Simple Photonic Emission Attack with Reduced Data Complexity

Elad Carmon[1], Jean-Pierre Seifert[2,3], and Avishai Wool[4(✉)]

[1] Tel-Aviv University, 69978 Tel-Aviv, Israel
eladca@gmail.com
[2] Security in Telecommunications, Technische Universität Berlin, Berlin, Germany
Jean-Pierre.Seifert@telekom.de
[3] FhG SIT, Darmstadt, Germany
[4] Tel-Aviv University, 69978 Tel-Aviv, Israel
yash@eng.tau.ac.il

Abstract. This work proposes substantial algorithmic enhancements to the SPEA of Schlösser et al. [15] by adding cryptographic post-processing, and improved signal processing to the photonic measurement phase. Our improved approach provides three crucial benefits: (1) For some SBox/SRAM configurations the original SPEA method is unable to identify a unique key, and terminates with up to 2^{48} key candidates; using our new solver we are able to find the correct key regardless of the respective SBox/SRAM configuration. (2) Our methods reduce the number of required (complex photonic) measurements by an order of magnitude, thereby shortening the duration of the attack significantly. (3) Due to the unavailability of the attack equipment of Schlösser et al. [15] we additionally developed a novel Photonic Emission Simulator which we matched against the real equipment of the original SPEA work. With this simulator we were able to verify our enhanced SPEA by a full AES recovery which uses only a small number of photonic measurements.

1 Introduction

1.1 Background

While the phenomena of photonic emission from switching transistors in silicon is actually a very old one, cf. [5,12], the role of photons in cryptography as a practical side channel source has just recently emerged as a novel research direction, cf. [3,9,10,15,16]. Thus, it is important to include photonic side channels in future hardware evaluations of security ICs.

However, so far only the first steps within this direction have been successfully achieved: The work of [3,9,10,15,16], showed that the required equipment to carry out successful SPEA or DPEA against real world ICs is comparable in price to that of normal Power Analysis equipment.

A. Wool—Supported in part by a grant from the Interdisciplinary Cyber-Research Center at Tel Aviv University.

F.-X. Standaert and E. Oswald (Eds.): COSADE 2016, LNCS 9689, pp. 35–51, 2016.
DOI: 10.1007/978-3-319-43283-0_3

This is where the current paper fits in and continues the current state of the art in a better understanding of the Photonic Side Channel. It takes the next step by precisely characterizing a very low number of selected plaintexts as required for the respective photonic measurements and also relating the resulting measurements in terms of their SNR to the eventual workload of the final cryptographic key reconstruction phase.

1.2 Related Work

Photonic emission in silicon is a known physical phenomena which has been studied since the 1950s [12]. Specifically in the failure analysis community, hot-carrier luminescence has primarily been used to characterize implementation and manufacturing faults and defects [7,17]. Here, the technologies of choice to perform backside analysis are PICA (Picosecond Imaging Circuit Analysis) [1] and SSPDs (Superconducting Single Photon Detectors) [18]. Both technologies are able to capture photonic emissions with high performance in their respective field, but carry the downside of immense cost and complexity.

One of the first uses of photonic emissions in CMOS in a cryptographic application was presented in 2008 [8]. However, the authors increased the voltage supply to 7 V operating voltage, which is above the chips maximum limit for voltage. The authors utilize PICA to spatially recover information about binary additions related to the AddRoundKey operation of AES running on a 0.8 µm PIC16F84A microcontroller. As the authors state, such a PICA device "is available in several laboratories, for example, in the French space agency CNES". Employing PICA in this manner led to enormous acquisition times. This is especially true considering the size of the executed code. It took the authors 12 h to recover a *single* potential key byte [8]. In 2011, an integrated PICA system and laser stimulation techniques were used to attack a DES implementation on an FPGA [6]. The authors proved that the optical side channel might be used for differential analysis. However, the analysis strongly relied on a specific implementation of DES in which registers were always zeroed before their use. The results required a differential analysis and a full key recovery was also not presented. As the authors note, the use of equipment valued at more than 2,000,000 Euros does not make such an analysis particularly relevant.

Nevertheless, recently, a real breakthrough was achieved by [15,16]. This work presented a novel low-cost optoelectronic setup for time - and spatially resolved analysis of photonic emissions. The authors also introduced a corresponding methodology, named Simple Photonic Emission Analysis. They successfully performed such analysis of an AES implementation and were able to recover AES-128 keys by monitoring memory accesses. This work was also extended to AES-192 and AES-256 [16]. The same research group also introduced Differential Photonic Emission Analysis and presented a respective attack against AES-128 [10]. They successfully revealed the entire secret key with their DPEA. In 2015 Bertoni et al. [3] offered an improved Simple Photonic Emission Analysis, monitoring a different section of the SRAM logic. However, they assumed a specific SRAM structure which contains only single byte in every row. Their simulations

do not model the physical environment but rather an ideal model in which the value of every bit can be identified. They also described an attack of masked AES, however the attack is unrealistic since it assumes monitoring the photonic emission of a single experiment.

A side channel analysis using memory access patterns is reminiscent of the field of cache attacks. For instance, the first "real world" cache-based chosen plaintext attacks on AES were carried on OpenSSL implementations [2,13].

1.3 Contributions

In this work we enhance the original SPEA of Schlösser et al. [15] by adding cryptographic post-processing and an improved signal processing to the measurements phase. We call the resulting attack Enhanced SPEA, or E-SPEA for short.

Our first contribution is to record the photonic side-channel leaks from the first *two* AES rounds, covering 32 SBox activations. We show that these leakages embed enough constraints to allow the identification of the complete key, regardless of the placement of the SBox array in SRAM. This is in contrast to the original SPEA, which terminates with up to 2^{48} key candidates for certain SRAM configurations. Furthermore, taking advantage of the slow diffusion properties in the first AES round, we are able to mount this attack very efficiently, with a time complexity of 2^{20}. Our optimized cryptographic solver finds the correct key within minutes on a standard PC.

Next, we devise a strategy for choosing optimal plaintexts, that causes the photonic side-channel to produce constraints (specific SRAM accesses) which enable our solver to work very quickly for all SRAM configurations. We collect the necessary constraints with only 32 plaintexts, instead of the 256 plaintexts required by Schlösser et al. [15].

Moreover, we developed a special signal-processing decoder that automatically calibrates certain internal thresholds—relying on our chosen plaintext strategy. The decoder works even when the SNR is low, adjusting its thresholds differently to match the requirements of the cryptographic solver. To do so, the decoder uses a different (auto-calibrated) threshold for each AES round. Using the combination of our carefully crafted decoder and solver, we can trade off the number of measurements against the solver's running time: fewer measurements (i.e., a lower SNR) cause a longer running time—but without missing the correct key.

The combination of the above contributions provides two main benefits.

1. We are always able to quickly find the correct key, regardless of the SRAM configuration.
2. Our methods reduce the number of required *optical* measurements dramatically by an order of magnitude, and thus we are able to shorten the duration of the attack significantly.

Also, in order to validate our attacks we built a Monte-Carlo simulator of the underlying physics of the photonic emissions, with a noise model which incorporates

– internal noise within the detector,
– external noise from nearby transistors, and
– other effects.

We validated our simulator against the results as reported in Schlösser [14]. Our simulator can be used to explore alternative lab setups, taking into account various critical parameters such as the lens area, height above the chip, supply voltage, ambient temperature, and equipment sensitivity.

We also believe that our photonic emission simulator is of independent interest and is of great value for the research community lacking (so far) the optical equipment as described within Schlösser [14].

Organization. The organization of the present paper is as follows. Section 2 introduces the SPEA on AES. Section 3 describes our cryptographic solver. Section 4 explains our choice of plaintexts. Section 5 describes the Auto-calibrating decoder. Section 6 describes our performance evaluation, and we conclude in Sect. 7. The description of the photonic emission simulator can be found in the full technical report [4].

2 The Photonic Side Channel in AES

2.1 The SRAM and Its Use in AES

SRAM is a common type of volatile memory found in many ICs. The SRAM is built from memory cells arranged in rows and columns, and every memory cell can be approached using a row/column access logic. In particular, the access logic for each SRAM row includes a so called row-access transistor, which is activated whenever the IC needs to access any cell in that SRAM row. Due to to this functionality, i.e., enabling an entire row, the respective row-access transistor is very strong. This means that the photonic emission of this transistor is by magnitudes larger than the individual SRAM cells by itself. For a thorough introduction into SRAM and its physical implementation details we refer the reader to [19].

The number of bytes in an SRAM row depends on the underlying SRAM architecture. In [15] the authors found that on an AT-Mega328P a single SRAM row consists of 8 bytes, whereas an ATXMega128A1 stores 16 bytes in an entire row. Figure 1(a) shows a photo of the SRAM, with a row width of 8 bytes.

A central component of the AES cipher is the SBox. This is an array of 256 bytes which is most often implemented as a lookup table. In each AES round the algorithm performs 16 SBox lookups. In many ICs implementing AES in software the entire SBox array is placed in SRAM.

In this paper we will denote the SRAM row width by ω. In general the SBox starts at an *offset* within an SRAM row, $0 \leq \text{offset} \leq \omega - 1$, and occupies

$L = \lceil 256/\omega \rceil$ rows (see Fig. 1(b)). When $\omega = 8$, depending on the offset, we have $L = 32$ or $L = 33$. As we shall see, the value of the offset has an impact on the SPEA.

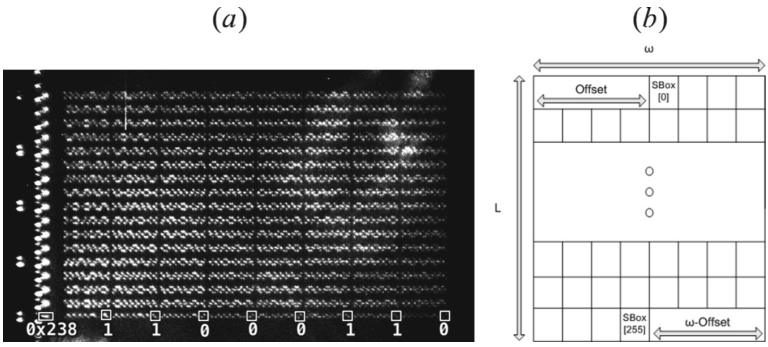


Fig. 1. The SRAM memory in (a) captured with a CCD by the courtesy of [15]. The row-access transistors appear to the left of the SRAM cells. In (b), a schematic of the SRAM section containing the SBox in L rows, ω cells per row and starting at some offset value.

2.2 Simple Photonic Emission Analysis (SPEA)

Monitoring the access patterns to the SRAM rows allows the SPEA as presented in [15]. Towards this goal, [15] first used a simple CCD camera approach to initially map the respective IC's layout, locating the SRAM memory, and specifically, the memory rows containing the SBox array and the offset value, cf. [11]. Hereafter, they placed a NIR (Near Infra Red) photon detector offering time resolved measurements over the row access transistor of some SRAM row containing SBox values. We call the SBox row on which the detector is placed *the detectable row*, and denote its number by d ($1 \leq d \leq L$). The authors ran the AES algorithm M times (by actually resetting the IC M times), encrypting the same plaintext. Consider one of the 16 SBox activations of the first AES round for plaintext byte p_i and key byte k_i. If the detector identifies an activation for $\text{SBox}(p_i \oplus k_i)$, then there are ω options for $p_i \oplus k_i$ and since the plaintext is known, they have ω options for k_i.

Using all possible plaintext bytes $\{0, 1, \ldots, 255\}$ (M times each) they revealed sets of ω potential candidates for every byte of the key, then they analyzed each key byte separately, intersecting sets of candidates for every key byte reducing the number of potential candidates. The success of the SPEA method depends on two factors:

1. Using a large enough number of measurements M, providing a sufficient SNR.

2. The offset value. The SPEA works best when the offset is odd. In other cases its performance is limited, and in particular when offset = 0 the number of candidates for every key byte can't be reduced below ω candidates for each byte, resulting in ω^{16} key candidates.

3 The E-SPEA

Our attack depends on several ideas:

1. Use the lab setup of [15], with a NIR photon detector placed over the row access transistor of some row d in the SBox, to record the photonic emissions from the SBox activations in *2 full AES rounds* and use the dependence between rounds to identify the correct key.
2. Use a careful choice of plaintexts to quickly reduce the entropy.
3. A novel auto-thresholding method, based on the choice of plaintexts, lets us avoid the need to calibrate and lets us handle noise.

During the AES encryption process, there are ten rounds, each accessing SRAM memory to use the SBox array. In every round 16 bytes of the current state matrix are replaced by 16 bytes copied from the SRAM memory using the SBox as a lookup table.

Following [15] we place a detector over the location of the transistor controlling access to a row of SRAM containing ω cells of the SBox array. Thus each of the 16 SBox accesses per AES round has a $\approx 1/L$ probability that the row on which the detector is located ("the detectable row") will be accessed, assuming a random plaintext. Our attack requires knowing the offset value (recall Fig. 1(a)) and the row number (d) of the detectable row.

3.1 The Attack Structure

The attack activates the AES IC to encrypt plaintexts of the form $\{a, a, \ldots, a\}$ (all plaintext bytes are the same) for different values of a. For each key byte k_j, if the detectable row is accessed in the first AES round while looking up state byte j in the SBox, we obtain a constraint on the possible value of k_j, which reduces the number of possibilities for its value from 256 to ω. In [15] the authors iterated over all 256 plaintext options, guaranteeing that the detectable row is accessed at least once for every key byte in the first AES round (in Sect. 4 we show that we can achieve the same with much fewer plaintexts). Thus we obtain at most ω^{16} AES key candidates based only on constraints from round 1 one of which is the correct key. When $\omega = 8$ we get $\omega^{16} = 2^{48}$.

Now we can use the detected leakage from round 2 to identify the correct key and discard the false ones. For a fixed plaintext and a given key candidate, we can deterministically compute the 2^{nd} round key and the state at the end of round 1. We can then deduce the 16 SBox cells that are accessed in round 2 and compare them to the access pattern measured by the detector. The probability

of matching the detected pattern is $\omega^{16}/2^{128}$. Therefore, for the ω^{16} candidates from round 1, we can expect $\approx \omega^{32}/2^{128}$ candidates to fit the leakage from both rounds. For $\omega = 8$ we get $\approx 2^{96}/2^{128} \ll 1$, so it is very likely that we will find just the single correct key.

Note that the above process is a naive method used only to illustrate that the leakage from the first two AES rounds is sufficient to uniquely identify the correct key. However, we can do much better: We devised a specialized solver that has a time complexity of 2^{20} and space complexity of 2^{23} bits, when $\omega = 8$.

3.2 The Solver

Let a *partial key* be an array of 16 cells, each of which may contain either a value 0...255 or 'undefined'. The main algorithm maintains a set of partial key candidates, and works in stages. Each stage corresponds to a particular state byte, or a set of state bytes, in round 2: In the stage for state byte j the algorithm first grows the set of candidates, by extending each candidate partial key so all the key bytes that state byte j depends on are well defined. Then the algorithm rejects all the (extended) candidates that are inconsistent with 2^{nd} round leaks. A stage can correspond to several state bytes if the extended candidate keys are well defined for all the depended-upon key bytes of the stage. The pseudo-code for a single stage has the following structure:

```
//stage for state byte j
input: set prevCandidates
Let enumBytes(j) be the set of additional key bytes that state byte j depends on and
are still 'undefined' in all partial keys in prevCandidates.
1: for all C in prevCandidates do
2:    for all possible values V for key bytes in enumBytes(j) do
3:       if Consistent (j, C||V) then
4:          nextCandidates ← nextCandidates ∪ {C||V}
5:       end if
6:    end for
7: end for
8: prevCandidates ← nextCandidates
9: nextCandidates = ∅
```

We keep the results of the 2^{nd} round row activations in a data structure denoted by R2A: R2A$\{p^t\}$ is a vector of L bits such that $(\text{R2A}\{p^t\})_j = 1$ if plaintext p^t caused a detectable SBox access in round 2 on state byte j.

For a given partial key X and state byte $1 \le j \le 16$ line 3 calls a function to test whether X is consistent with the 2^{nd} round leaks for state byte j:

```
1: Consistent (j, X)
2: for all plaintexts p^t do
3:    v_jt ← RowLookupOf (j, X, p^t)
4:    if ((v_jt == d and (R2A{p^t})_j==0) or (v_jt != d and (R2A{p^t})_j==1)) then
5:       return FALSE //partial key X is inconsistent
6:    end if
7: end for
8: return TRUE //partial key X is consistent
```

The function RowLookupOf (j, X, p^t) at line 3 returns the SBox row that is looked up for state byte j with plaintext p^t and partial key X. We ensure that all the key bytes that state byte j depends on are well defined in X by a careful ordering of the enumeration (see below), that also ensures the algorithm's ability to disqualify partial keys early. The time complexity of Consistent (j, X) is clearly $O(N_p)$, where N_p is the number of plaintexts.

Table 1. The algorithm going over bytes of the second round state matrix column by column. For every stage of the solver the number of candidates increases due to the newly enumerated key bytes—but the number of remaining candidates after the stage is reduced due to the second round constraints. This analysis assumes one second round activation for each of the state matrix byte j, and $\omega = 8, L = 32$, thus each stage cuts down the number of candidates by a factor of $\sim 2^5$.

Stage	Column	State byte	Bytes enumerated	Candidates	Running complexity	Space (bits)
1	1	1	1, 6, 11, 14, 16	2^{15}	2^{18}	$5 \cdot 2^3 \cdot 2^{15}$
2	1	3	3	$2^{10} \cdot 2^3$	2^{16}	$6 \cdot 2^3 \cdot 2^{13}$
3	1	2	2, 15	$2^8 \cdot 2^6$	2^{17}	$8 \cdot 2^3 \cdot 2^{14}$
4	1	4	4, 13	$2^9 \cdot 2^6$	2^{18}	$10 \cdot 2^3 \cdot 2^{15}$
5	2	5, 6	5, 10	$2^{10} \cdot 2^6$	2^{19}	$12 \cdot 2^3 \cdot 2^{16}$
6	2	7	7	$2^6 \cdot 2^3$	2^{12}	$13 \cdot 2^3 \cdot 2^9$
7	2	8	8	$2^4 \cdot 2^3$	2^{10}	$14 \cdot 2^3 \cdot 2^7$
8	3	9, 10, 11	9	$2^2 \cdot 2^3$	2^8	$15 \cdot 2^3 \cdot 2^5$
9	3	12	12	$2^{-10} \cdot 2^3$	2^{-4}	$16 \cdot 2^3$
10	4	13, 14, 15, 16	-	2^{-27}	2^{-24}	$16 \cdot 2^3$

3.3 Selecting the Enumeration Order

According to the appendix, state byte 1 depends on key bytes 1, 6, 11, 16 after the round 1 MixColumns step, and byte 1 of round key 2 depends on key bytes 1, 14. Thus immediately before the SBox lookup of round 2, state byte 1 depends on 5 key bytes: 1, 6, 11, 14, 16 (see Fig. 2a). So in the solver's stage 1 we enumerate over a set of ω^5 candidates. Roughly speaking when a 2^{nd} round row activation is detected for state byte 1, the consistency check will reduce the set to about $\frac{\omega^5}{L} \approx 2^{10}$ candidates. In the same way we find that state byte 3 depends on key bytes 1, 3, 6, 11, 16—4 of which we've already enumerated in stage 1 (see Fig. 2b). So we only need to extend each candidate partial key by a single byte. Thus we enumerate on byte 3 for the second stage. After this stage (assuming 2^{nd} round activation for the corresponding state byte) the number of candidates becomes $\approx (\frac{\omega^5}{L}) \cdot \omega \cdot \frac{1}{L} = \frac{\omega^6}{L^2}$, which is 2^8 when $\omega = 8$.

Continuing in a similar manner, we find that state byte 2 depends on 6 key bytes: 1, 2, 6, 11, 15, 16 so we need to extend the partial keys by 2 bytes (2 and 15), ending the stage with $\frac{\omega^8}{L^3} = 2^9$, and so forth column by column. Table 1 illustrates the whole process. The figure shows that stage 5 dominates the time

complexity (of 2^{19}) and space complexity (of 2^{21}) for a total time complexity of $\approx 2^{20}$.

Note that the state bytes of the first column (state bytes 1–4) collectively depend on 10 key bytes. A simpler algorithm would have enumerated over all 10 bytes together. However, such an approach would have had a time complexity of $\omega^{10} = 2^{30}$ (for $\omega = 8$)—significantly worse than the time complexity of our stages 1–4 combined.

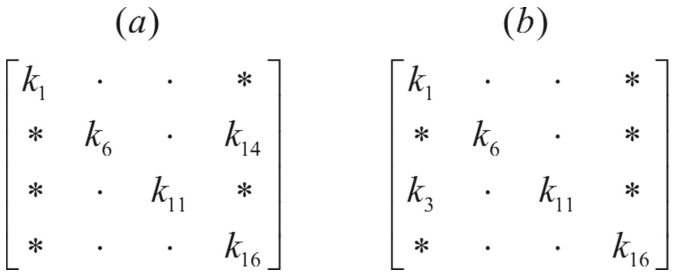


Fig. 2. The key bytes affecting the round 2 SBox accesses: (a) for state byte 1, (b) for state byte 3. Note that the key bytes on the diagonal $(1, 6, 11, 16)$ determine the state bytes of the 1^{st} column at the end of round 1, and the key bytes on the left and right columns determine the 2^{nd} round key.

4 Choosing the Plaintexts

As stated in Sect. 3 when a row access is detected in round 1, the number of key candidates for that byte is reduced to ω. The SBox values are located over L sequential rows of the SRAM memory, so the probability to observe a row access for randomly chosen plaintext is $\approx 1/L$.

For the set of plaintexts $p^t = (a_t, \ldots, a_t)$ we use, we want to have at least one detectable row access in round 1 for every key byte. This can of course be guaranteed by using all 256 plaintexts, as done by [15]. However we can achieve the same result with much fewer plaintexts. For a given offset (recall Fig. 1(a)), a plaintext byte a_t, and key byte k_j, the AES SubBytes step generates an SRAM row access to row l

$$l = \left\lfloor \frac{a_t \oplus k_j + \text{offset}}{\omega} \right\rfloor + 1 \tag{1}$$

We capitalize on this by using a "ω-jump" strategy for plaintext ordering. We choose the following plaintexts:

$$p^t = \{c + j \cdot \omega, \ldots, c + j \cdot \omega\} \tag{2}$$

for $c = \{0, \ldots, \omega - 1\}$, and $j = \{0, \ldots, L - 1\}$ for offset $= 0$ or $j = \{0, \ldots, L - 2\}$ for offset $\neq 0$. Essentially for every value of c this strategy holds

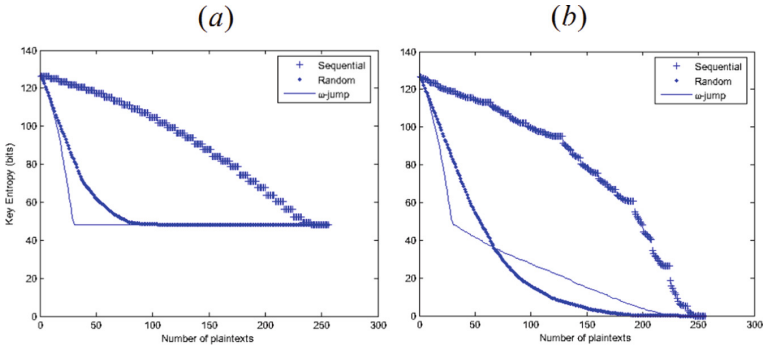


Fig. 3. The entropy of the key as function of the number of plaintexts, using only first round leakages for offset = 0 (a) and offset = 1 (b). The graphs show the sequential plaintext selection used in [15], a uniformly- random selection strategy and our "ω-jump" strategy. We can see that using only round-1 information, the entropy can't be reduced below 48 bit when offset = 0. We can see that using "ω-jump" the entropy decreases fast and using only 32 plaintexts we have a 48bit entropy, which is the "working point" of our solver, for all offsets.

the least-significant-bits fixed (e.g., the 3 LSBs for $\omega = 8$) and goes over all options for the MSBs.

By choosing some c and going over all options of j to multiply the row width ω we force a row access to all of the SRAM rows $\{1, 2, 3, \ldots, L\}$ for offset $= 0$ regardless of the key value k. If offset $\neq 0$, the "ω-jump" strategy causes a detectable row access for all the rows $\{2, 3, \ldots, L-1\}$ plus one more row access—to the first or the last row depending on the chosen value of c. After going over all the values of j we increment c and repeat. By setting the detectable row d to be $2 \leq d \leq L-1$ and using a set of L (or $L-1$) plaintexts of Eq. (2) we are guaranteed to have one detectable row activation for every key byte during the first AES round. Figure 3 shows the drop in key entropy as a function of the number of plaintexts. Figure 3(b) shows that for offset = 1 the random strategy of plaintexts selection reduces the entropy to 0 quicker than the "ω-jump" strategy, but using the "ω-jump" strategy the entropy reaches the desired working point of our solver (48 bit entropy) using only L carefully chosen plaintexts.

Note that unlike the first round, the second round row activations can't be controlled by the choice of plaintexts since the access pattern in round 2 also depends on the key diffusion caused by round 1.

5 Decoding the Photonic Traces with Auto Threshold Calibration

For each of the plaintexts p^t we activate the IC (or, in our case, the simulator) M times. For each activation we count the number of detected photons per time step, while the detector is fixed at SRAM row d. We summarize the detection

counts per time step, to obtain a "photonic trace" $T(p^t)$ for each plaintext, for the time duration of the first 2 AES rounds. Following [14,15] we assume an IC instruction cycle of 800 ns[1], a photonic trace spans 25.6 µs, represented by a vector of 1280 samples, one per 20 ns. For plaintext p^t we now need to decode the trace to extract two arrays of 16 bits: R1A and R2A recording the results of the 2 AES rounds' SBox activations. A bit value of 1 indicates that the plaintext caused a detectable SRAM access on the current SBox activation. A natural decoding rule is to use a threshold: if the number of detected activations during SBox access j in round 1 exceeds the threshold, we set $(R1A\{p^t\})_j = 1$, and 0 otherwise, and similarly for R2A.

A crucial task is calibrating the threshold so it can reliably distinguish between true detections and noise. Calibrating a threshold is often a heuristic trial-and-error process. However, since we choose the plaintexts in a specific way, we can calibrate the threshold automatically to its optimal value.

5.1 Calibration at High SNR

For illustration purposes we start by considering what happens when the SNR is high. Our method of choosing plaintexts guarantees a first round detectable row activation for every state byte j for at least one plaintext. Therefore we aggregate the N_p photonic traces (one per plaintext) by taking the *maximum* count per time step:

$$(maxT)_i = \max_{t=1\ldots N_p} \left\{(T(p^t))_i\right\} \tag{3}$$

for the time duration of AES round 1.

This max-trace should exhibit 16 distinct peaks, at the time-steps corresponding to the 16 SBox activations of AES round 1. If we sort maxT in descending order, we expect to see a clear drop between the 16^{th} peak value, and the 17^{th} (which is the highest peak caused by the noise). We can use this fact and choose our threshold to be the midpoint between the two peaks:

$$Threshold = \frac{peak_{16} + peak_{17}}{2} \tag{4}$$

where $peak_{16}$ and $peak_{17}$ are the 16th and 17th largest samples of $maxT$.

Even though the threshold is calibrated on $maxT$ for the first AES round, it is valid for every individual trace $T(p^t)$, and for both AES rounds. Thus we can use this threshold for all N_p traces to set the bit arrays $R1A\{p^t\}$ and $R2A\{p^t\}$.

However, we do not use this basic calibration. Instead, in the next section we show a more delicate calibration with two thresholds, that converge to the basic threshold when SNR is high.

[1] Note that this clock frequency is a slow 1.25 MHz. The AT-Mega328p can operate at faster clock frequencies, up to 20 MHz- we simulated the 1.25 MHz clock to allow a comparison of the simulated results with the findings of [14,15].

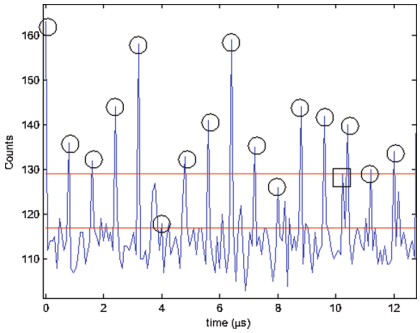


Fig. 4. A trace and the low and high thresholds for M = 1,000,000 (low SNR). In circles, peaks at expected time slots. In a box, a peak at an unexpected time slot. Thus, Thr_1 is set just below the lowest circled value, and Thr_2 is set just above the boxed value.

5.2 Calibration at Low SNR

When the number of measurements M for each plaintext is low, the SNR drops and the threshold calibration method of Sect. 5.1 starts to introduce decoding errors. We can define 2 error types:

1. False negative: a missed row activation (threshold was set too high).
2. False positive: an incorrect row activation (threshold was set too low).

We separate the discussion of the errors into two cases, for the first and second rounds of the AES process.

Recall that our solver (Sect. 3.2) uses the first AES round activations to reduce the number of candidates from 256 to ω for every key byte. When a false positive occurs during the first AES round we will have more than ω options for the key byte, since we will have ω options for each activation. This could make the solver running time slower and cause the set of final key candidates to be larger. However, when a false negative occurs during the first AES round, we are left with 256 options for this key byte. Since the key bytes options are used to enumerate over all key options, too many options can make the solver running time unaffordable. Thus in AES round 1 we prefer to set the threshold low, and suffer occasional false positives.

Second AES round activations set constraints that the solver uses to disqualify key candidates obtained from first round leakages. A false negative during the second round would cause fewer constraints and weaker disqualifications—so the solver may end with more keys. However, a false positive would disqualify true key values. Therefore in AES round 2 we prefer to set the threshold too high, and suffer occasional false negatives.

Our solution is to use two thresholds: one for each AES round. The first threshold (Thr_1) is set low in order to avoid false negative errors of first round activations. The second threshold (Thr_2) is set higher in order to avoid second round false positives. To calibrate the thresholds we again use the max-trace

$maxT$. We utilize the fact that we know the time-steps in which the 16 S-Box accesses occur. We use the following process to calibrate the two thresholds.

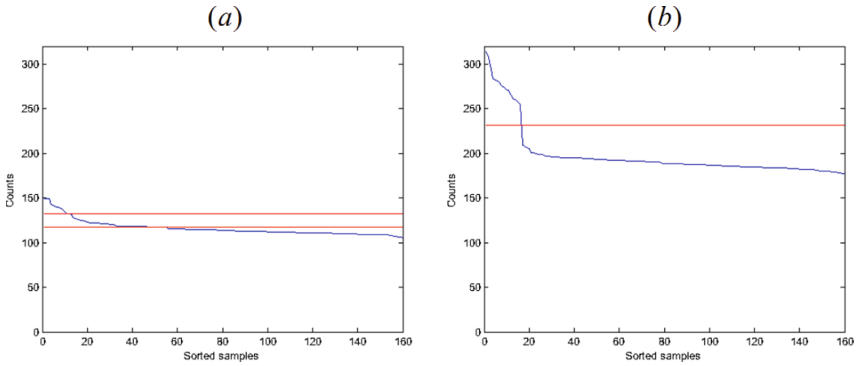


Fig. 5. The sorted maxT trace and the auto-calibrated thresholds (lines) for (a) M = 1,500,000 and (b) M = 3,750,000 measurements. We can notice on (b) a gap between the 16th and the 17th samples and the two thresholds converge.

1. Generate the max-trace $maxT$ as in Sect. 5.1.
2. Thr_1 is set to the maximal value for which $(maxT)_i \geq \text{Thr}_1$ for all 16 time-steps i at which there is a first round activation.
3. Thr_2 is the minimal value for which $(maxT)_i < \text{Thr}_2$ for all time-steps i at which there is *no* first-round activation (see Fig. 4).

If the SNR is high then peaks at the 16 true activations will be all higher than the noise—so we will get $\text{Thr}_1 \geq \text{Thr}_2$. In such a case we fall back to the method of Sect. 5.1 and set both thresholds to be $(\text{Thr}_1 + \text{Thr}_2)/2$ (see Fig. 5).

We take key candidates based on first round activations using Thr_1, and we collect the constraints from the second round activations using Thr_2.

6 Practical Results

We implemented the photonic emissions simulator in Matlab. The solver was implemented in python. The experiments were run on a relatively old Intel Core Duo T2450 2 GHz, 2 GB RAM PC running Windows Vista. We simulated the ATmega328P IC with SRAM row width of $\omega = 8$ and generated the plaintexts according to the "ω-jump" strategy of Sect. 4.

In order to evaluate the performance of our attack we performed an extensive set of experiments. All the experiments were done with $\omega = 8$, and with either L = 32 (for offset = 0) or L = 33 (for all other offsets). We used the "ω-jump" strategy to generate L plaintexts for each offset.

Table 2. A comparison between the SPEA and our E-SPEA methods.

Method	Final key candidates	Plaintexts	Measurements per plaintext	Total measurements	Time (hours)
SPEA	1 for odd offset 2^{48} for offset = 0 2^{32} for offset = 4 2^{16} for offset = 2, 6	256	5 M	1280 M	6.4
E-SPEA	1 for 75 % ~ 8 for 24 % 2^{48} for 1 %	32	1.5 M	48 M	0.5

For each plaintext we used 100 random keys, and for each key-plaintext combination we generated between M = 1,000,000–5,000,000 traces from the photonic emission simulator, with the detector at a random row $2 \leq d \leq L - 1$. We used the threshold setting of Sect. 5 to decode the traces, and used the solver to find the key. For each run we set a timer on the solver: if the run time exceeded 5000 s we stopped it and recorded a failure. Figure 6 shows the attack's behavior for various values of M. We can see that as long as $M \geq 1,500,000$ the attack works well, with the median key entropy at the end of the attack dropping below 3 bits, and a single (correct) key was found in 75 % of the runs. When $M \geq 1,500,000$ the attack takes under 10 min, on our slow PC. The results for other offsets were similar (graphs omitted).

Table 2 shows a comparison of our Enhanced SPEA with the original SPEA, and Fig. 7 shows the running time of solver. The Table shows that due to the reduced number of required plaintexts, and reduced number of required measurements M, our total attack time drops by an order of magnitude, from 6.4 h down to 30 min- while succeeding in finding a single (correct) key in 75 % of the

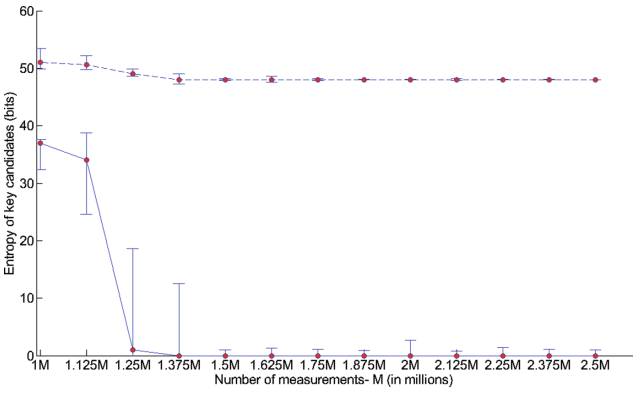


Fig. 6. The entropy of the round-1 key candidates (dashed line) and the final key candidates (solid line) as a function of the number of measurements M, for offset = 0 and using different random keys and a different detector row for every test. The upper and lower bounds indicate the 5–95 percentiles and the dots mark the median values.

cases- regardless of the offset. The E-SPEA method however had difficulty with 1 % of the cases, not getting below 2^{48} key candidates: in those cases the number of second round activations was very low and the solver reached a timeout of 5000 s without being able to reduce the number of key candidates.

7 Conclusions, Future Work and Countermeasures

In this paper we demonstrated that using cryptographic post-processing, careful plaintext selection, and better signal processing, we are able to significantly improve upon the SPEA of [15]. We are able to uniquely extract the correct key regardless of the offset at which the SBox is placed in SRAM. We achieve this while reducing the required number of photonic measurements by an order of magnitude, which directly implies a similar drop in the attack's time complexity. Our cryptographic solver is extremely efficient, with a time complexity of 2^{20}, and extracts the key within minutes on a rather old PC.

Following [15] we evaluated our attack assuming an SRAM row width of $\omega = 8$, as in the ATMega328P. However, we note that a row width of $\omega = 16$ (as in the ATXMega128A1) would pose a harder challenge: we expect to find $\approx \omega^{32}/2^{128} = 1$ key candidates that fit the leakage from the first two AES rounds, as opposed to the $\approx 2^{-32}$ expected when $\omega = 8$. I.e., in the intermediate stages we will have many more key candidates, the run time will be longer, and the attack will terminate with more possible keys, than when $\omega = 8$. Conversely, if $\omega = 32$ then our attack should become equally efficient as when $\omega = 8$: we can set the detector on the *column*-access transistor. We leave evaluating alternative SRAM configurations for future work.

Note also that our photonic emissions simulator allows us to test hypothetical lab setups, since we can experiment with the lens area and height above the IC,

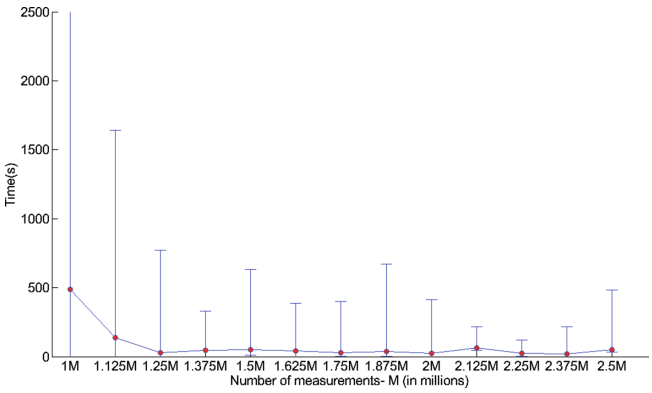


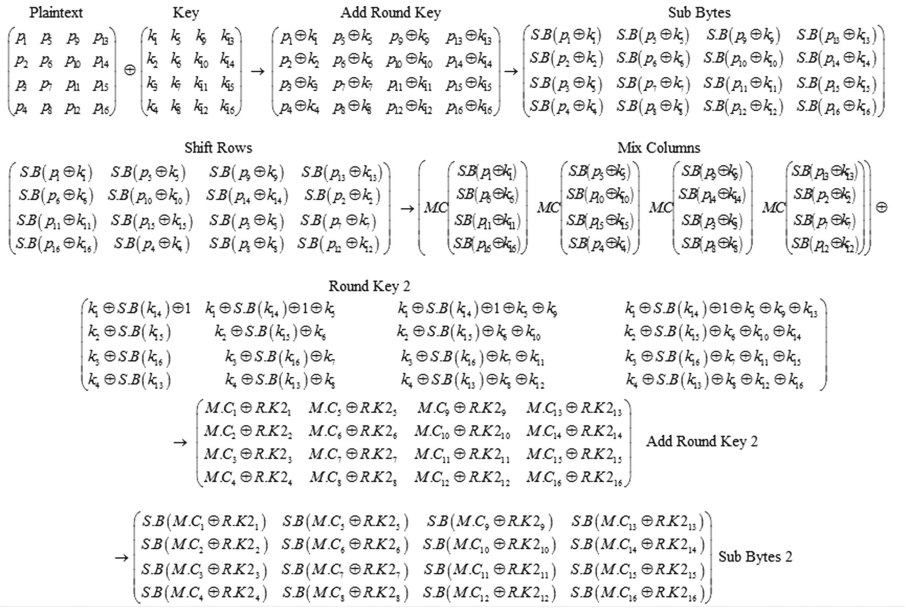
Fig. 7. Solver running time for different M values, for offset = 0 and using different random keys and a different detector row for every test. The upper and lower bounds indicate the 5–95 percentiles and the dots mark the median values.

the supply voltage, the temperature, and the detector sensitivity. It would be interesting to use the simulator's results to guide the design of better future detectors.

The attack is susceptible to countermeasures such as delays and dummy operations which can obfuscate the time a photonic emission may occur. Masking also can make the attack more difficult. Memory protection countermeasures such as memory encryption or scrambling have no effect on the emission pattern, but they can make the preliminary stage of finding the SBox values inside the SRAM memory more difficult.

Appendix

The AES Process Until the Second SubBytes Operation



References

1. Bascoul, G., Perdu, P., Benigni, A., Dudit, S., Celi, G., Lewis, D.: Time resolved imaging: from logical states to events, a new and efficient pattern matching method for VLSI analysis. Microelectron. Reliab. **51**(9), 1640–1645 (2011)
2. Bernstein, D.J.: Cache-timing attacks on AES (2004). Preprint, http://cr.yp.to/papers
3. Bertoni, Y.M., Grassi, L., Melzani, F.: Simulations of optical emissions for attacking AES and masked AES. In: Chakraborty, R.S., Schwabe, P., Solworth, J. (eds.) Security, Privacy, and Applied Cryptography Engineering (SPACE). LNCS, vol. 9354, pp. 172–189. Springer, Verlag (2015)

4. Carmon, E., Seifert, J.-P., Wool, A.: Simple photonic emission attack with reduced data complexity. Cryptology ePrint Archive, Report 2015/1206 (2015). http://eprint.iacr.org/2015/1206
5. Chynoweth, A., McKay, K.: Photon emission from avalanche breakdown in silicon. Phys. Rev. **102**(2), 369 (1956)
6. Di-Battista, J., Courrege, J.C., Rouzeyre, B., Torres, L., Perdu, P.: When failure analysis meets side-channel attacks. In: Mangard, S., Standaert, F.X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 188–202. Springer, Heidelberg (2010)
7. Egger, P., Grützner, M., Burmer, C., Dudkiewicz, F.: Application of time resolved emission techniques within the failure analysis flow. Microelectron. Reliab. **47**(9), 1545–1549 (2007)
8. Ferrigno, J., Hlavác, M.: When AES blinks: introducing optical side channel. Inf. Secur. **2**(3), 94–98 (2008)
9. Krämer, J., Kasper, M., Seifert, J.-P.: The role of photons in cryptanalysis. In: 19th Asia and South Pacific, Design Automation Conference (ASP-DAC), pp. 780–787. IEEE (2014)
10. Krämer, J., Nedospasov, D., Schlösser, A., Seifert, J.P.: Differential photonic emission analysis. In: Prouff, E. (ed.) COSADE 2013. LNCS, vol. 7864, pp. 1–16. Springer, Heidelberg (2013)
11. Nedospasov, D., Seifert, J.-P., Schlosser, A., Orlic, S.: Functional integrated circuit analysis. In: IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), pp. 102–107. IEEE (2012)
12. Newman, R.: Visible light from a silicon pn junction. Phys. Rev. **100**(2), 700–703 (1955)
13. Osvik, D.A., Shamir, A., Tromer, E.: Cache attacks and countermeasures: the case of AES. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 1–20. Springer, Heidelberg (2006)
14. Schlösser, A.: Hot electron Luminescence in silicon structures as photonic side channel (in German). Ph.D. thesis, Faculty of Mathematics and Natural sciences, Berlin Institute of Technology (2014)
15. Schlösser, A., Nedospasov, D., Krämer, J., Orlic, S., Seifert, J.-P.: Photonic emission analysis of AES. In: Workshop on Cryptographic Hardware and Embedded Systems (CHES) (2012)
16. Schlösser, A., Nedospasov, D., Krämer, J., Orlic, S., Seifert, J.-P.: Simple photonic emission analysis of AES. J. Cryptographic Eng. **3**(1), 3–15 (2013)
17. Selmi, L., Mastrapasqua, M., Boulin, D.M., Bude, J.D., Pavesi, M., Sangiorgi, E., Pinto, M.R.: Verification of electron distributions in silicon by means of hot carrier luminescence measurements. IEEE Trans. Electron Devices **45**(4), 802–808 (1998)
18. Song, P., Stellari, F., Huott, B., Wagner, O., Srinivasan, U., Chan, Y., Rizzolo, R., Nam, H., Eckhardt, J., McNamara, T., et al.: An advanced optical diagnostic technique of IBM z990 eserver microprocessor. In: Proceedings IEEE International Test Conference (ITC), p. 9. IEEE (2005)
19. Weste, N., Harris, D., Design, C.: A Circuits And Systems Perspective, 4/E. Pearson Education, (2010)