumption that $X=x$. In practice, it is usually approximated taking the sample mean and variance considering a set of traces of fixed size.

As a first step, we performed a SNR analysis for both the A7_d2 and S6_d2 as a preliminary step in order to identify the most leaking points of the traces (our points of interest – POIs). More particularly, we computed the SNR associated to both shares of every state's byte after the main steps performed in the first round, namely the bytes resulting from the initial key addition denoted pAK, at the output of the Sboxes layer denoted pSB and at the output of the MixColumn operation denoted pMC (as explained in the IP's documentation, the ShiftRows is implemented at the routing level without dedicated logic and is therefore not of particular interest). Next, the SNR computation were made relying on 2^{24} traces of the variable-key dataset associated to both target (namely, SMAesH-A7_d2-vk0 and SMAesH-S6_d2-vk0), using SCAlib's SNR implementation [CB23].

In the remaining of the paper, we refer to the state's byte manipulated by using indexes ranging from 0 to 15. In order to simplify the results interpretation, the byte indexing follows the ordering of the initial state and keys across the different steps of the round, without considering the indexes change involved by the ShiftRows layer (e.g., when referring to i-th byte of pSB, we refer to the byte resulting from the SubBytes operation

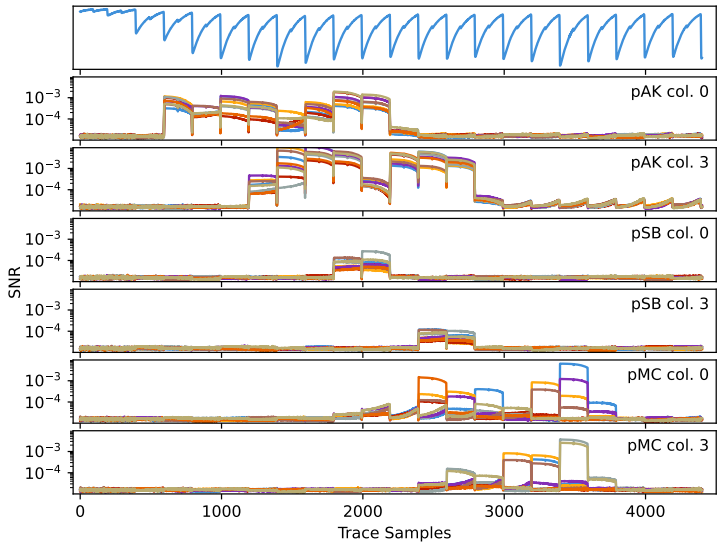

Figure 4: SNR for the bytes of the first and last column processed by SMAesH, after the AddRoundKey, SubBytes and MixColumn layers for the S6_d2 dataset.

of the i-th byte of pAK).

Starting with the Artix-7 target, the SNR results associated to the first and last columns processed during an execution of SMAesH (i.e., byte indexes in [0,5,10,15] for the first and [12,1,6,11] for the last) are depicted in Figure 3. As a first observation, we notice that all the operation performed are leaking with somehow similar SNR withing localized area in the traces. The initial key addition starts at time index 547, which we seems to correspond to the second cycle of the execution (since it is the second cycle consuming a significant amount current). Similarly, the key addition related to the last column happens at the 1047 time index, i.e., three cycles after the first one, which is coherent with the behavior described in the IP's documentation. Interestingly, SNR related to these operation are still observable for the 7 cycles following their computation. We believe that these behavior are caused by the bytes resulting from the key addition that are forwarded back to the data pipeline at the same cycle they are entering in the Sbox (and are then manipulated for following cycles until the completion of the MixColumn operation). The results of Sboxes output for the first (resp. last) column appear at time index 1748 (resp. 2300) , or 6 cycles after the key addition, which is once again coherent with the IP's documentation. The computation of the (combinational) MixColumns operation is synchronized with the Sboxes results, and signal remain observable during several following cycle due to the fact that some of them remain active in the data pipeline during the key addition of the following round. Interestingly, the same discussions holds for the Spartan-6 target, as shown in Figure 4. The main notable difference is that the SNR peaks are less pronounced and appears more like decreasing plateau lasting a full clock cycle (probably due to capacitive effects).

As a second step, we focused on the execution of the Sboxes, that are organized as a 6 layers pipeline and therefore are targets of particular interest with multiple leaking points. More particularly, each masked Sbox instance is a masked implementation of the Boyard Peralta representation that internally manipulates 128 (shared) bit-wide variables from which signal can be obtained. In particular, the Figure 5 and Figure 6 depict the maximum SNR value obtained (taking into account both shares of all the variables but excluding input and output) associating to the processing of the four columns of the state. We observe the similar similar behavior for both target, with peaks starting at the same

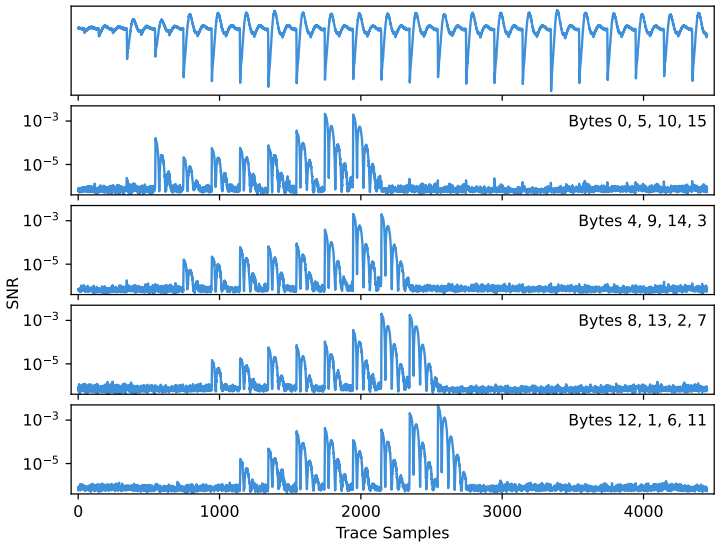

Figure 5: SNR for the bytes after the AddRoundKey, SubBytes and MixColumn layers for the A7_d2 dataset.

location than the key addition. Interestingly, we observe that despite that Sboxes consist in 6 layers pipeline, SNR are observable over 8 cycles. Without clear explanation relative to the architecture, we note the in depth research of this signal's source as an interesting line for further works.

6.2 Quantitative leakage analysis

In this section, we perform an in depth analysis of the exploitable information that can be extracted from the datasets. We perform a "worst-case evaluation" of the SMAesH IP by trying to quantify the best attack complexity (i.e., the number of traces required to successfully perform a key recovery), assuming complete knowledge of the design and unrestricted leakage profiling capability (i.e., knowledge of all inputs during profiling, including the seed of the mask's PRNG). For the purpose of this section, the A7_d2-vk and S6_d2-vk datasets were each randomly partitioned into a profiling set containing $2^{24} - 2^{21}$ traces and a validation set made of the remaining 2^{21} traces.

Linear Subspace Template Attack The template attack [CRR02] is a powerful profiled attack strategy when the values of the targeted intermediate variable is known at profiling time. It works by modeling the leakage as a multivariate Gaussian distribution whose mean depends on the intermediate variable (in pooled template attacks, the variance is assumed to be independent of the intermediate variable). When the number of points of interest is large, the method can be made more robust by using linear discriminant analysis (LDA), which linearly projects the leakage into a space of lower dimension p before fitting the templates [SA08]. This reduces greatly the dimensions of the means vectors and covariance matrix to estimate, reducing the number of profiling traces required for reliable estimation. The projection matrix itself is computed from the profiling data, in such a way that the SNR along the projected dimensions is maximized (and the projected dimensions must be orthogonal to each other). The likelihood of the leakage L (after selection of the m POIs in the trace with the highest SNR) conditioned on the intermediate variable X is

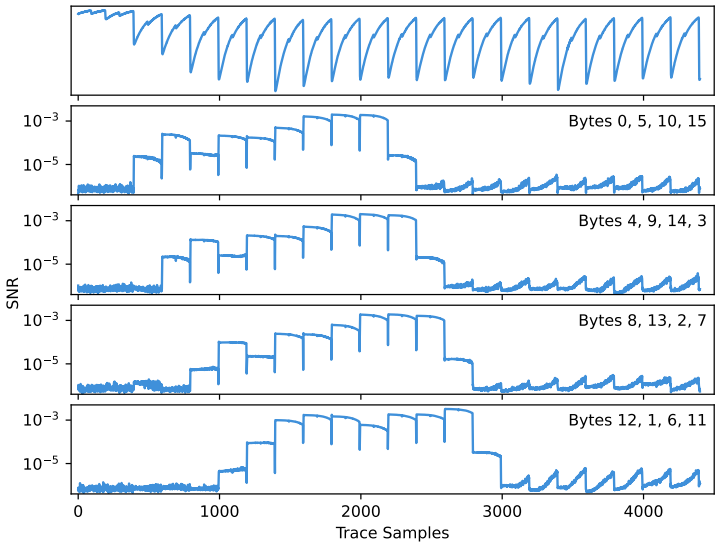

Figure 6: SNR for the bytes after the AddRoundKey, SubBytes and MixColumn layers for the A7_d2 dataset.

thus estimated as

$$\hat{\mathsf{f}}\left(L=l|X=x\right)=\frac{1}{\sqrt{(2\pi)^{p}\,\mathsf{det}(\hat{\Sigma}_{x})}}\mathsf{exp}\left(-\frac{1}{2}(\mathbb{W}l-\hat{\mu}_{x})^{T}\hat{\Sigma}_{x}^{-1}(\mathbb{W}l-\hat{\mu}_{x})\right)$$

where $\mathbb{W}$ is the $p \times m$ projection matrix, $\hat{\Sigma}_x$ is the $p \times p$ estimated covariance matrix and $\hat{\mu}_x$ is the p-dimensional mean vector. The estimated distribution for the intermediate variable $\hat{\Pr}[X = x|L = l]$ (hereafter the "model") can then be computed using Bayes law.

Information Theory Security Metrics The Mutual Information (MI) quantifies the amount of information related to a target variable that can be extracted from a single trace, and has been shown to be theoretically linked to the trace complexity of a side-channel attack [dCGRP19]. However, computing it requires knowledge of the exact leakage distribution, therefore surrogate metrics are used. In particular, the Perceived Information (PI) [RSV+11] is a lower-bound for the MI [BHM+19] that can be efficiently computed. Given a model built using the profiling set $\mathcal{L}_p$ and an independent validation set $\mathcal{L}_v$, the PI can be estimated as follows:

$$\hat{\mathsf{PI}}\left(L;X\right)=\mathsf{H}\left(X\right)+\frac{1}{|\mathcal{L}_{v}|}\sum_{(x,l)\in\mathcal{L}_{v}}\log_{2}(\hat{\Pr}[X=x|L=l])$$

where H is the Shannon entropy. Informally, the PI represent the amount of information that a specific model can recover about the target variable. We also use the Training Information (TI) [MCHS23] which is an upper bound to the PI and is computed similarly to the PI, but using the training set in place of the validation set. Evaluating the gap between the PI and the TI as a function of the amount of profiling traces is a way to identify if the profiling phase of a model has converged, or if a higher PI could be exploited by increasing the size of $\mathcal{L}_p$.

Information leakage analysis We consider multiple choices for the parameters of the LDA. We take $m = 64, 128, \ldots, 2048$, which goes from a very small subset of the trace to

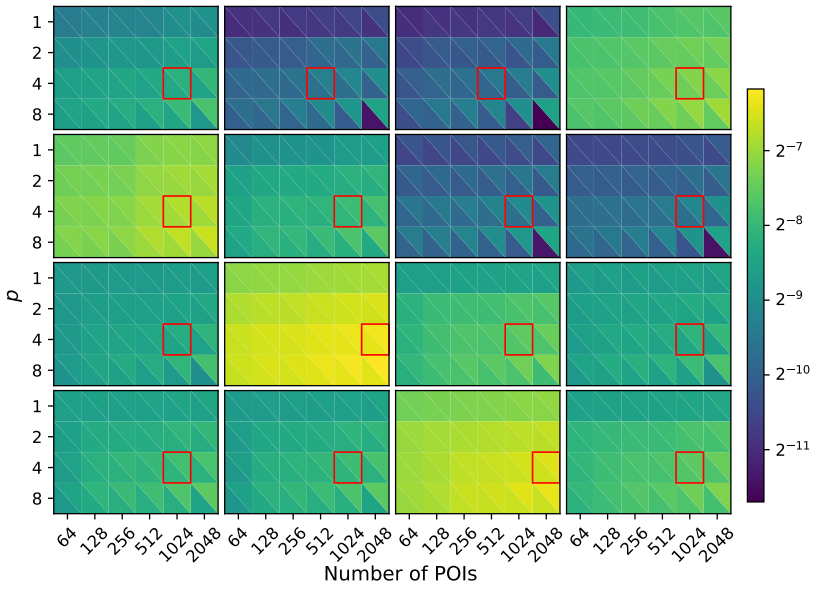

Figure 7: Parameter selection for the first share of pAK bytes on A7_d2 (the plot order follows AES input state byte order, e.g., first column is bytes 0 to 3). The lower (respectively upper) triangles represent the PI (resp. TI), with the red square indicating the best model (highest PI).

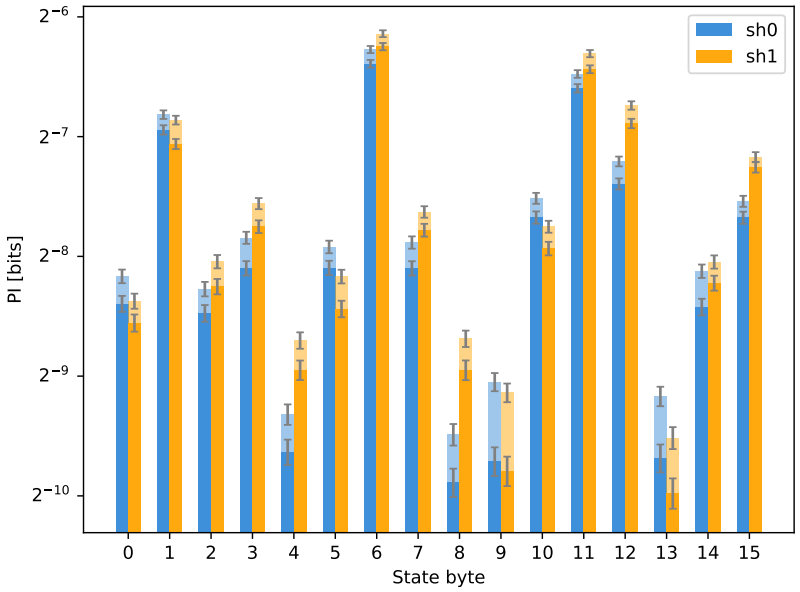

Figure 8: PI (vivid) and TI (pastel) for the best model for the shares of each pAK byte on A7_d2.

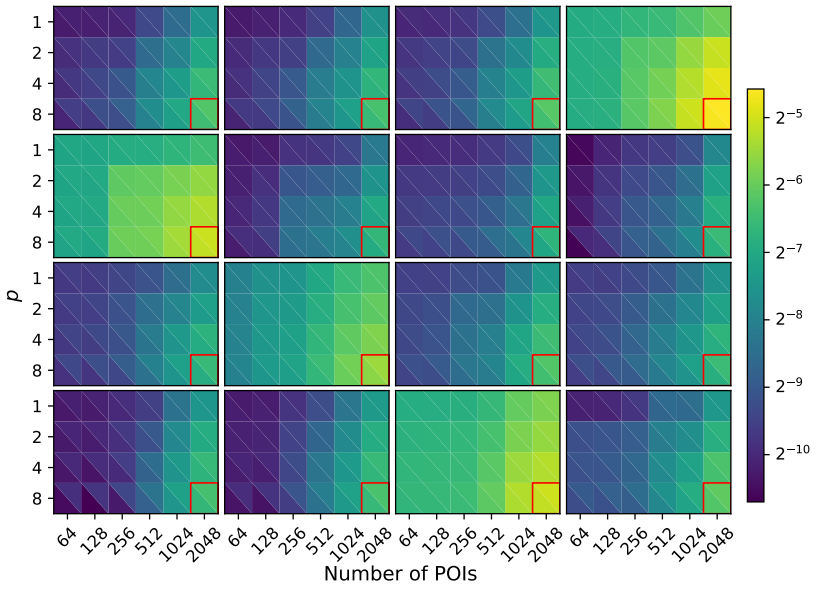

Figure 9: Parameter selection for the first share of pAK bytes on S6_d2 (the plot order follows AES input state byte order, e.g., first column is bytes 0 to 3). The lower (respectively upper) triangles represent the PI (resp. TI), with the red square indicating the best model (highest PI).

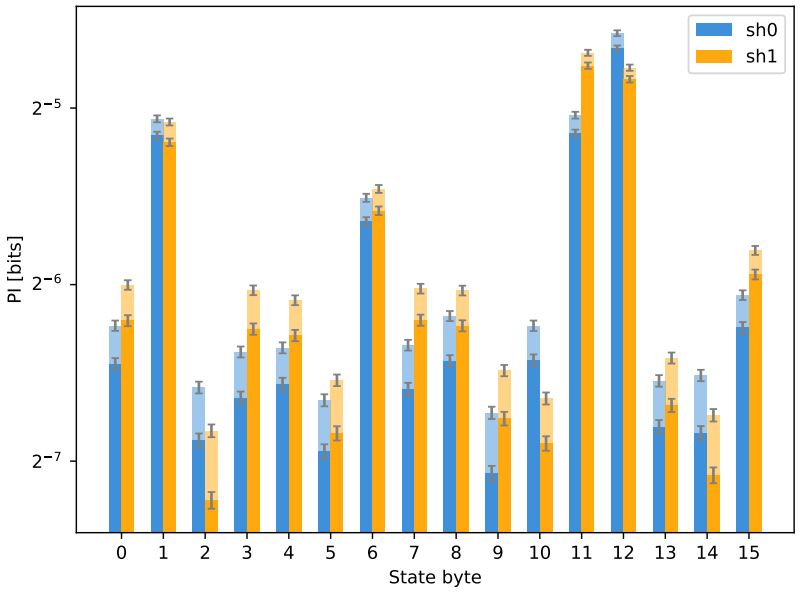

Figure 10: PI (vivid) and TI (pastel) for the best model for the shares of each pAK byte on S6_d2.

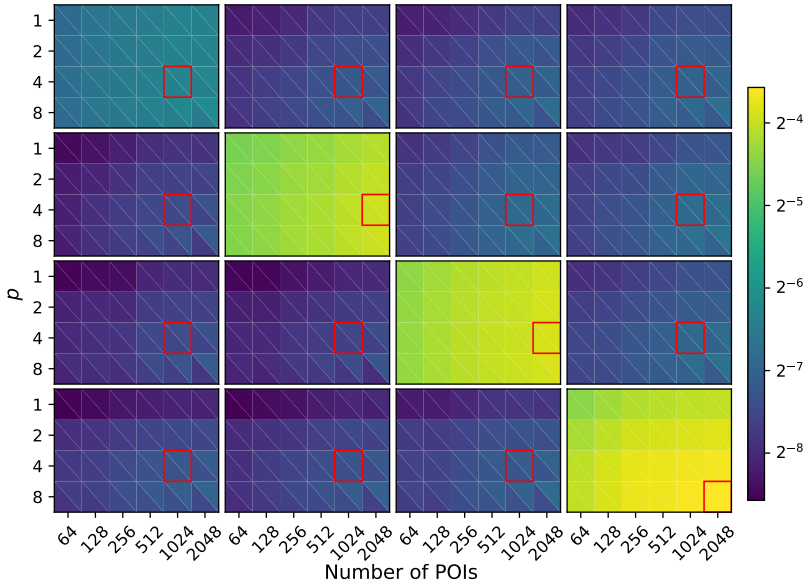

Figure 12: Parameter selection for transition leakage on the first share of pAK bytes on A7_d2 (the plot order follows AES input state byte order, e.g., first column is bytes 0 to 3). The lower (respectively upper) triangles represent the PI (resp. TI), with the red square indicating the best model (highest PI).

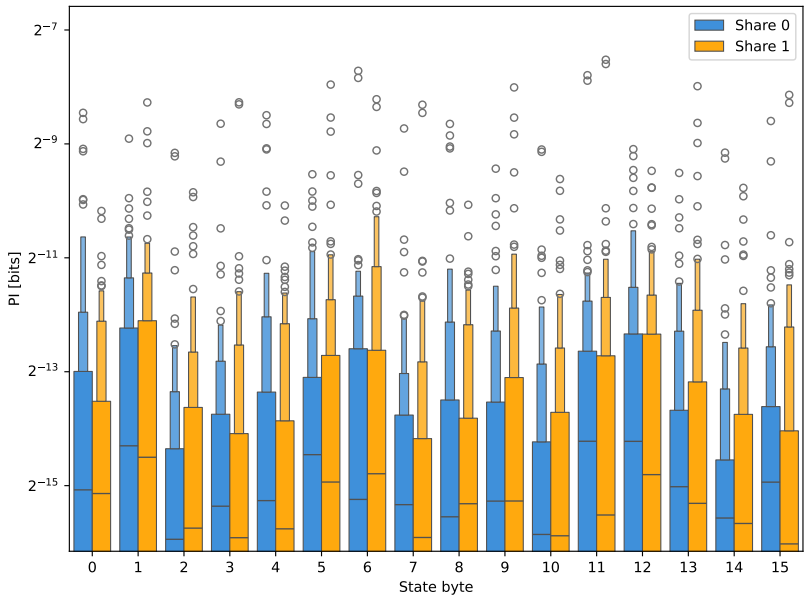

Figure 13: Distribution of the PI for the shares of the intermediate single-bit variables in the S-boxes, for each byte. The distributions are represented as letter-value plots [HKW11], where the largest horizontal line representing the median and then each rectangle above (and below) representing ranges of exponentially-decaying probability (25 %,12.5 %,...). For each variable, the LDA parameters maximizing the PI are chosen.

almost half of the trace. Increasing the upper limit above 2048 would be computationally very intensive and would likely not bring significant improvements given the number of leaking points observed in Subsection 6.1. For the number of dimensions, we use $p = 1, 2, 4, 8$ since higher values do not seem very useful, as shown in previous works [BCS21, CDSU23].

The PI and TI for these parameters is shown in Section 6.2 for the first share of pAK on A7_d2 and for S6_d2 in Figure 6.2. The results for both shares is represented in Figure 6.2 and Figure 6.2 for A7_d2 and S6_d2 respectively, using the best model parameters (i.e., the one with the highest PI). On these plots, we also represent the confidence intervals on the value of the PI and the TI to validate that the validation/training (sub)set is large enough (PI and TI are essentially means over a trace set).

The most noticeable characteristics of these plots is the higher leakage of the bytes $[1, 6, 11, 12]$, which correspond to the last column to be fed through the S-box in the round. After this last column, the input of the S-box has both shares set to 0, which means that the corresponding transition leakage only depends on the shares of the last column, leading to a higher amount of leakage than for the other columns where the transition occurs between two column (the first column is preceded by the round key update). We also observe that the leakage for the S6_d2 dataset is higher and is best exploited using a high number of POIs. This observation correlates with our observation about the SNR in Subsection 6.1: the many POIs with high SNR lead to more information leakage, even though the SNR peaks are not larger than A7_d2 (e.g., they may enable better noise averaging). Further, on A7_d2, we can see that the first column fed to the S-box (bytes $[0, 5, 10, 15]$) are also leaking more than the others, which may be due to higher leakage signal coming from the multiplexer switching between the key storage and the state storage, or to the lower noise level at the beginning of the execution (less activity in the circuit). We see a similar effect on the transition leakage on the S-box input (i.e., considering the XOR between the previous and current value of a byte as the target variable) in Figure 6.2 (which overall has higher leakage than the plain value model). Finally, still for A7_d2, we see effects of the synthesis and routing: two out of the four hardware S-box instances are less leaky than the others: the ones processing the bytes in the first two rows of Section 6.2.

Lastly, we look at single-bit variables inside the S-box in Figure 6.2. Given the large number of intermediate values in the Boyar-Peralta S-box [BP12], we depict the distribution of the variable's PIs. We see that quite a few variables leak significantly, indicating that these leakages may also be an attack vector. Similarly to pAK, we see that the bytes $[1, 6, 11, 12]$ leak more than the other columns.

6.3 Worst-case security bounds

In this section, we exploit the PI computed in Subsection 6.2 to derive bounds and estimates on the attack trace complexity of various attack paths. For this purpose, we use the 2-shares version of the bound of Béguinot et al. [BCG+23, Corollary 1]: if (X_0, X_1) is a Boolean sharing of X, then

$$\mathrm{MI}(X; L) \leq H_b\left(H_b^{-1}(\mathrm{MI}(X_0; L)) \star H_b^{-1}(\mathrm{MI}(X_1; L))\right)$$

where $H_b(x) = -x\log_2(x) - (1-x)\log_2(1-x)$ is the binary entropy function with $H_b^{-1} : [0, 1] \to [0, \frac{1}{2}]$ its inverse restricted to $[0, \frac{1}{2}]$, and $\star$ is the binary convolution: $x \star y = (1-x)y + x(1-y)$. This allows us to bound the the leakage on bit and byte sharings, which we then use to bound the number of traces needed for key recovery [dCGRP19, Theorem 2]:

$$q \geq \frac{2^n - (1 - P_s)\log_2(2^n - 1) - H_b(P_s)}{\mathrm{MI(X; L)}}$$

where q is the number of attack traces, n is the number of bits of the secret and P_s is the success rate. Where multiple variable are used together, we further use the bound $\mathrm{MI}((X, Y), L) \leq \mathrm{MI}(X, L) + \mathrm{MI}(Y, L)$.

Table 3: Bound on PI and estimation of required number of attack traces for 50 % recovery success rate for each of the A7_d2 key bytes, using the leakage on the S-box shared input/output byte, or on all S-box shared bits.

	pAK		pSB		SB bits	
Byte	PI	Traces	PI	Traces	PI	Traces
0	4.37×10^{-5}	68 670	2.11×10^{-5}	142 649	3.53×10^{-5}	85 122
1	3.36×10^{-4}	8929	1.53×10^{-5}	195 879	5.48×10^{-5}	54 835
2	5.11×10^{-5}	58 771	9.46×10^{-7}	3 173 567	2.25×10^{-5}	133 377
3	9.37×10^{-5}	32 039	5.01×10^{-6}	599 707	7.62×10^{-5}	39 385
4	1.40×10^{-5}	214 597	1.76×10^{-5}	170 357	3.39×10^{-5}	88 613
5	5.84×10^{-5}	51 417	3.05×10^{-5}	98 339	5.63×10^{-5}	53 316
6	8.64×10^{-4}	3474	3.48×10^{-7}	8 630 911	1.82×10^{-4}	16 481
7	9.18×10^{-5}	32 709	4.27×10^{-6}	703 587	6.65×10^{-5}	45 164
8	1.18×10^{-5}	253 525	2.15×10^{-5}	139 800	3.09×10^{-5}	97 252
9	7.48×10^{-6}	401 631	2.64×10^{-5}	113 732	5.48×10^{-5}	54 749
10	1.11×10^{-4}	27 116	9.01×10^{-7}	3 334 332	2.73×10^{-5}	109 884
11	6.58×10^{-4}	4561	1.26×10^{-6}	2 382 317	2.69×10^{-4}	11 178
12	2.77×10^{-4}	10 841	8.57×10^{-6}	350 223	4.40×10^{-5}	68 197
13	6.70×10^{-6}	448 209	2.63×10^{-5}	114 034	5.03×10^{-5}	59 710
14	5.41×10^{-5}	55 535	9.13×10^{-7}	3 289 226	2.33×10^{-5}	128 897
15	1.77×10^{-4}	16 981	4.73×10^{-6}	634 729	8.42×10^{-5}	35 646

Table 4: Bound on PI and estimation of required number of attack traces for 50 % recovery success rate for each of the S6_d2 key bytes, using the leakage on the S-box shared input/output byte, or on all S-box shared bits.

	pAK		pSB		SB bits	
Byte	PI	Traces	PI	Traces	PI	Traces
0	8.55×10^{-4}	3510	1.13×10^{-6}	2 663 127	1.63×10^{-4}	18 369
1	4.20×10^{-3}	714	5.26×10^{-7}	5 703 917	3.25×10^{-4}	9246
2	3.14×10^{-4}	9577	1.97×10^{-7}	15 239 700	9.42×10^{-5}	31 884
3	7.25×10^{-4}	4144	1.44×10^{-6}	2 085 394	1.83×10^{-4}	16 390
4	7.47×10^{-4}	4022	1.17×10^{-6}	2 567 874	1.59×10^{-4}	18 908
5	3.91×10^{-4}	7688	3.28×10^{-7}	9 143 616	3.99×10^{-5}	75 267
6	2.30×10^{-3}	1308	1.58×10^{-7}	18 974 045	1.80×10^{-4}	16 688
7	7.77×10^{-4}	3865	1.21×10^{-6}	2 479 407	1.81×10^{-4}	16 608
8	8.48×10^{-4}	3540	1.22×10^{-6}	2 459 585	1.48×10^{-4}	20 302
9	3.80×10^{-4}	7912	4.70×10^{-7}	6 383 554	4.35×10^{-5}	69 046
10	5.37×10^{-4}	5595	1.37×10^{-7}	21 961 257	9.74×10^{-5}	30 824
11	5.71×10^{-3}	526	7.07×10^{-7}	4 248 406	4.03×10^{-4}	7456
12	7.54×10^{-3}	398	1.12×10^{-6}	2 670 956	4.03×10^{-4}	7451
13	4.80×10^{-4}	6255	3.94×10^{-7}	7 615 405	4.98×10^{-5}	60 321
14	3.56×10^{-4}	8429	1.03×10^{-7}	29 109 285	9.74×10^{-5}	30 843
15	1.18×10^{-3}	2539	1.47×10^{-6}	2 044 868	1.89×10^{-4}	15 889

Table 5: Bound on PI and estimation of required number of attack traces for 50 % recovery success rate for each of the A7_d2 rows (after the first round ShiftRows), using the leakage on the shared MixColumns output bytes (pMC).

Key Bytes	PI	Traces
(0, 5, 10, 15)	9.29×10^{-4}	16 143
(4, 9, 14, 3)	7.15×10^{-4}	20 981
(8, 13, 2, 7)	1.65×10^{-4}	90 827
(12, 1, 6, 11)	2.99×10^{-5}	502 305

Table 6: Bound on PI and estimation of required number of attack traces for 50 % recovery success rate for each of the S6_d2 rows (after the first round ShiftRows), using the leakage on the shared MixColumns output bytes (pMC).

Key Bytes	PI	Traces
(0, 5, 10, 15)	6.83×10^{-4}	21 976
(4, 9, 14, 3)	2.36×10^{-4}	63 615
(8, 13, 2, 7)	4.72×10^{-4}	31 796
(12, 1, 6, 11)	5.56×10^{-4}	26 979

Our first series attack paths are based on the S-box value leakages, shown in Subsection 6.3 and Table 6.3. We can see that the two most interesting paths are the S-box input and the S-box internal bits, while the S-box output does not leak much. This can be explained by the architecture: the output S-box is not stored in any register and is instead directly routed to MixColumns. Quantitatively, it seems that it should be possible to recover most bytes of A7_d2 using about 10^5 traces, while the much higher leakage for S6_d2 gives a bound closer to 5000 traces.

Another attack path is to exploit the outputs of MixColumns (which depend on 32 bits of the key, therefore enumeration is still practical) as shown in Table 6.3 and Table 6.3. This attack path is very promising for A7_d2, interesting reduction in number of required traces, except for the last column (which is the easiest to recover using pAK leakage).

Finally, let us remark that we discussed only the attack paths that are the most obvious and probably easiest to exploit. Other paths exists, such as exploiting the transition leakage (which is a bit harder for exploitation, due to dependency on larger key chunks), or the key schedule which can be more leaky than the state (e.g., up to 6×10^{-3} bit per byte for A7_d2) at the cost of not being in a differential attack setting (i.e., the leakage does not depend on the plaintext).

7 Attacks

During the CHES 2023 challenge, two main attacks have been introduced against the SMAesH dataset. In this section, we present a brief overview of these attacks and compare them to the bounds of the previous section.

Bit-level templates This first attack [Cri23] targets the intermediate variables $t_{i,j} \in \mathbb{F}_2$ in the computation of the Boyar-Peralta S-box. Each such variable can be written as $t_{i,j} = f_j(k_i \oplus p_i)$, where $i \in \{0, \ldots, 15\}$ identifies the S-box, and $j \in \{1, \ldots, 258\}$ identifies the intermediate bit. In SMAesH, all these bits are masked: $t_{i,j} = t^0_{i,j} \oplus t^1_{i,j}$ such that only the shares $t^0_{i,j}$ and $t^1_{i,j}$ are computed in the circuit. At a high level, the attack is similar to

the one described in [BCS21].

First, in the profiling stage, for each variable, the points of interest (POIs)[4] are selected as the points with the highest SNR for that variable. A linear disctiminant analysis (LDA) model is then built from these POIs: if l is a leakage trace and $l_{i,j,k}$ is its restriction to the POIs for $t^k_{i,j}$, then the LDA model is trained to approximate the probabilities $P^k_{i,j}(0) = \widehat{\Pr}[t^k_{i,j} = 0 \mid l_{i,j,k}]$ and $P^k_{i,j}(1) = \widehat{\Pr}[t^k_{i,j} = 1 \mid l_{i,j,k}]$.

Second, at the attack stage, we apply the LDA models for every trace l in the attack dataset, giving $P^k_{i,j}$. We recombine the shares by computing the estimation distribution of $t_{i,j}$, using the equation $t_{i,j} = t^0_{i,j} \oplus t^1_{i,j}$: $P_{i,j}(0) = \widehat{\Pr}[t_{i,j} = 0] = P^0_{i,j}(0)P^1_{i,j}(0) + P^0_{i,j}(1)P^1_{i,j}(1)$ and $P_{i,j}(1) = \widehat{\Pr}[t_{i,j} = 1] = 1 - P_{i,j}(0)$. Then, for every key byte $i = 0, \dots, 15$, we compute the key likelihood $P_i(k_i) = P_{i,j}(f_j(k_i \oplus p_i))$ for $k_i = 0, \dots, 255$, where the plaintext p_i is known ($P_i(k_i)$ should be normalized to 1 to be a proper distribution). Finally, multiplying $P_i(k_i)$ across all traces in the attack dataset gives the finaly key byte distribution.

Given the 16 key byte distributions, we assume that the adversary can enumerate the keys from the most likely to the least likely. Using an key rank estimation algorithm, the number of keys to enumerate can be efficiently computed, showing that this attack succeeds to bring the rank of the key below 2^{68} using 290 000 traces on the `A7_d2` dataset[5] [CMS23]. This number is only a few time higher ($\approx \times 3$) than the bounds of Subsection 6.3 for the S-box bit attack path, which shows that this attack is quite efficient and that the bounds are reasonably tight.

Byte-level transitions with deep-learning The second attack [Mar23] is based on a multi-task deep learning model [MO24] and exploits leakage on the S-box input value, as well as transition leakage on the S-box input.

More precisely, SMAesH [CC23] instantiates a column of four S-boxes, which sequentially processes all necessary the key and state S-box computations for each round. For the first round (the only one targeted by the attack), the S-box column is first used to compute the key schedule: its input data is $(k_{13}, k_{14}, k_{15}, k_{12})$. Then, in the next four clock cycles, the AES state is fed through the S-box after going through the ShiftRows permutation. Denoting $z_i = k_i \oplus p_i$ for $i = 0, \dots, 15$ (k_i and p_i are the key and plaintext bytes respectively), the input columns to the S-box are therefore $(z_0, z_5, z_{10}, z_{15})$, (z_4, z_9, z_{14}, z_3), (z_8, z_{13}, z_2, z_7) and finally $(z_{12}, z_1, z_6, z_{11})$. After these clock cycles, the input of the S-box column is kept to 0 until the next round.

The attack exploits this sequence of inputs by training soft classification models (i.e., models that output distributions for the target value) to recover transitions on S-box inputs: between the round key and the first state column $(k_{13} \oplus z_0, k_{14} \oplus z_5, \dots)$ and between the consecutive state columns

$$(z_0 \oplus z_4, \dots, z_{15} \oplus z_3)\,, (z_4 \oplus z_8, \dots, z_3 \oplus z_7)\,, (z_8 \oplus z_{12}, \dots, z_7 \oplus z_{11})\,.$$

Finally, the value of the last column is also modelled (which is equivalent to the transition with 0, the next input value): $(z_{12}, \dots, z_{11})$. Unlike the previous attack, these models target the unmasked value which makes it possible to run the profiling without knowing the randomness used for the shares: only the key and plaintext are needed.[6] This restriction however makes the training more difficult: given e.g. the transition between sharings (z^0_0, z^1_0) and (z^0_4, z^1_4), a common assumption (Hamming Distance leakage [BCO04]) is that the leakage relates to $z^0_0 \oplus z^0_4$ and $z^1_0 \oplus z^1_4$. In order to recover the target transition leakage

[4] Between 1 and 645 POIs for each variable, with 107 POIs on average.

[5] For the sake of the CHES challenge, this was considered as the threshold of feasible enumeration, based on the hash rate of the Bitcoin network.

[6] That was a restriction of the CHES 2023 challenge for the `S6_d2` dataset, however the full SMAesH dataset contains the masking randomness for both `A7_d2` and `S6_d2`.

$z_0 \oplus z_4$, the model therefore has to infer the value of $(z_0^0 \oplus z_4^0) \oplus (z_0^1 \oplus z_4^1)$, which is similar to breaking first-order masking, a non-trivial task in deep-learning [MCLS23].

Using these models, the attack processes the columns iteratively. First, the last column $(z_{12}, z_1, z_6, z_{11})$ is attacked: from the distribution inferred by the model on these variable for all traces in the attack dataset, and from the knowledge of the plaintext, the values of k_{12}, k_1, k_6 and k_{11} can be recovered. Then, the key corresponding to third column of the state $(z_8, \ldots, z_7)$ is recovered in a similar manner, from knowledge of key for the last column. For example, for the first byte of the column, the model gets a distribution for $z_8 \oplus z_{12}$. Since p_8, p_{12} as well as k_{12} are known, a distribution for k_8 can be inferred. The attack proceeds similarly for the second column $(z_4, \ldots, z_3)$, except that the key of the third column is not assumed to be exactly known anymore, and instead the distribution of those key bytes (obtained at the previous step) are used in the computation. Finally, for the first column $(z_0, \ldots, z_{15})$, the attacks proceeds similarly to the second column, exploiting the transition leakage with the second column. It however also builds another estimator for those key bytes, using the transition with the leakage from the previous computation (round key update): distributions for the key bytes of this column can be built from the models for the leakage of $k_{13} \oplus z_0$, $k_{14} \oplus z_5$ and $k_{12} \oplus z_{15}$ and the knowledge of (the distributions of) the k_i from the other columns. The two estimators for the first column (transitions with the second column and with the round key update) are merged by multiplying distributions.

Finally, this attack produces a distribution for all key bytes, and key enumeration can be applied similarly to the other attack, resulting in a key rank below 2^{68} using 901 120 traces on the `S6_d2` dataset [CMS23]. Unlike the previous attack, this attack is not at all tight with the bounds of Subsection 6.3 (which, for pAK byte transitions on S6_d2 gives number of attack traces in the range 50-5000). This gap can be explained in part by the "black-box" modeling and by the more complex key recovery process due to dependencies between columns.

8 Conclusion

We propose the SMAesH dataset as a challenge and benchmark for the research in side-channel attacks and evaluation methodologies. This dataset comes with many useful and interesting features: an AES implementation with first-order masking, public randomness, well-aligned traces, random-key and fixed-key traces, measurements of the same HDL source on two distinct FPGA platforms, a mixed serial-parallel architecture, etc. In this paper, we perform a baseline leakage evaluation of the dataset, which indicates multiple possible attack paths. We quantify the amount of leakage which, using quantitative information metrics, allows us to bound the number of traces needed for an attack. Even tough these numbers vary greatly with the targeted part of the key and with the attack path, it seems that attacks below 10^5 traces should be feasible, and attacks below 10^4 traces will be quite challenging for A7_d2, while an attack under 10^3 traces may be feasible against the more leaky S6_d2. The initial attacks against the dataset use respectively 2.9×10^5 and 9.0×10^5 traces, however they were performed under the constraints of the CHES 2023 challenge and they can probably be further optimized.

Since the SMAesH IP is open-source, it is possible to perform in-depth investigation of the links between the observed leakage and the architecture. This is interesting not only for exploring attack paths or shortcut evaluation methodologies, but also for IP designers. Indeed, SMAesH is an overall simple design with masking as the sole countermeasure. Despite this simplicity, it appears to still be able to protect some of its computations, while others are much more leaky. We discussed how some of these differences can be explained by architectural features, which hints at possible solutions to make all values uniformly leaky. These observations could also be used to develop pre-silicon validation

tools based on worst-case but quantitative leakage assumptions.

References

[BBB+20] Davide Bellizia, Francesco Berti, Olivier Bronchain, Gaëtan Cassiers, Sébastien Duval, Chun Guo, Gregor Leander, Gaëtan Leurent, Itamar Levi, Charles Momin, Olivier Pereira, Thomas Peters, François-Xavier Standaert, Balazs Udvarhelyi, and Friedrich Wiemer. Spook: Sponge-based leakage-resistant authenticated encryption with a masked tweakable block cipher. *IACR Trans. Symmetric Cryptol.*, 2020(S1):295–349, 2020. URL: https://doi.org/10.13154/tosc.v2020.iS1.295-349, doi:10.13154/TOSC.V2020.IS1.295-349.

[BBC+20] Davide Bellizia, Olivier Bronchain, Gaëtan Cassiers, Vincent Grosso, Chun Guo, Charles Momin, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Mode-level vs. implementation-level physical security in symmetric cryptography - A practical guide through the leakage-resistance jungle. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part I*, volume 12170 of *Lecture Notes in Computer Science*, pages 369–400. Springer, 2020. doi:10.1007/978-3-030-56784-2_13.

[BBC+21] Davide Bellizia, Olivier Bronchain, Gaëtan Cassiers, Charles Momin, François-Xavier Standaert, and Balazs Udvarhelyi. Spook SCA CTF, 2021. doi:10.14428/DVN/W2SV5G.

[BCG+23] Julien Béguinot, Wei Cheng, Sylvain Guilley, Yi Liu, Loïc Masure, Olivier Rioul, and François-Xavier Standaert. Removing the field size loss from duc et al.'s conjectured bound for masked encodings. In Elif Bilge Kavun and Michael Pehl, editors, *Constructive Side-Channel Analysis and Secure Design - 14th International Workshop, COSADE 2023, Munich, Germany, April 3-4, 2023, Proceedings*, volume 13979 of *Lecture Notes in Computer Science*, pages 86–104. Springer, 2023. doi:10.1007/978-3-031-29497-6_5.

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2004. doi:10.1007/978-3-540-28632-5_2.

[BCS21] Olivier Bronchain, Gaëtan Cassiers, and François-Xavier Standaert. Give me 5 minutes: Attacking ASCAD with a single side-channel trace. *IACR Cryptol. ePrint Arch.*, page 817, 2021. URL: https://eprint.iacr.org/2021/817.

[BHM+19] Olivier Bronchain, Julien M. Hendrickx, Clément Massart, Alex Olshevsky, and François-Xavier Standaert. Leakage certification revisited: Bounding model errors in side-channel security evaluations. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part I*, volume 11692 of *Lecture Notes in Computer Science*, pages 713–737. Springer, 2019. doi:10.1007/978-3-030-26948-7_25.

[BJP20] Shivam Bhasin, Dirmanto Jap, and Stjepan Picek. AES HD dataset - 500 000 traces. AISyLab repository, 2020. https://github.com/AISyLab/AES_HD_2.

[BP12] Joan Boyar and René Peralta. A small depth-16 circuit for the AES s-box. In Dimitris Gritzalis, Steven Furnell, and Marianthi Theoharidou, editors, *Information Security and Privacy Research - 27th IFIP TC 11 Information Security and Privacy Conference, SEC 2012, Heraklion, Crete, Greece, June 4-6, 2012. Proceedings*, volume 376 of *IFIP Advances in Information and Communication Technology*, pages 287–298. Springer, 2012. doi:10.1007/978-3-642-30436-1_24.

[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. URL: https://doi.org/10.1007/s13389-019-00220-8, doi:10.1007/S13389-019-00220-8.

[BUS21] Davide Bellizia, Balazs Udvarhelyi, and François-Xavier Standaert. Towards a better understanding of side-channel analysis measurements setups. In Vincent Grosso and Thomas Pöppelmann, editors, *Smart Card Research and Advanced Applications - 20th International Conference, CARDIS 2021, Lübeck, Germany, November 11-12, 2021, Revised Selected Papers*, volume 13173 of *Lecture Notes in Computer Science*, pages 64–79. Springer, 2021. doi:10.1007/978-3-030-97348-3_4.

[CB23] Gaëtan Cassiers and Olivier Bronchain. Scalib: A side-channel analysis library. *J. Open Source Softw.*, 8(86):5196, 2023. URL: https://doi.org/10.21105/joss.05196, doi:10.21105/JOSS.05196.

[CC23] Momin Charles and Gaëtan Cassiers. SMAesH v1.0.0, 2023. doi:10.5281/zenodo.8359650.

[CDSU23] Gaëtan Cassiers, Henri Devillez, François-Xavier Standaert, and Balazs Udvarhelyi. Efficient regression-based linear discriminant analysis for side-channel security evaluations towards analytical attacks against 32-bit implementations. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(3):270–293, 2023. URL: https://doi.org/10.46586/tches.v2023.i3.270-293, doi:10.46586/TCHES.V2023.I3.270-293.

[CGLS21] Gaëtan Cassiers, Benjamin Grégoire, Itamar Levi, and François-Xavier Standaert. Hardware private circuits: From trivial composition to full verification. *IEEE Trans. Computers*, 70(10):1677–1690, 2021.

[CM23] Gaëtan Cassiers and Charles Momin. Power leakage traces of SMAesH, October 2023. doi:10.3217/bk4fx-rbh46.

[CMM+24] Gaëtan Cassiers, Loïc Masure, Charles Momin, Thorben Moos, Amir Moradi, and François-Xavier Standaert. Randomness generation for secure hardware masking - unrolled trivium to the rescue. *IACR Commun. Cryptol.*, 1(2):4, 2024. URL: https://doi.org/10.62056/akdkp2fgx, doi:10.62056/AKDKP2FGX.

[CMS23] Gaëtan Cassiers, Charles Momin, and François-Xavier Standaert. SMAesH challenge, 2023. URL: https://smaesh-challenge.simple-crypto.org/.

[Cri23] Valence Cristiani. Donatella, 2023. URL: https://github.com/simple-crypto/SMAesH-challenge-submissions/blob/main/Donatellla.zip.

[CRR02] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28. Springer, 2002. doi:10.1007/3-540-36400-5_3.

[dCGRP19] Eloi de Chérisey, Sylvain Guilley, Olivier Rioul, and Pablo Piantanida. Best information is most successful mutual information and success rate in side-channel analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(2):49–79, 2019. URL: https://doi.org/10.13154/tches.v2019.i2.49-79, doi:10.13154/TCHES.V2019.I2.49-79.

[Fei14] Yunsi Fei. Northeastern university TeSCASE dataset, 2014. URL: https://chest.coe.neu.edu/?current_page=POWER_TRACE_LINK&software=ptmasked.

[HKW11] Heike Hofmann, Karen Kafadar, and Hadley Wickham. Letter-value plots: Boxplots for large data. Technical report, had.co.nz, 2011.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999. doi:10.1007/3-540-48405-1_25.

[Man04] Stefan Mangard. Hardware countermeasures against DPA ? A statistical analysis of their effectiveness. In Tatsuaki Okamoto, editor, *Topics in Cryptology - CT-RSA 2004, The Cryptographers' Track at the RSA Conference 2004, San Francisco, CA, USA, February 23-27, 2004, Proceedings*, volume 2964 of *Lecture Notes in Computer Science*, pages 222–235. Springer, 2004. doi:10.1007/978-3-540-24660-2_18.

[Mar23] Thomas Marquet. Morningstar-xxx, 2023. URL: https://github.com/simple-crypto/SMAesH-challenge-submissions/blob/main/Morningstar-xxx.zip.

[MCHS23] Loïc Masure, Gaëtan Cassiers, Julien M. Hendrickx, and François-Xavier Standaert. Information bounds and convergence rates for side-channel security evaluators. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(3):522–569, 2023. URL: https://doi.org/10.46586/tches.v2023.i3.522-569, doi:10.46586/TCHES.V2023.I3.522-569.

[MCLS23] Loïc Masure, Valence Cristiani, Maxime Lecomte, and François-Xavier Standaert. Don't learn what you already know scheme-aware modeling for profiling side-channel analysis against masking. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(1):32–59, 2023. URL: https://doi.org/10.46586/tches.v2023.i1.32-59, doi:10.46586/TCHES.V2023.I1.32-59.

[MO24] Thomas Marquet and Elisabeth Oswald. Exploring multi-task learning in the context of masked AES implementations. In Romain Wacquez and Naofumi Homma, editors, *Constructive Side-Channel Analysis and Secure Design - 15th International Workshop, COSADE 2024, Gardanne, France, April 9-10, 2024, Proceedings*, volume 14595 of *Lecture Notes in Computer Science*, pages 93–112. Springer, 2024. doi:10.1007/978-3-031-57543-3_6.

[MS21] Loïc Masure and Rémi Strullu. Side channel analysis against the anssi's protected AES implementation on ARM. *IACR Cryptol. ePrint Arch.*, page 592, 2021. URL: https://eprint.iacr.org/2021/592.

[PPM+23] Stjepan Picek, Guilherme Perin, Luca Mariot, Lichao Wu, and Lejla Batina. Sok: Deep learning-based physical side-channel analysis. *ACM Comput. Surv.*, 55(11):227:1–227:35, 2023. doi:10.1145/3569577.

[RSV+11] Mathieu Renauld, François-Xavier Standaert, Nicolas Veyrat-Charvillon, Dina Kamel, and Denis Flandre. A formal study of power variability issues and side-channel attacks for nanoscale devices. In Kenneth G. Paterson, editor, *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, volume 6632 of *Lecture Notes in Computer Science*, pages 109–128. Springer, 2011. doi:10.1007/978-3-642-20465-4_8.

[SA08] François-Xavier Standaert and Cédric Archambeau. Using subspace-based template attacks to compare and combine power and electromagnetic information leakages. In Elisabeth Oswald and Pankaj Rohatgi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings*, volume 5154 of *Lecture Notes in Computer Science*, pages 411–425. Springer, 2008. doi:10.1007/978-3-540-85053-3_26.

[WHJ+21] Yoo-Seung Won, Xiaolu Hou, Dirmanto Jap, Jakub Breier, and Shivam Bhasin. Back to the basics: Seamless integration of side-channel pre-processing in deep neural networks. *IEEE Trans. Inf. Forensics Secur.*, 16:3215–3227, 2021. doi:10.1109/TIFS.2021.3076928.

Additional figures

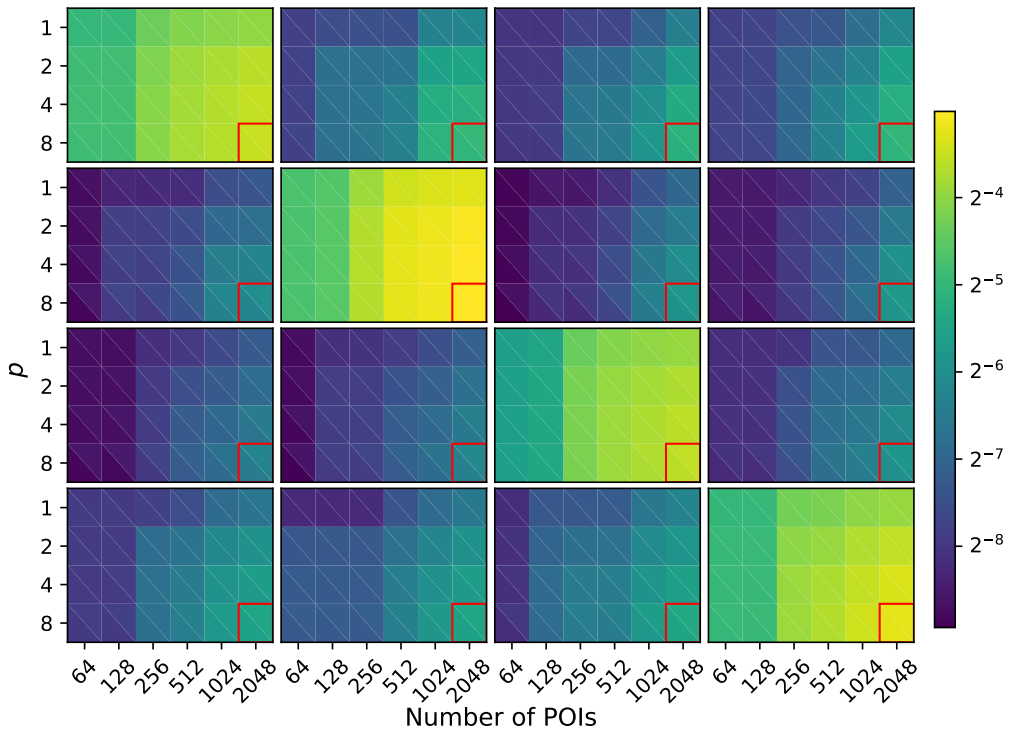

Figure 14: Parameter selection for the first share of pAK (XOR-)transition on S6_d2

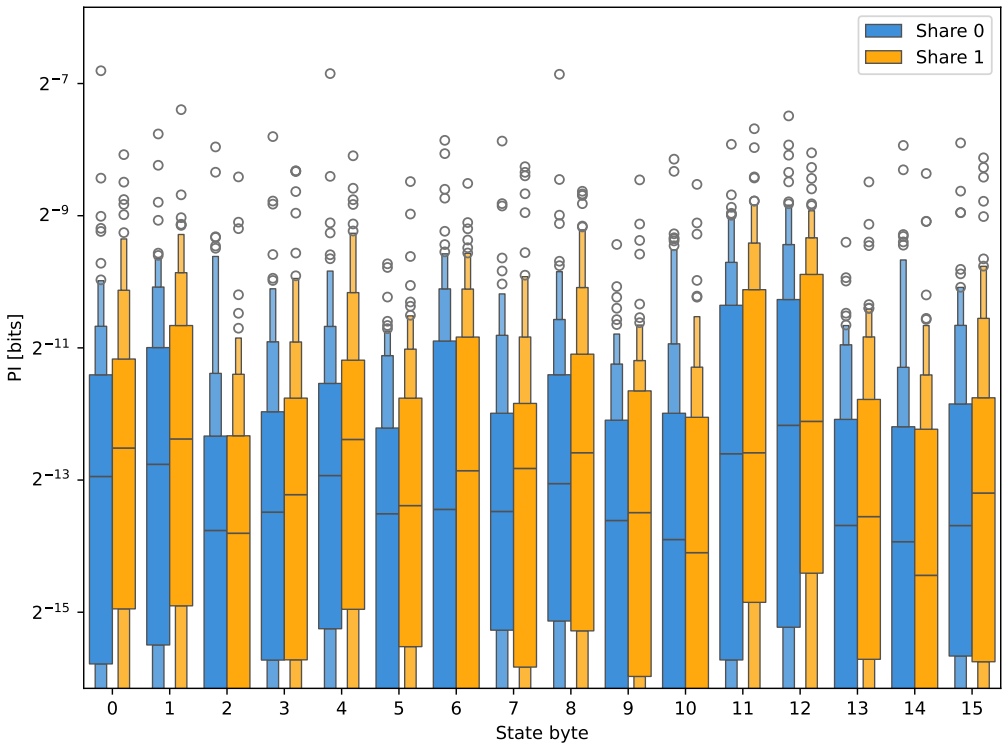

Figure 15: Information distribution for the shares of the sboxes variables Sboxes considering the model maximising the PI on S6_d2.

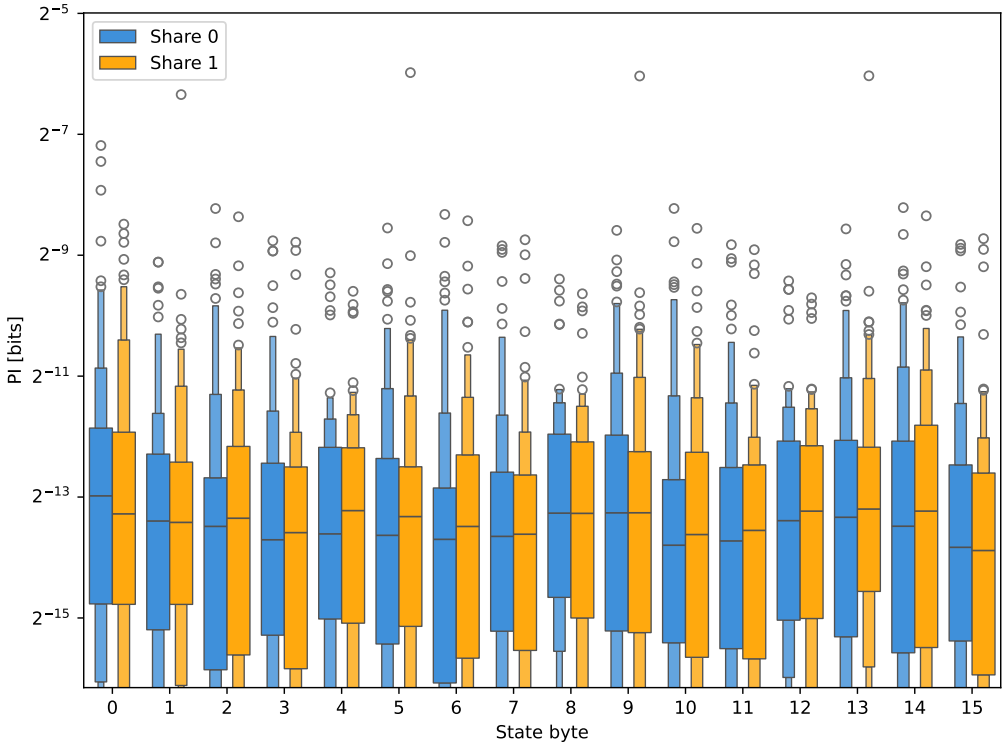

Figure 16: Information distribution for the (XOR-) transition of shares of the sboxes variables Sboxes considering the model maximising the PI on A7_d2.

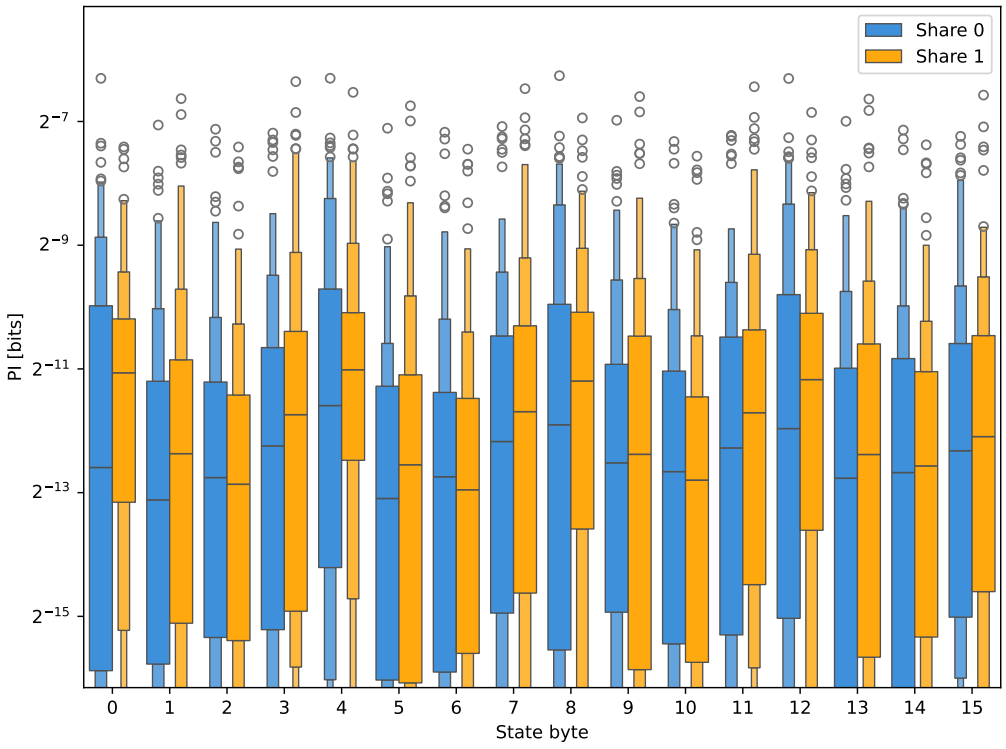

Figure 17: Information distribution for the (XOR-) transition of shares of the sboxes variables Sboxes considering the model maximising the PI on S6_d2.