Side-Channel Evaluation Report on Implementations of Several NIST LWC Finalists

Lejla Batina, Ileana Buhan, Łukasz Chmielewski, Ellen Gunnarsdóttir, Vahid Jahandideh, Tom Stock, and Léo Weissbart

Cryptographic Engineering & Side-Channel Analysis (CESCA) Lab,
Radboud University Nijmegen, The Netherlands
https://cescalab.cs.ru.nl/

August 19, 2022

1 Introduction

This report summarizes preliminary results of side-channel analysis of several NIST LWC finalists by Cryptographic Engineering & Side-Channel Analysis (CESCA) Lab [1] from Radboud University Nijmegen, The Netherlands, between April 1st and August 19, 2022.

We have performed side-channel evaluation of software implementations of the following finalists:

- ASCON (unprotected implementation, Section 2 and first-order masked implementation, Section 3);
- Xoodyak (Section 4);
- ISAP (Section 5).

In our evaluation we first perform a short assessment on how feasible a side-channel attack is and when needed we verify that our attacks work on simulated traces. Subsequently, we mostly use test vector leakage assessment (TVLA) [1, 2] for concentrated leakage detection and Correlation Power Analysis (CPA) [3] for performing unsupervised attacks; for details, about attacks on each finalist see the relevant sections.

1.1 Measurement Setup

We have evaluated all software implementations on a a Piñata development board [1] with an STM32F407IGT6, 32-bit ARM-based microcontroler, running at the clock frequency of 168 MHz and powered with a stabilized power supply at 3.3V. The board is modified to perform SCA through power consumption. We measure the power consumption with a current probe [1]. Communication between the board and the recording computer is implemented with UART.

We collect power traces with the Picoscope model 3206D. The measurements are performed with a sampling frequency of between 100 MHz and 1.0 GHz with 8-bit resolution and the trigger is implemented in software.

[1]`https://cescalab.cs.ru.nl/`
[1]Pinata Board: `https://www.riscure.com/product/pinata-training-target/`
[1]Current Probe: `https://www.riscure.com/product/current-probe/`

2 Power Analysis of ASCON

In this section we first analyze how to perform a CPA attack on ASCON and then we run TVLA and apply the CPA distinguisher on the acquired traces.

Samwel and Daemen [4] have successfully recovered the complete the key with a DPA attack on a hardware implementation of ASCON [2]. They mounted a bitwise attack on the x_0 and x_1 output state registers after a single round of permutation in the initialization of the algorithm. They found that three (x_0, x_1, x_4) of the five output registers contain a non-linear term with both a key bit and a nonce bit, allowing for a DPA attack. As these terms are algorithmically identical in the hardware and software setting, this attack should in theory be possible in both domains. Our attack is therefore based on the successful attack proposed by Samwel and Daemen.

2.1 Selection functions and power model

We use the same selection functions as in [4]. These selection functions consider the activity of a single bit of the respective output register after one round of permutation in the initialization phase of ASCON. To simplify, the functions omits all constant values as they do not contribute to the activity of the register. As these selection functions focus on activity caused by non-linear interaction between one known and one secret value we can apply a bitwise divide-and-conquer technique.

In ASCON [5] all permutations are done over the state of five registers of 8 bytes. A static value is loaded into the first register, the key is loaded in the second and third register and the nonce is loaded in the last two registers. After this twelve rounds of permutations are executed on the state to finish the initialization. A round of permutation consists of three phases; adding constants, executing a 5-bit S-box and combining each register with two linear shifts of itself. As adding constants does not contribute to the activity of the output state, this phase is ignored in the selection functions.

Equation 1 and Equation 2 show the activity of the S-box for their respective output register. This is generalized and combined with the linear diffusion in Equation 3 and Equation 4; these are the final equations that are being used in this DPA attack. Due to the linear diffusion layer, three key bits are needed to compute a single output state bit. The number of possible keys for each output state bit is therefore 2^3 which amounts to eight possible key values.

$$y_0 = k(m' + 1) + m \tag{1}$$

$$y_1 = m(k + 1) + m' \tag{2}$$

$$\begin{aligned} S0_i(M, K^*) =& k_0^*(m_i' + 1) + m_i + k_1^*(m_{i+45}' + 1) \\ &+ m_{i+45} + k_2^*(m_{i+36}') + m_{i+36} \end{aligned} \tag{3}$$

$$\begin{aligned} S1_i(M, K^*) =& m_i(k_0^* + 1) + m_i' + m_{i+3}(k_1^* + 1) \\ &+ m_{i+3}' + m_{i+25}(k_2^* + 1) + m_{i+25}' \end{aligned} \tag{4}$$

The power model used in Samwel and Daemen's work is the Hamming Distance model This power model is often used in the hardware setting as the power consumption is related to the bit

[2]Note that we use the term DPA and CPA interchangeably, when it is clear that CPA is a type of DPA using Pearson correlation.

flips in the registers of the chip. However, in the software setting, we have no guarantees that the relevant variables are present in the registers at the required times to capture the power usage. Therefore, we use the Hamming Weight of the output of the selection functions instead. This is more viable as the variables will be transmitted over the bus of the chip, and the number of high bits determines the power usage.

2.2 TVLA

The code used to record the TVLA traces is the unmodified official 32-bit low-size ASCON implementation provided by the ASCON team [6]. We setup a fixed-versus-random test on the nonce value. For this experiment a total of 50 000 traces were recorded at 1GHz sampling rate. The traces are cropped to contain only the encryption phase to the point were the nonce is consumed. Traces are then aligned for better results.

We performed alignment using a windowed offset and correlation approach. With this approach each trace is compared to a hand picked reference trace. A specific sample window is selected to reduce the computation complexity, this window of the trace is compared to the window of the reference trace with a specific offset. This comparison is done $2x$ times, where x is the offset limit and and the offset moves from $-x$ to x. The comparison is done by computing a Pearson correlation between the reference and offset window. The offset with the highest correlation value is chosen and the whole trace will be offset with this value. If the best correlation value of a trace does not exceed the specified correlation threshold this trace is removed from the trace set. The offset steps ($2x$) are removed from the beginning and end of each trace as these samples cannot be guaranteed to have sane values due to the shifting. The aligning sample window for this experiment was set to (15 000;16 500), this window was chosen upon manual inspection of the trace set. The reference trace was set to the first trace in the set, also upon manual inspection. The offset steps value was set at 100 and the correlation threshold was set at 0.85. After the alignment there were 43 597 traces left in the set, each having 119 800 samples.

We perform the TVLA test by executing Welch t-test for each group of samples at the same index over all traces as described by Goodwill et al. [1]. This statistical test is an extension of the students t-test for unequal variance and sample sizes. Let T be a set of traces, where each trace T_i consists of m samples. T^j denotes all samples at index j across T. The leakage L is computed for each j in T, where $0 \leq j < m$, producing an output of m values. Equation 5 shows the Welch t-test for the two groups for a single value j. If $|L_j| > 4.5$ for $L_j \in L$ we reject the null hypothesis that the fixed and dynamic groups for T^j have equal means and thus that there is no leakage between these two groups. This threshold implies a confidence of 99.999% for $n > 5\,000$.

$$L_j = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{s_1^2}{N_1} + \frac{s_2^2}{N_2}}} \tag{5}$$

The result of the TVLA test is shown in Fig.1.

2.3 Characterization for CPA

Since we want to execute CPA, we now focus on the initialization function of ASCON. We work with a fixed-key setup and modify the nonce at random for all recordings. To reduce the effort of trace recording, we isolate the initialization function from the encryption phase and only measure this operation. Fig. 2 shows the same TVLA test as in Fig. 1, but only for the initialization function

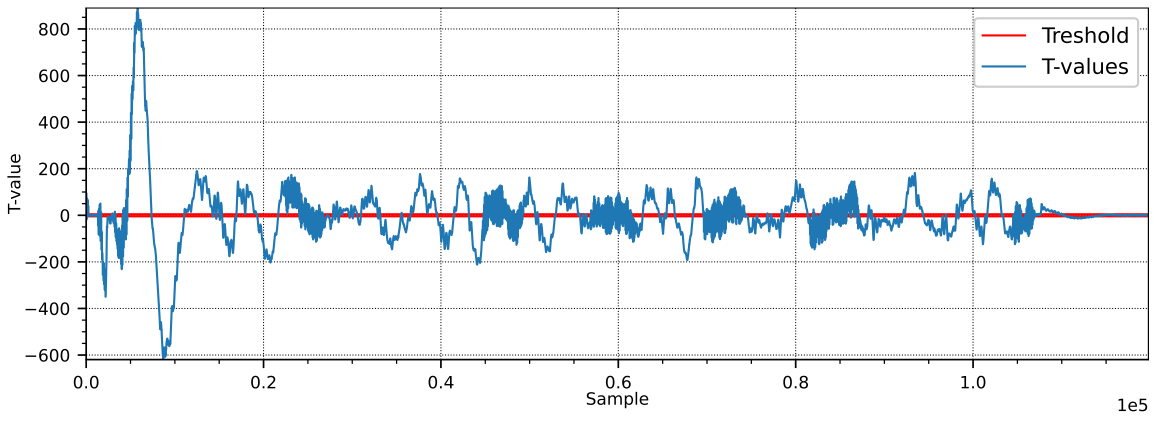

Figure 1: The TVLA results. The red lines indicates the leakage threshold and the blue line the correlation values. As can be seen, the correlation values are significantly higher then the leakage threshold. This TVLA test was run with 50 000 traces.

with 100 000 traces. This TVLA results can be used to crop properly the traces recorded for CPA to the exact place of the operation.

The alignment was done with the same methodology as described in the previous section with the first trace acting as reference trace, the aligning window set to (3 000;4 000) of the cropped trace, the offset steps value at 100 and with a threshold of 0.85. During this alignment no traces were removed from the trace set. The alignment window was chosen based on the input correlation of the output state registers and the computed intermediate values, which are shown in Figure 3 and Figure 4. These plots show the highest correlation within this window, we therefore expect that the DPA correlation will also occur in this window.

To extract the best key guess, we compare the correlation between the each group of samples at the same index over all traces with the computed predictions using the selection function for each trace and each possible key in the selection function. This results in a correlation matrix with the dimension of the number of samples in a trace times the possible keys. The key with the highest absolute correlation value in this matrix is the best key guess.

2.4 CPA results

Figure 6 shows the success probability of the CPA attack for both halves of the key. It shows that the key can reliably be extracted with the use of 500 000 traces. Interesting to note is that extracting the second half of the key has a consistent higher success rate.

The progression of the attack can be seen for a single bit in Figure 5. It shows the correlation value and rank at specific trace intervals as well as the correlation of each sample for the maximum number of traces.

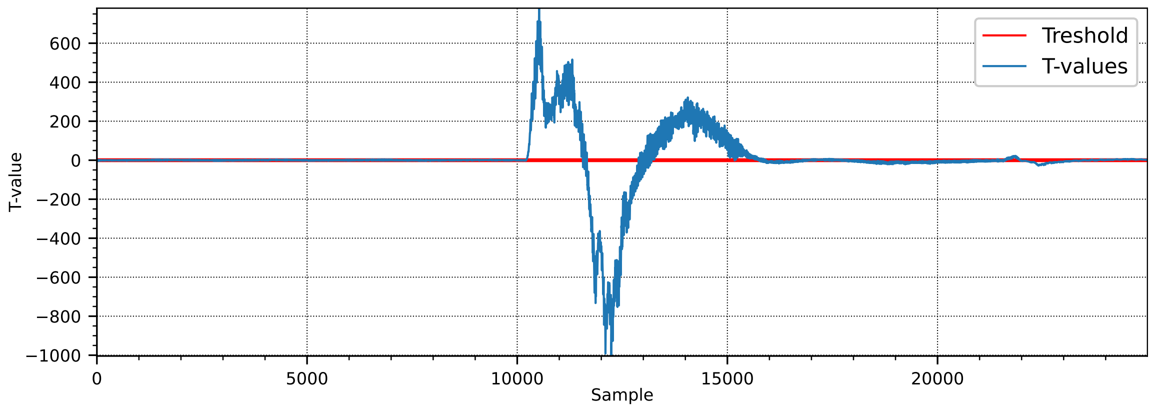

Figure 2: The TVLA results with the ASCON initialization function wrapped in NOP operations. The execution of the leaking code can clearly be seen in the middle. The red lines indicate the leakage threshold and the blue line the correlation values. This TVLA test was run with 100 000 traces.

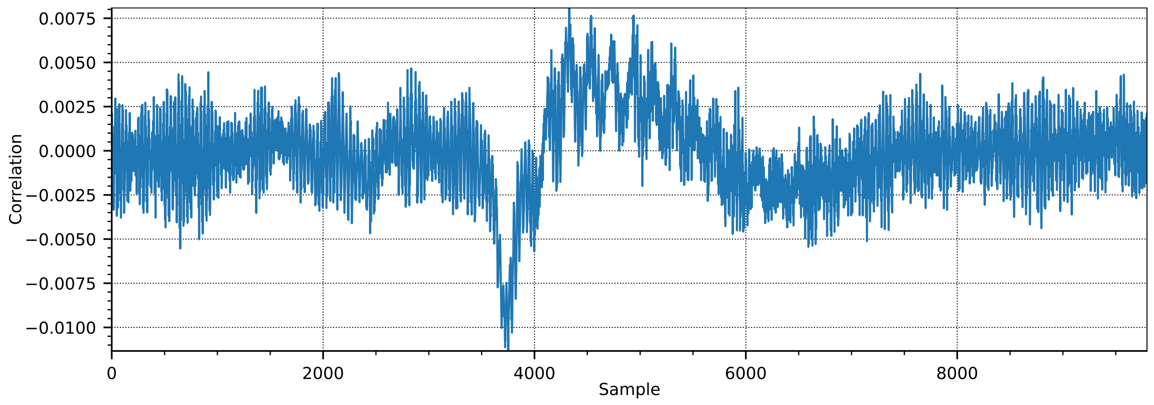

Figure 3: Pearson correlation of the first output bit of the state x_0 with each sample. Between sample 3 500 and 6 000 we can see a correlation peak.

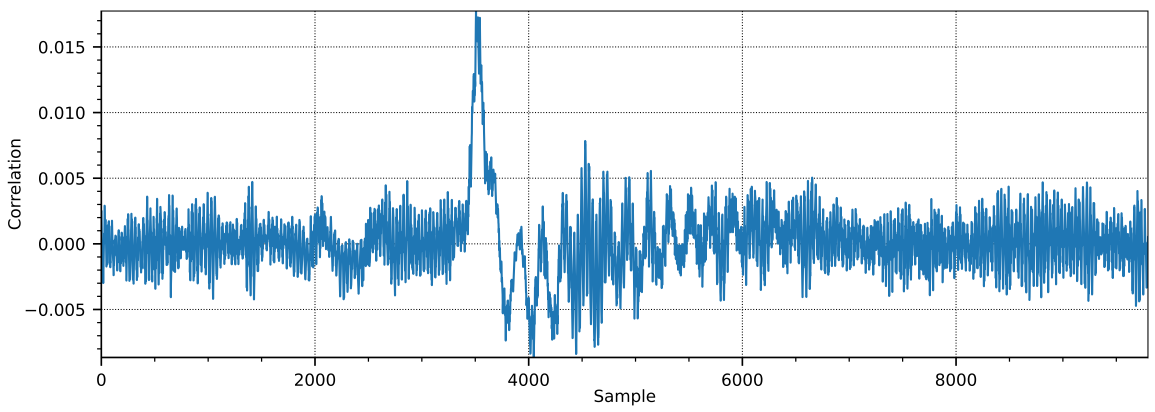

Figure 4: Pearson correlation of the first output bit of the intermediate value of x_0 with each sample. Between sample 3 000 and 4 000 a big correlation is apparent.

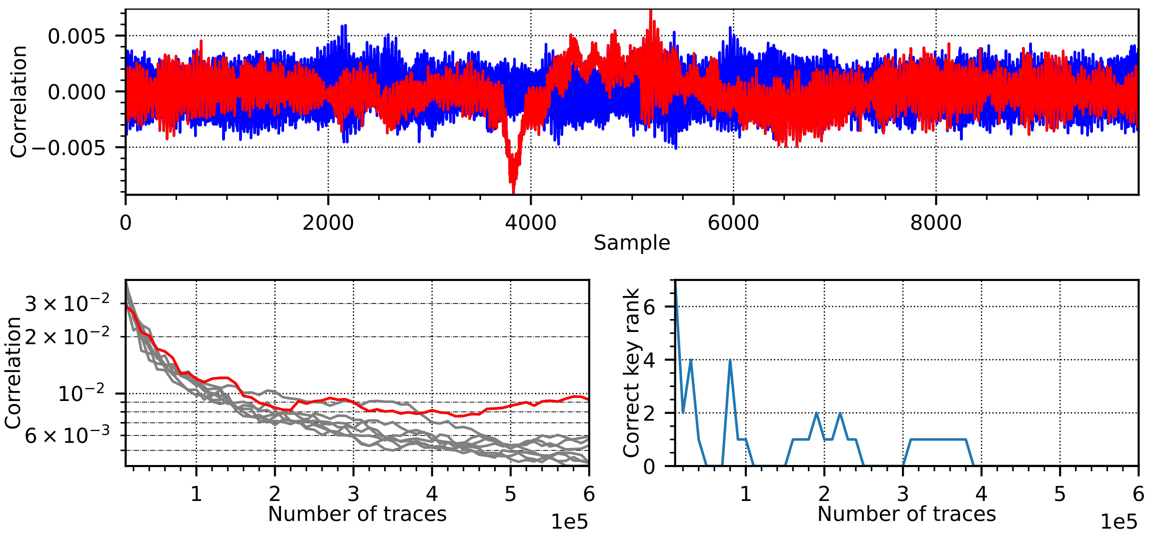

Figure 5: The CPA progression results of bit 1 of the key. The top plot showcases the correlation for each key with each sample. The correct key is highlighted in red. It shows a clear correlation of the correct key, even though the correct key only has a correlation with a factor of two higher than the next best key hypothesis. The bottom left plot shows the highest correlation value of each possible key for a given number of traces used in the analysis. The bottom right plot shows the rank of the correct key at the same interval of number of traces used in the analysis. Rank seven is the worst key ranking, and thus having the lowest correlation. Rank zero is the highest key ranking and means the correct key has the highest correlation. We can see that from 230 000 traces used in the analysis the correct key became apparent and could be extracted.

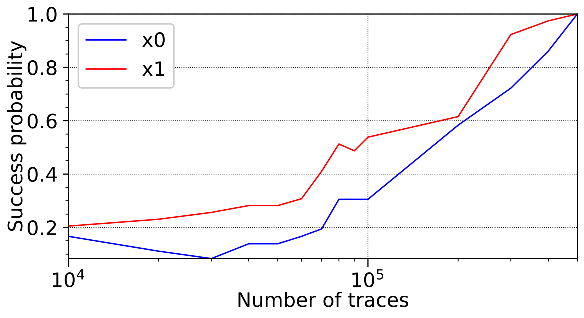

Figure 6: The probability to get the correct key for both halves of the key for a given number of traces. It shows that with 600 000 traces you are guaranteed to get the correct key with our setup.

3 Power analysis on masked ASCON

In this section we describe results of a second-order CPA attack on traces acquired from a masked ASCON implementation. The official code submission of ASCON [6] includes a masked Armv6 implementation. This implementation utilizes masks with (almost) no fresh randomness, between two and four rotated shares. We performed a 2nd order DPA attack targeting the protected implementation with two shares.

This experiment is run with the same setup as the unprotected version. As we already established a successful attack on the unprotected version, we opted not to perform an additional TVLA test for the protected version. In this attack we used the same selection functions and power model as in the first order attack.

3.1 Characterization for CPA

For this experiment we only recorded the first permutation round of the initialization phase of ASCON. We modified the official protected implementation to stop after this permutation round. All the required parameters were send over the serial connection. All data, apart from the nonce was fixed. The nonce was generated with the Python Numpy random number generator. A total of 15 000 000 traces were collected.

The post processing consisted of cropping the trace in the same manner as the unprotected attack. Alignment was omitted in this case, as the traces were aligned well already. The cropping sample window used was (9 000:20 000).

A total of two attacks were performed, a first-order attack and a second-order attack. The first-order attack is identical as in the previous section and is done to confirm that there is no first order leakage in the protected implementation. The second order attack aims to exploit the second-order leakage by performing the attack on the combined samples of each trace. For the combination of the samples we follow Prouff et al. [7], using the 'mean centred product of samples' approach. The output of this combination function is used as input for the first order attack.

Let T be a set of traces having t entries that is indexed with $0 \leq i < t$. Each trace consists of a set M, containing m samples, indexed by $0 \leq j < m$. Each sample has an average across T given by μ^j. The combined output sample is described in Equation 6 and given as $C_s^i \in C^i$, C^i is the set that contains the mean centred product of all possible subsets of J in T_i. $|J|$ determines the number of subsets and thus the order of the combination function. In our case $|J| = 2$. The value of $|C^i|$ is given by $m^{|J|}$ and is exponential with base m and the order as exponent.

$$C_s^i = \prod_{j \in J} \left(t_i^j - \mu^j \right) \tag{6}$$

As to reduce the computation footprint, we limit the subsets of T_i. Instead of using all possible subsets, we use all possible subsets of two sample windows T^x and T^y of T. The sample windows used in this attack are given as $5\,750 \leq m^{'} < 6\,050$ and $5\,800 \leq m^{''} < 6\,100$. These windows are chosen based on the correlation values of the relevant output state registers. These can be seen for the first output byte of each of the two shares of the x_0 register in Figure 7 and Figure 8. The sample windows are centred on the first and highest peak, as this is the most likely place for the sensitive data to leak.

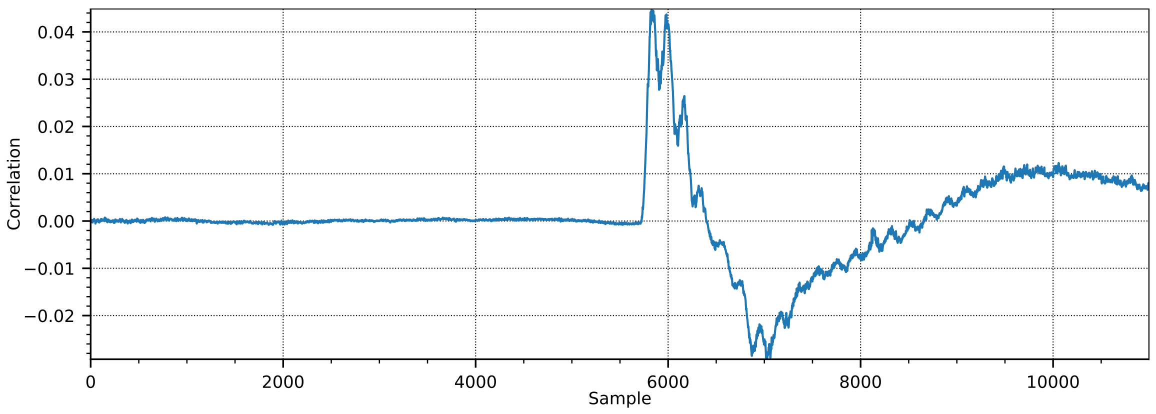

Figure 7: Correlation values of the first byte of the first share of the output state with each sample.

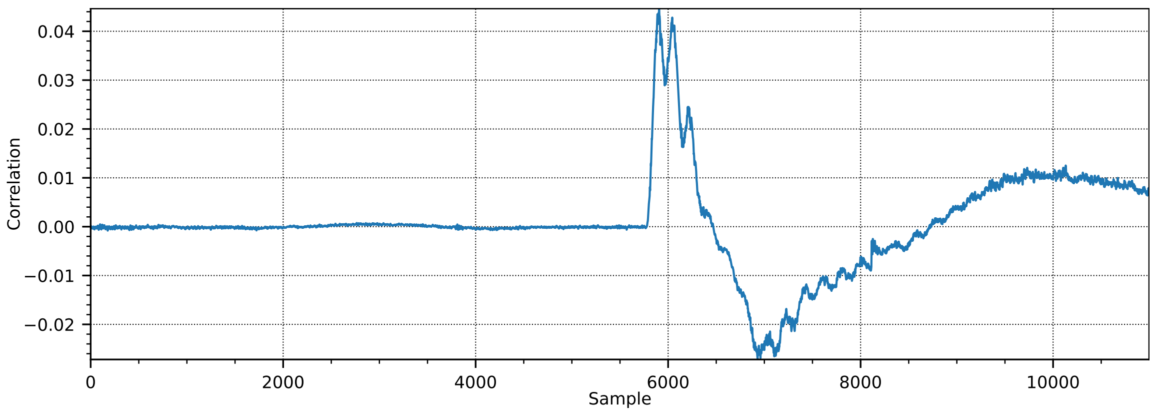

Figure 8: Correlation values of the first byte of the second share of the output state with each sample.

3.2 CPA Results

Figure 9 shows the results for the first order attack. It is apparent that no correlation has taken place and that the attack was unsuccessful. The results of second order attack can be seen in Figure 10. This attack also showed no correlation and was unsuccessful with 15M traces. We suspect that the reason for that is not enough traces being used.

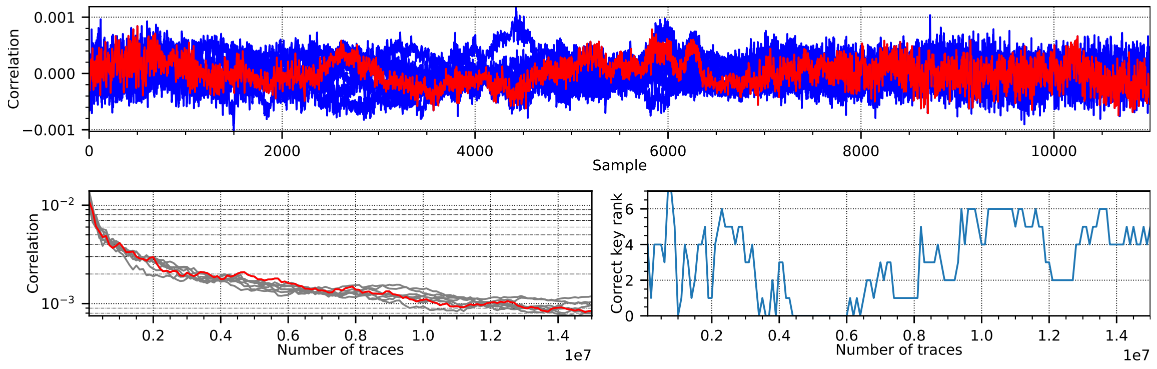

Figure 9: The result of first-order CPA on the masked ASCON implementation.

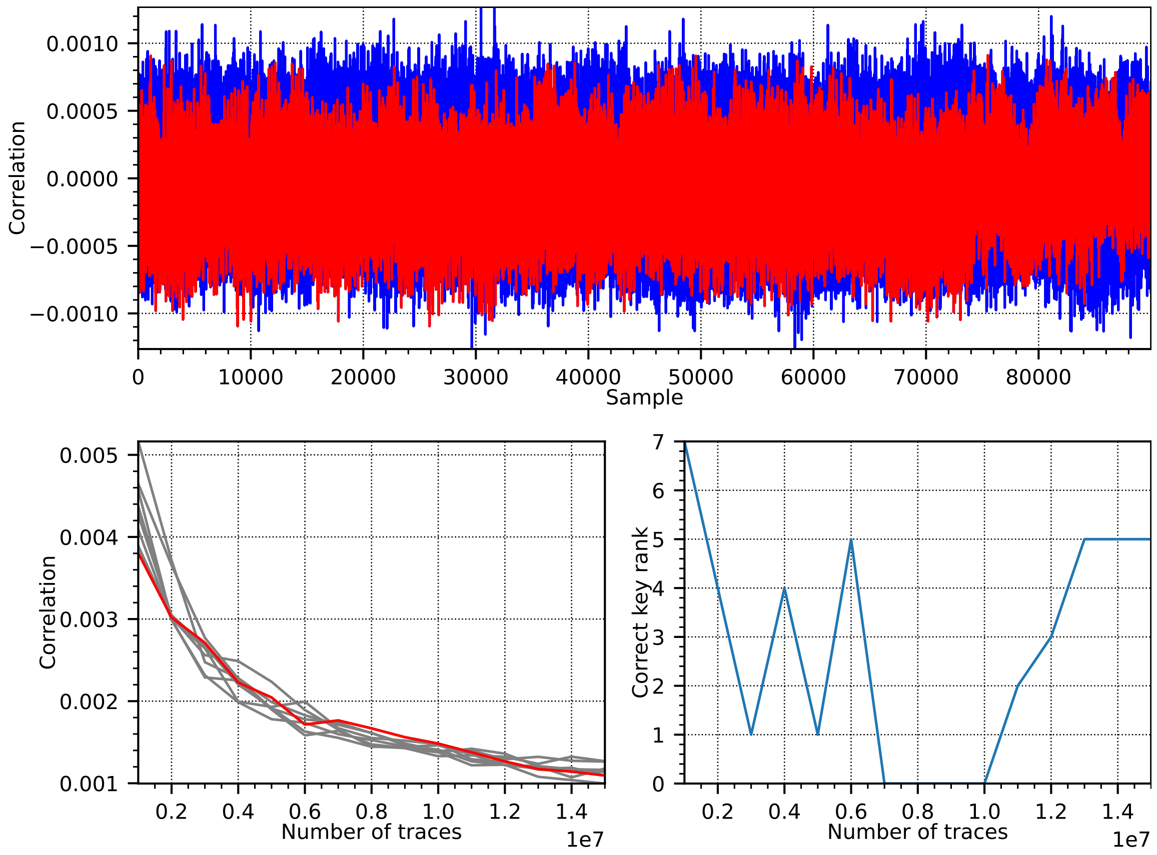

Figure 10: The result of second-order CPA on masked ASCON implementation.

4 Power Analysis of Xoodyak

In this section we first analyze how to perform CPA on Xoodyak and then we run TVLA and CPA on the acquired traces.

Samwel and Daemen [4] published a DPA side-channel attack on Keyak, an authenticated encryption scheme based on Keccak-p. Keccac-p is a permutation close in design to the Xoodoo permutation function used in Xoodyak. The attack recovers two bits of Keyak state using a single bit leakage in the S-box on the non-linear step of the permutation and; this attack is also extended to a row of the state. We can adjust this attack on Xoodyak because of the similarities between Xoodoo and Keccak-p. We target the state of Xoodyak before an Absorb function. The first operation of absorb in most modes is to apply the Xoodoo permutation function. This permutation consists of three linear functions on the planes of the state (i.e., Theta, ρ_{east}, a non-linear function Chi and again a linear function (i.e., Phi west).

$$\chi(a_{(x,y,z)}) = a_{(x,y,z)} + (a_{(x+1,y,z)} + 1)a_{(x+2,y,z)} \tag{7}$$

4.1 TVLA

In this section we attack a Xoodyak software implementation optimized for the ARMv7M microprocessor from XKCP [8]. First we used TVLA to evaluate here potential leakage vectors in Xoodyak.

First, we perform a fixed-versus-random key TVLA test with 50 000 traces sampled at 1GHz. We used the traditional leakage detection threshold of ± 4.5. The result of this test is shown in Fig. 11. The power trace captured the full encryption of a 20 byte message from the initialization function using a 16 byte key and a 16 byte nonce, absorb function of a 16 byte associated data, encrypt of the message and squeeze of the tag. From the t-value trace, we can see an activity only during the initialization function. During this function, the key is xored in the state and should leak in a straight forward way because of load and store operations.

Secondly, we perform a fixed-versus-random nonce TVLA test with 10 000 traces sampled at 500MHz. The result is shown in Fig. 12. We can see high activity in the absorb of the nonce and smaller activity in several part of the later execution of the encryption process.

4.2 Simulated CPA

For this attack the simulated traces are the state taken after the first message has been absorbed and the following linear layer has been executed for only the first round constant. Then the prediction matrix is made by performing the non-linear layer (Chi) for the first column of the state only with two bits of linear-layer(K). This gives two bits of the state. The result is correlated with the bit of the state after performing the Chi function to obtain simulation traces. Given this simulation, the correct key stands out after a few traces as shown in Fig. 13.

We then collected traces for the CPA attack on the XKCP [8] implemenation. We capture the power consumption during the execution of the permutation that follows the absorb(nonce) function. The traces are then cropped to contain only the first round Chi function of the permutation. We measure the trace at a 1GHz sampling rate and applied window re-sampling and alignment to enhance the bit leakage. The result of the CPA attack is shown in Fig. 14 on 30 000 traces. The

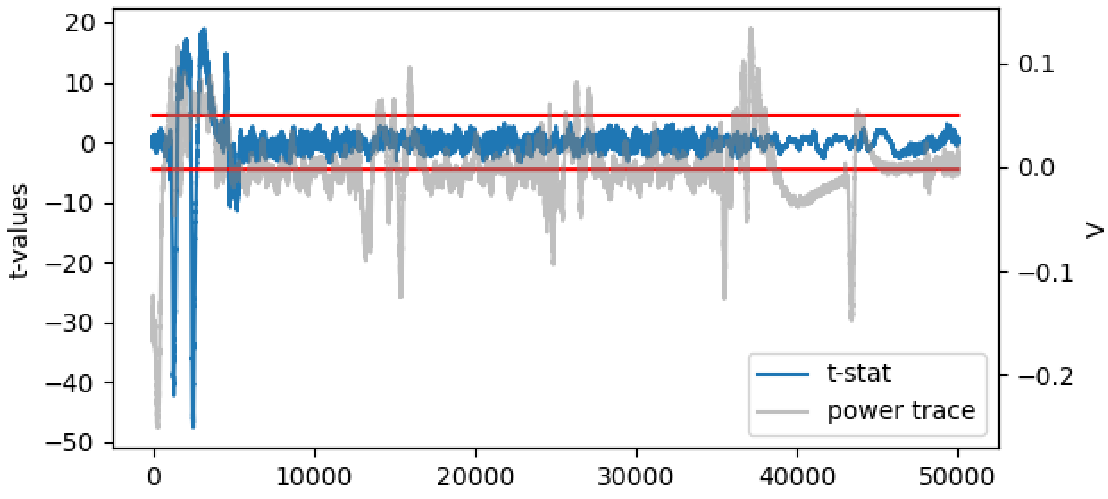

Figure 11: Xoodyak fixed-versus-random key TVLA for 50k traces; the horizontal axis represents acquired number of samples.

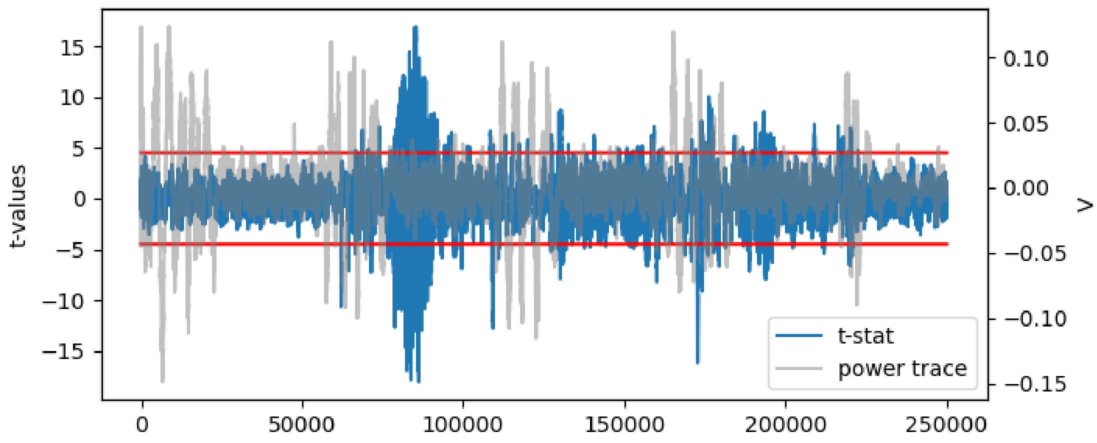

Figure 12: Xoodyak fixed-versus-random nonce TVLA results for 10k traces; the horizontal axis represents acquired number of samples.

two bits of the state can be early (within a few hundred traces), but statistically diverge of other candidates after 20 000 traces.

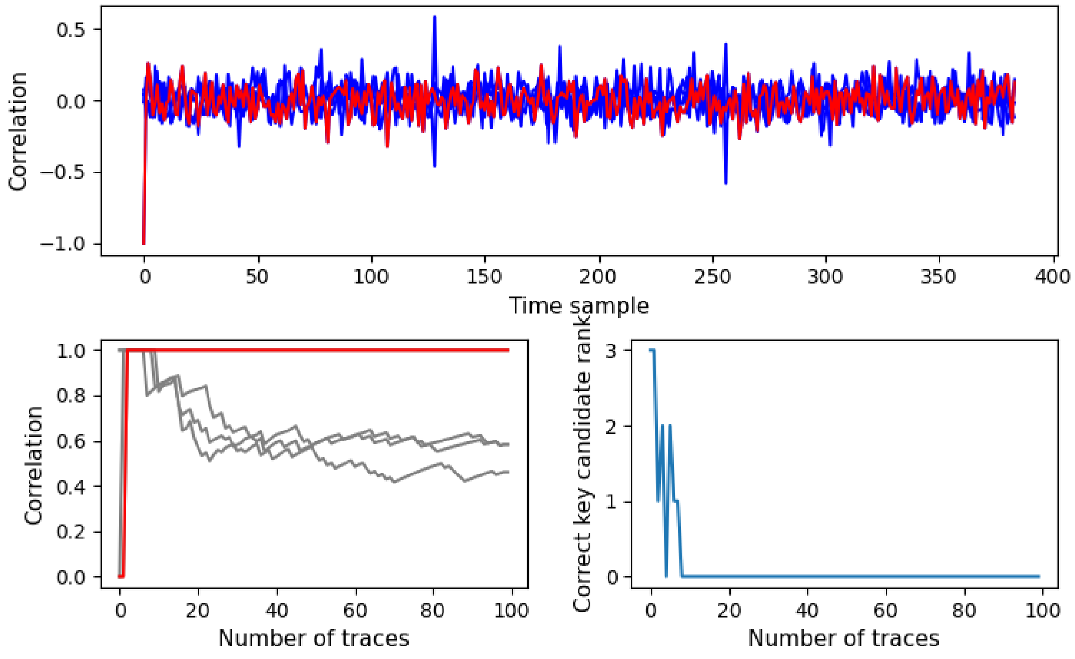

Figure 13: CPA attack on the Chi function with 100 simulated traces. The red color represents the correlation with the correct guess of two bits. In the top, it reaches -1 around the offset 0.

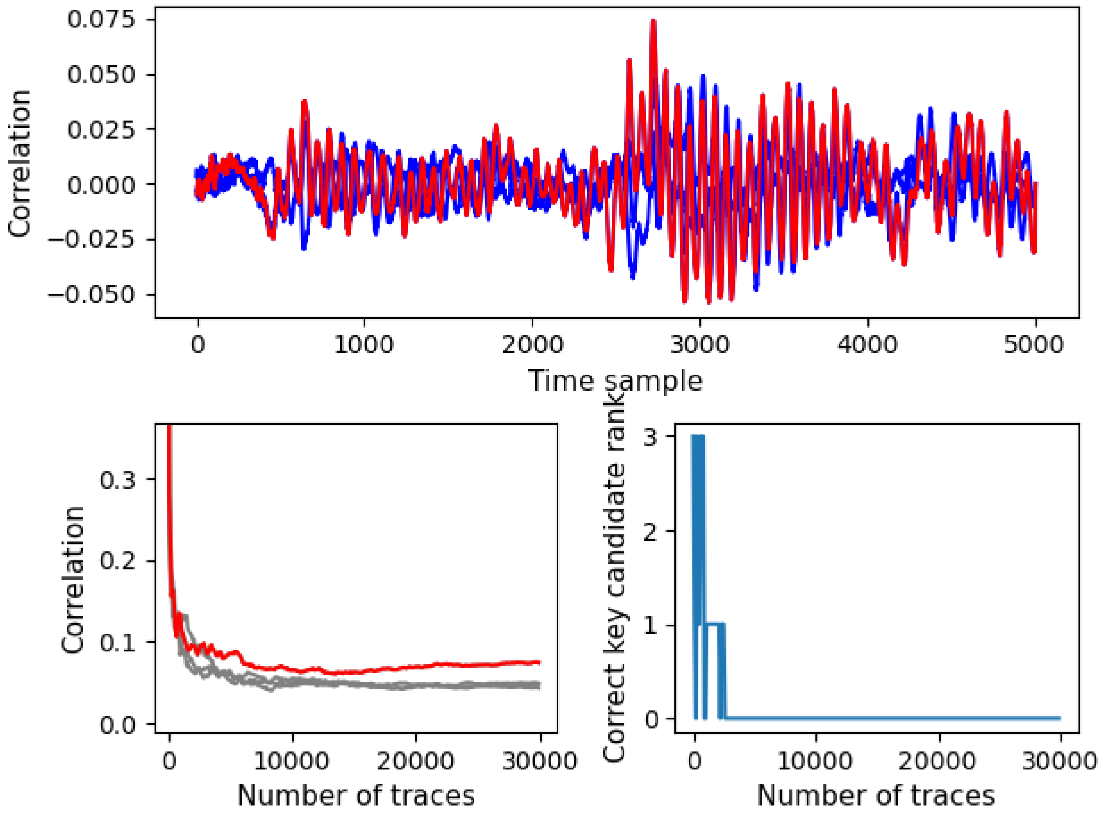

Figure 14: CPA attack on the Chi function with 30k traces.

5 Power Analysis of ISAP

In this section, we first analyze how to perform a side-channel attack on ISAP and then we run TVLA on an unprotected ISAP implementation.

ISAP is one of the AEAD candidates in the current portfolio of the NIST-LWC competition with conjectured robustness against various side-channel attacks. In this report, we study this cipher suite regarding power analysis attacks. Very briefly: we discuss how it has confronted a divide-and-conquer strategy required for DPA-like attacks; throughout the computation flow at Re-Keying, Encryption, Authentication, and Decryption. With an exception regarding the Re-Keying algorithm in the randomness absorption phase. Nevertheless, we will argue that setting the *absorption rate* r_b to a minimum possible, i.e., $r_b = 1$, has damped information leakage. We also note how utilizing separate IVs and distinct computation paths for Encryption and Authentication adds to its implementation security. The current state of the soft-analytical attacks against ISAP is also part of this report. Another relevant fact we cover in this report is the role of leakage-resilient *tag* comparison added to version 2 of ISAP. Moreover, the impact of unmasked-*key* transfer occurring at the start of Re-Keying is included. For it, we discuss how averaging observations of *key* being copied from memory, combined with precise *profiling*, might open the door for the *master key* recovery, at least for the 8- and 16-bit structures.

Structure of this section. A brief overview of the construction of ISAP is in Part 5.1. Part 5.2 explains impossibility of the divide-and-conquer strategy and other considerations about DPA attacks for ISAP. In the absence of practical DPA attacks, the attacker may go for soft-analytical attacks that are profiling attacks and assume the strongest adversary. Part 5.3 reviews relevant soft-analytical attacks. There, the impact of the *word size*, unmasked-*key* transfer, *quantization*, and *over-sampling* is covered. Part 5.4 is the conclusion and supplementary notes.

5.1 Composition of ISAP

ISAP has four variants that only differ in the parameters and the internal permutation π. Ascon-p, with state size $n = 320$, and Keccak-p, with $n = 400$, are two lightweight options for π. Nonce N and master key K are 128 bits. To encrypt a plaintext $M \in \{0, 1\}^*$, first, it is partitioned into M_i blocks with r_h bits. Except for the last block, that can be less than r_h bits. Then, the Encryption algorithm, as in Figure 16, is called. The encryption starts with creating a fresh $(n-128)$-bit session key by calling the Re-Keying algorithm as in Figure 15. In this call to Re-Keying, randomness bits Y_is are the nonce bits. With squeezing the sponge structure, M_is are masked to produce ciphertext block C_is. To authenticate C_i blocks and arbitrary-length associated data A_i blocks, a 128-bit tag T is generated with the Authentication algorithm as in Figure 17. For decryption, upon reception of a tuple (N, A_is, C_is, T), first, the validity of tag T is checked by computing the tag and comparing it with the given one. Then, the Encryption algorithm is called to unmask C_is and produce the corresponding plaintext blocks. ISAP parameters for its four variants are in table 1. For a more detailed explanation, see the official document [9].

5.2 Divide-and-conquer attacks

A Requirement of DPA-like attacks. These attacks need an intermediate variable inside the cipher that is a function of only a fraction of the target secret. More strictly, if K is the target, there

Variant	π	n	a	b	e	h	r_b	r_h
1	Ascon-p	320	12	1	6	12	1	64
2	Keccak-p	400	8	1	8	16	1	144
3	Ascon-p	320	12	12	12	12	1	64
4	Keccak-p	400	12	12	12	20	1	144

Table 1: Parameters of ISAP for different variants.

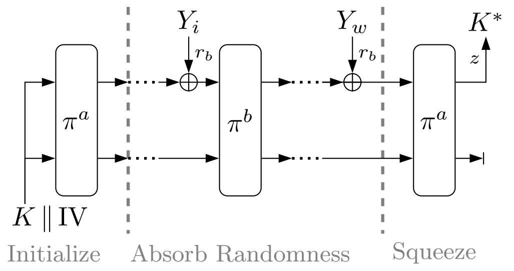

Figure 15: Re-Keying. The parameters a and b denote the number of rounds of permutations, and z is the number output bits.

should be a variable V and a function f such that $V = f(P, K_o)$, where P is a known (non-fixed) parameter, and $K_o \in \{0,1\}^t$ is part of K. For a practical attack, 2^t should be small enough that the attacker can examine all possibilities for K_o. It is also required that the relation $V = f(P, K_o)$ be valid during multiple executions of the algorithm under the same value of K_o.

Generalized case. If multiple variables inside the cipher collectively give $f(P, K_o)$, then divide-and-conquer is still possible. More precisely, if for collection of m intermediates $\{V_1, V_2, \ldots, V_m\}$ we can write $g(V_1, V_2, ..., V_m) = f(P, K_o)$, where g is some combination of input arguments, it is doable to conduct a DPA attack[3].

For ISAP, divide-and-conquer attacks are avoided by making it infeasible to find such intermediates; we will examine each algorithm to verify this property.

5.2.1 Attacks regarding Re-Keying

The Re-Keying, with schematic as in Figure 15, is used both for session key and tag generation. Value of IV, z, and Y_is are different for the two use cases. Note that IV is not random; for a fixed π, it has only three static values $\{\mathrm{IV}_{\mathrm{A}}, \mathrm{IV}_{\mathrm{KE}}, \mathrm{IV}_{\mathrm{KA}}\}$, each for a different purpose.

Initializing phase. In this phase, since no random value is involved, we can not find any function $f(P, K_o)$ with an associated intermediate variable V. Note that if IV was random, it could deteriorate the security of the implementation. With random IV, DPA attacks, like those for Ascon AEAD candidate, were possible. See the section for Ascon in this report.

[3]V is known as *sensitive* variable, f is *selection* function, and g is usually simply an XOR operation. Some works suggest that f is better to be non-linear.

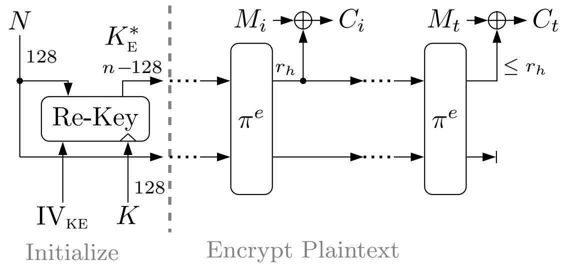

Figure 16: Encryption. N is nonce, and K is master key.

Absorbing randomness phase. At this phase, public and random parameters are available, but the main obstacle for the attacker is that the master key K is *well-mixed* before the start of this phase, in the sense that it is not possible to find any intermediate that is only based on a *fraction* of K. However, the attacker can target other secrets as $\hat{K} = \pi^a(K||IV)$ or the internal states of π. For these secrets, the requirement of DPA is met. For example, for $\hat{K}$, we see that $V = \hat{K}_o \oplus Y_1$, where V is the output of Xor operation, and Y_1 is 1-bit (with enough trailing zeros), and $\hat{K}_o$ is part of $\hat{K}$. In [10], it is argued that this combination with $b \geq 3$ is not enough for a successful DPA attack. Note that Y_1 has only two possible values, and $\hat{K}_o$ is 8-, 16-, 32-bit, or even more. In two of ISAP variants (see table 1), that are less conservative, b is set to one, and this is less than the recommended value in [10]. In our future works, we will consider power analysis impact of having $b = 1$.

Squeezing phase. This phase is also DPA-secure since no randomness is involved in its computations.

5.2.2 Attacks regarding Encryption

Encryption process is as in Figure 16. After computation of a fresh session key K_{E}^*, sponge is squeezed at rate r_h (see table 1), and the output bits are used to encrypt plaintext blocks M_i.

Encrypt plaintext phase. With the new nonce involvement in each Re-Keying invocation, the value of K_{E}^* is entirely fresh each time. Therefore, the requirement of DPA attacks that the targeted fraction of K_{E}^* should be constant during multiple executions of cipher is not satisfied. Also, note that the internal state of each π is wholly different for distinct nonce's.

5.2.3 Attacks regarding Authentication

In ISAP, to authenticate a tuple of ciphertext and associated data, a separate algorithm, independent of the Encryption, as depicted in Figure 17, is used. For a DPA attack, phases before *Finalize* are not helpful since no secret is involved. The *Finalize* itself is composed of Re-Keying with randomness Y and an extra permutation π^h. The final permutation π^h involves no randomness.

Usage of separate IVs for Encryption and Authentication makes states inside Re-Keying different for the two cases, even for the same input randomnesses.

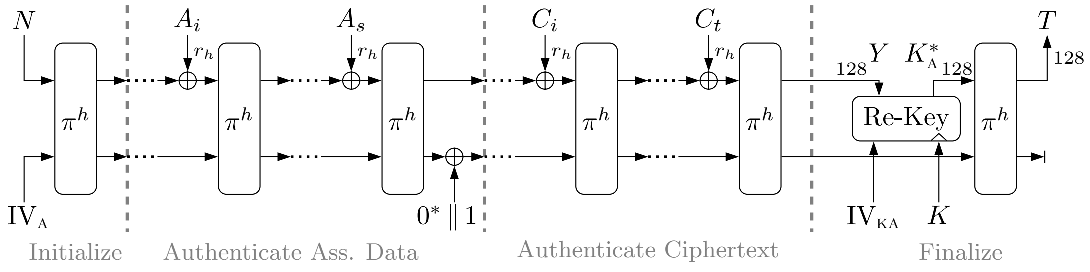

Figure 17: Authentication. A_is are blocks of the associated data, and T is the generated tag.

5.2.4 Attacks regarding Decryption

After receiving a tuple (N, A_is, C_is, T), first, the authenticity of tag T should be verified. If the tag is invalid, the receiver is not allowed to call the Decryption. However, if the tag is valid, the Decryption will be invoked. This algorithm works in the same way as Encryption; only the places of ciphertext and plaintext blocks are swapped. We have already discussed the impossibility of DPA-like attacks for Authentication and Encryption.

5.2.5 Attacks regarding Tag comparison

In [11], it is shown that naive comparison of a valid (secret) tag T with a random value T' can help an adversary to recover T with a DPA attack. If $T[i]$s are words of T, a direct (constant time) comparison of T and T' requires XORing corresponding $T[i]$ and $T'[i]$ for all is. This computation has all the requirements of a DPA attack. So, it can reveal secret values of $T[i]$ with the help of adequate collection of measurements done with random parameters $T'[i]$.

To prevent tag recovery, [12] suggested the usage of an extra permutation. In this way, to validate T', a comparison is made *only* on initial words[4] of $\pi^r(T)$ and $\pi^r(T')$ for some constant r. Note that *full* comparison helps the adversary to recover $\pi^r(T)$ with a DPA attack and hence obtain T since π^r is an easy-to-invert permutation (in ISAP). So, the comparison is only made on truncated parts of $\pi^r(T)$ and $\pi^r(T')$. Although the attacker will the learn value of the compared parts of $\pi^r(T)$, this knowledge is not enough to obtain T.

5.3 Soft-analytical attacks

When it is impossible to extract any fraction of the secret separately, the attacker can consider a more advanced and more complex class of attacks known as *soft-analytical attacks*. For this purpose, leakages of as many as possible variables are gathered and merged with the *belief propagation* method to recover the value of the secret. Belief prorogation is a commonly used tool in the *coding theory* for decoding a *code word* with an observation of its noisy variables. There, the belief propagation approach utilizes the algebraic interconnections (also known as *parity relations*)[5] of the variables of the code word. In a somewhat similar process, belief propagation can use the parity relations among the intermediate variables for combining their leakage information [13, 14].

[4]If word size is Byte, then the 16 initial Bytes are compared.

[5]For the best possible results, the parity relations should be *sparse*, as in the case of *LDPC* codes.

The soft-analytical attacks are successfully used in simulations for recovering the internal state of Keccak-p with $n = 1600$ [6] and word size of at most 16-bit [15]. For ISAP, the internal state size is $n = 400$, which is much less. In [16], larger word size as 32-bit are used for a soft-analytical attack against Keccak-p. Again, this work is considering only $n = 1600$. In [11], the authors successfully applied this attack for Keccak-p with $n = 400$ and word size of 16-bit on Cortex-M0, which is known for very low observation noise. It is currently unknown to what extent refinements to the belief propagation methods can be a threat in the moderate noise levels. However, it is commonly accepted that bigger word sizes as 32-bit is notably helping to the side-channel security of implementations. In the reminder of this section, we consider two points related to internal state recovery.

5.3.1 State recovery in the Re-Keying leads to full key recovery

Recovering the state of each permutation during Re-Keying directly leads to recovery of the master key K since the permutation π is easily convertible for both of the options.

5.3.2 Attacking key transfer

Since the schema is *unmasked*, the same K is always transferred from device memory for computation of the session key, which with enough averaging and precise profiling, can lead to recovery of K. A footprint of this effect is also evident in the following TVLA results.

Interpreting TVLA results. TVLA result for *fixed-key* versus *random-key* is plotted in Figure 18. For it, $100k$ traces for each case were collected where the message was 8-Byte random, and the associated data was 16-Byte random with the device operating at 100MHz. There are notable peaks in the TVLA result. Some of these peaks are because of key transfer. Note that the master key K is directly used in the Re-Keying, and both the Encryption and the Authentication call the Re-Keying.

Value recovery for Bytes. If word size is small enough (for example, if it is 8 bits), an attacker can create (in this example, 256) precise templates to determine the value of each word of the key K by averaging multiple noisy observations [17]. This approach does not rely on the parity relations of the intermediate variables. Creating these precise templates requires an exponential effort in the word size, so, for 32-bit implementations, this attack is almost not a practical threat.

5.3.3 Impact of word size and other parameters

We already discussed one impact of word size. Another consideration relevant to side-channel security is its connection to the available *quantization levels* in ADCs. Current trace measurement devices are equipped with 16-bit or less ADCs, which means that if only one sample in each clock cycle, i.e., single PoI[7], is gathered, it will contain at most 16 bits of information about a *word* that can be 32-bit or even more. This remark demonstrates some inherent security of using bigger word sizes in the implementations. To overcome the quantization limitation, the attacker can invoke the over-sampling technique by collecting more measurement points for each operation, which adds to the work's complexity.

[6]This state size is used in SHA3 hash function.
[7]Point of Interest

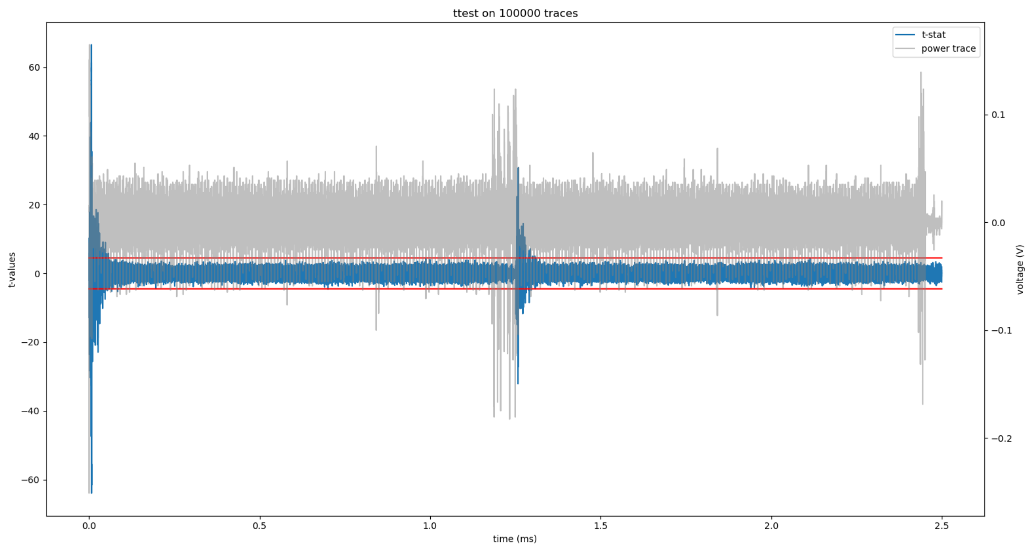

Figure 18: TVLA results for *fixed-key* vs. *random-key* for ISAP. Notable peaks are due to the *key* transfer at the start of Re-Keying.

5.4 Conclusion

Our discussions in this section show that DPA attacks are not an option for ISAP. The only exception is the tag comparison leakage, which should be understood well and defeated effectively. In the absence of DPA vulnerabilities, an attacker may opt to mount soft-analytical attacks that assume the strongest adversary. These attacks require a lot of effort in profiling and combining and are not so practical on actual devices. Anyhow, for soft-analytical attacks, currently, there is no guarantee that refined versions of templates or more complicated belief propagation methods such as *generalized belief propagation* cannot recover the internal states. Because of this, it is advisable to modify Re-Keying algorithm of ISAP to break the permutations chain that is leading *state recovery* to *master key recovery*.

References

[1] Gilbert Goodwill, Benjamin Jun, Joshua Jaffe, and Pankaj Rohatgi. A testing methodology for side channel resistance. 2011.

[2] Georg T. Becker, Jeremy Cooper, Elke DeMulder, Gilbert Goodwill, Joshua Jaffe, Gary Kenworthy, Timofei Kouzminov, Andrew J. Leiserson, Mark E. Marson, Pankaj Rohatgi, and Sami Saab. Test vector leakage assessment (TVLA) methodology in practice. International Cryptographic Module Conference, 2013.

[3] 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*, volume 3156, pages 16–29, 2004.

[4] Niels Samwel and Joan Daemen. DPA on hardware implementations of Ascon and Keyak. In *Proceedings of the Computing Frontiers Conference*, pages 415–424, Siena Italy, May 2017. ACM.

[5] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon v1.2. Submission to Round 1 of the NIST Lightweight Cryptography project, 2019.

[6] Ascon. Ascon/ascon-c: Ascon - lightweight authenticated encryption and hashing. `https://github.com/ascon/ascon-c`.

[7] E. Prouff, M. Rivain, and R. Bevan. Statistical Analysis of Second Order Differential Power Analysis. *IEEE Transactions on Computers*, 58(6):799–811, June 2009.

[8] XKCP. XKCP / XKCP: the eXtended Keccak Code Package. `https://github.com/XKCP/XKCP`.

[9] Christoph Dobraunig, Maria Eichlseder, S.Stefan Mangard, Florian Mendel, Bart Mennink, Robert Primas, and Thomas Unterluggauer. Isap v2.0. submission to the NIST lightweight cryptography standardization effort, 2019.

[10] Mostafa Taha and Patrick Schaumont. Side-channel countermeasure for sha-3 at almost-zero area overhead. In *2014 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pages 93–96, 2014.

[11] 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. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020*, pages 369–400, Cham, 2020. Springer International Publishing.

[12] Christoph Dobraunig and Bart Mennink. Leakage resilient value comparison with application to message authentication. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021*, pages 377–407, Cham, 2021. Springer International Publishing.

[13] Nicolas Veyrat-Charvillon, Benoît Gérard, and François-Xavier Standaert. Soft analytical side-channel attacks. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014*, pages 282–296, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[14] Qian Guo, Vincent Grosso, and François-Xavier Standaert. Modeling soft analytical side-channel attacks from a coding theory viewpoint. *IACR Cryptol. ePrint Arch.*, 2018:498, 2018.

[15] Matthias J Kannwischer, Peter Pessl, and Robert Primas. Single-trace attacks on keccak. *Cryptology ePrint Archive*, 2020.

[16] Shih-Chun You and Markus G. Kuhn. Single-trace fragment template attack on a 32-bit implementation of keccak. In Vincent Grosso and Thomas Pöppelmann, editors, *Smart Card Research and Advanced Applications*, pages 3–23, Cham, 2022. Springer International Publishing.

[17] Marios O. Choudary and Markus G. Kuhn. Efficient stochastic methods: Profiled attacks beyond 8 bits. In Marc Joye and Amir Moradi, editors, *Smart Card Research and Advanced Applications*, pages 85–103, Cham, 2015. Springer International Publishing.