Building a Side-Channel Attack Scheme on SipHash FPGA Implementation

Vít Mašek, Vojtěch Miškovský, Matúš Olekšák

FIT CTU in Prague
PESW 2025

June 28, 2025

Acknowledgment

This work was supported by the Czech Technical University (CTU) grant No. SGS23/208/OHK3/3T/18, by the Student Summer Research Program 2024 of FIT CTU in Prague, and the grant VJ02010010 of the Ministry of the Interior of the Czech Republic, “Tools for AI-enhanced Security Verification of Cryptographic Devices” in the program Impakt1 (2022-2025).

Outline

- Introduction to DPA
- Introduction of SipHash MAC function
- Presentation of the proposed attack
  - Analyzing SipRound function
  - Using simplified models
  - Using real traces

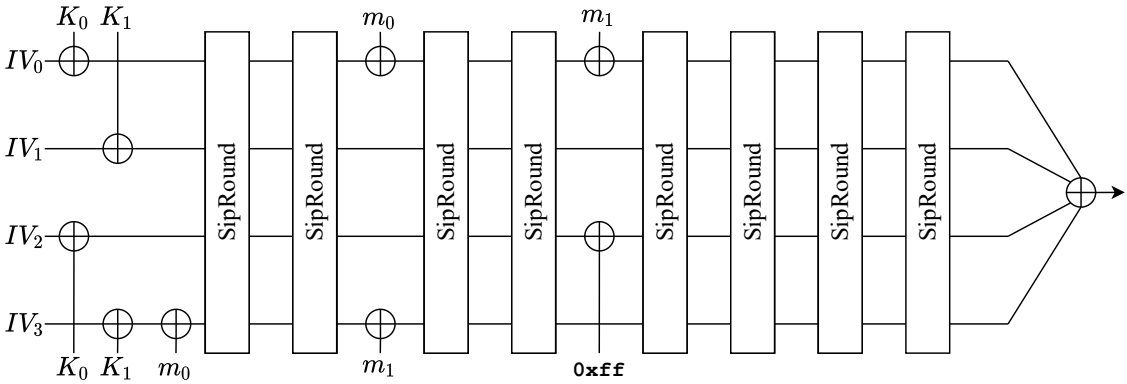
Introduction to DPA

1. Measure a set of measurements O for different inputs X_i.
2. Assume a single-bit leakage function $\hat{L}(k, X) \in \{0, 1\}$.
3. Enumerate all subkey guesses $\hat{k}$.
4. For each guess $\hat{k}$, partition the measurements $o_{X_i} \in O$ into two partitions, based on $\hat{L}(\hat{k}, o_{X_i})$:

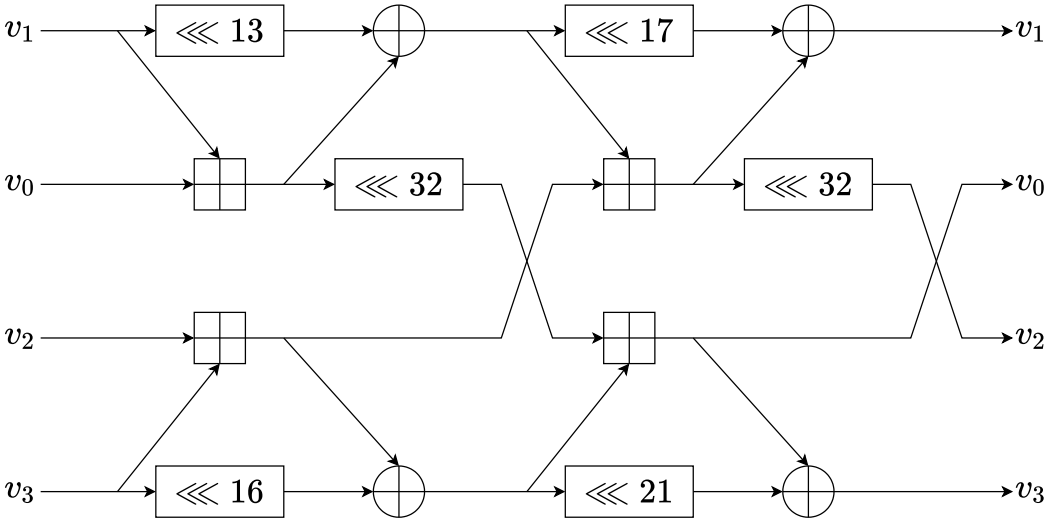
   $$O_0^{\hat{k}} = \{o_{X_i} | \hat{L}(\hat{k}, X_i) = 0\}, \quad O_1^{\hat{k}} = \{o_{X_i} | \hat{L}(\hat{k}, X_i) = 1\}$$

   and compute the average power consumption in each time point for both partitions.
5. Select the guess $\hat{k}$, for which the absolute difference of means (DoM) across all time points is the highest.

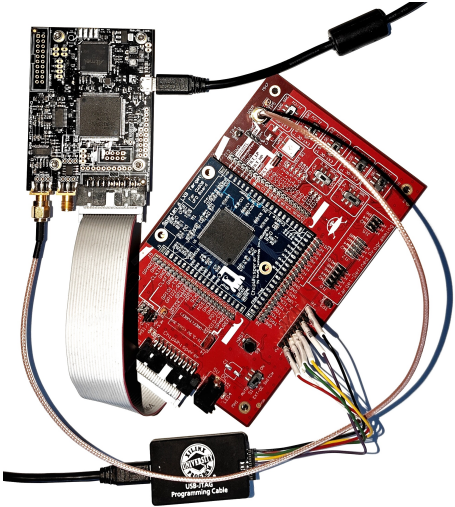
SipHash


K_0 K_1
IV_0
IV_1
IV_2
IV_3
K_0 K_1 m_0
SipRound
SipRound
m_0
m_1
SipRound
SipRound
m_1
0xff
SipRound
SipRound
SipRound
SipRound

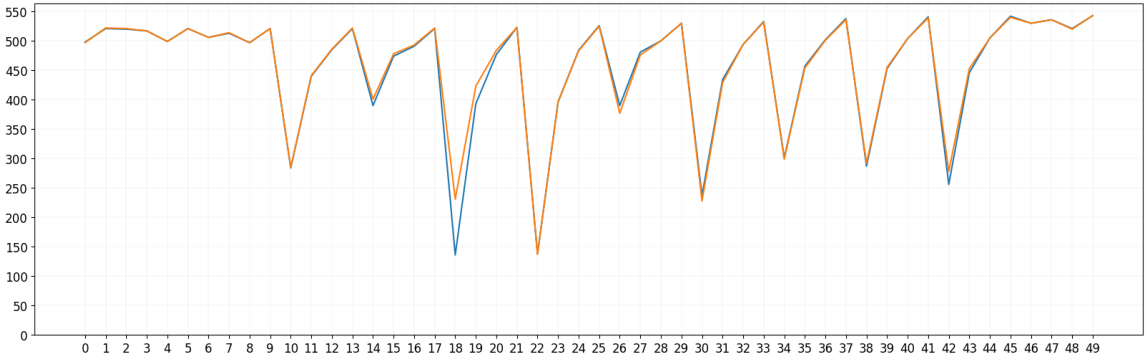
SipHash – SipRound



ChipWhisperer CW308 Setup

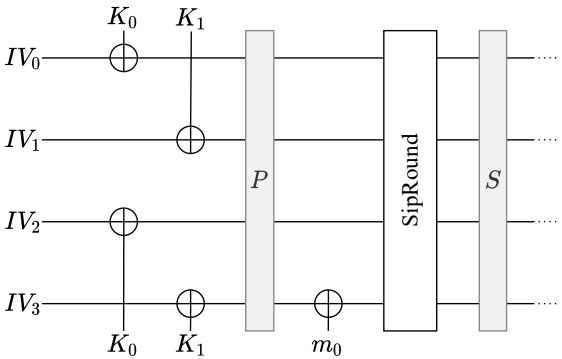


Traces


550
500
450
400
350
300
250
200
150
100
50
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49

Attack

Focus on the result after the first SipRound (S)



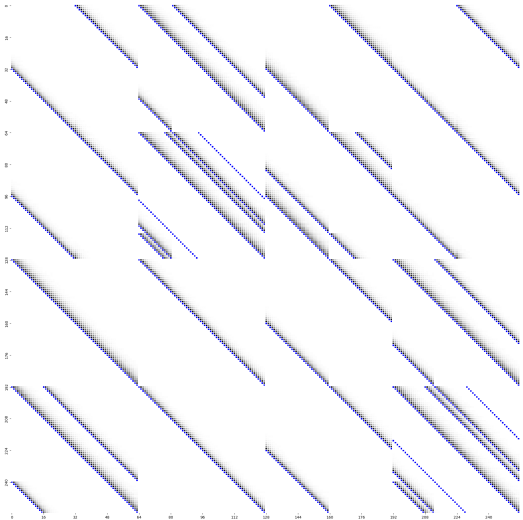
PROBLEM: How do we compute a bit S_i from a feasible number of bits of key K?

Attack – Finding “Weak Bits”

Lets analyze the SipRound.

1. Generate random state P
2. Flip bit $P_i \rightarrow \overline{P}$
3. Compute $S = \text{SipRound}(P)$ and $\overline{S} = \text{SipRound}(\overline{P})$
4. Observe difference of S and $\overline{S} \rightarrow S \oplus \overline{S}$
5. Do this for all 256 bits many times
6. Observe the probability $(S \oplus \overline{S})_i = 1$

Attack – Finding “Weak Bits”



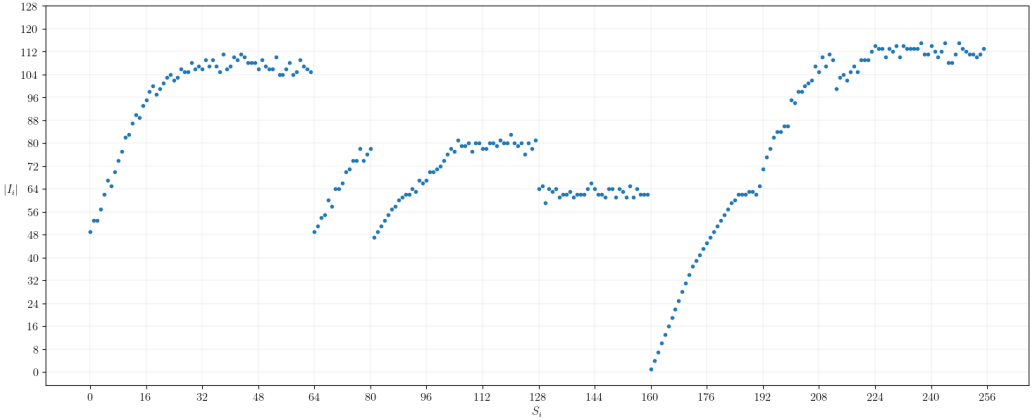
Attack – Finding “Weak Bits”

FINDING: Flipping a single bit of P affects the result S only in a few bits.

Lets do similar thing, but use the key K instate of state P.

1. Generate random key K
2. Flip bit $K_b \rightarrow \overline{K}$
3. Compute the results S and $\overline{S}$ when $m_0 = 0...0$
4. If $(S \oplus \overline{S})_i = 1$, put b to a set I_i
5. Do this across $b \in [0, 127]$ and $i \in [0, 255]$ many times

Attack – Finding “Weak Bits”



FINDING: To compute bits of S starting from $i = 160$, we need to know only a few bits of K.

Attack – Finding “Weak Bits”

i	$\lvert I_i \rvert$	I_i
160	1	$\{115\}$
161	4	$\{0, 64, 115, 116\}$
162	7	$\{0, 1, 64, 65, 115, 116, 117\}$
163	10	$\{0, 1, 2, 64, 65, 66, 115, 116, 117, 118\}$
164	13	$\{0, 1, 2, 3, 64, 65, 66, 67, 115, 116, 117, 118, 119\}$

Attack – Success Rate Model

Leakage function

$$L_{hw}(k_{l_i}, m_0, i) = \text{SipRound}(IV \oplus k_{l_i} \oplus m_0)_i$$

Measurement function (model)

$$O_{hw}(K, m_0, i) = \text{SipRound}(IV \oplus K \oplus m_0)_i$$

Success rate

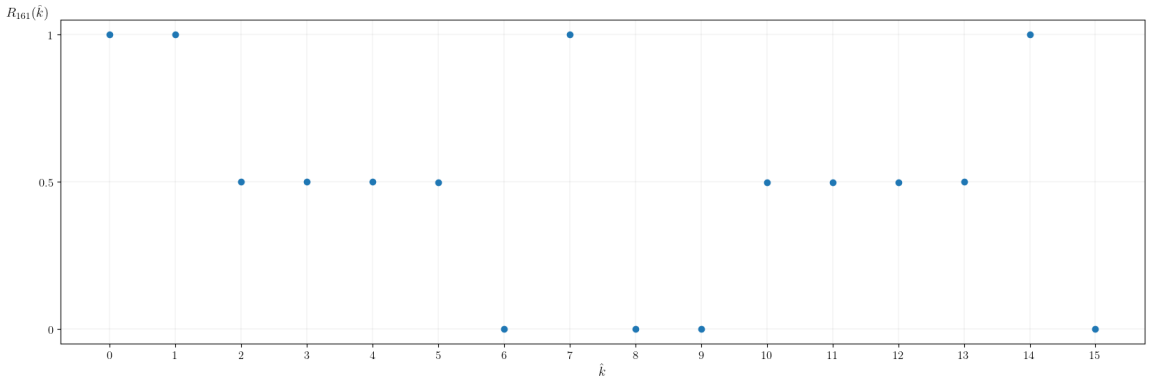
$$R_i(k_{l_i}) = \frac{|M_{eq}|}{|M|}$$

where

$$M_{eq} = \{m_0 | O_{hw}(K, m_0, i) == L_{hw}(k_{l_i}, m_0, i)\}$$

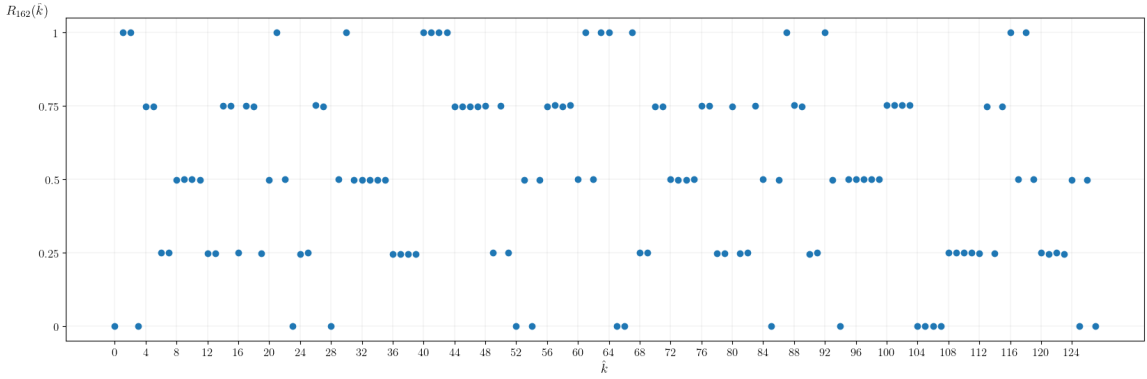
Attack – Success Rate Model – bit $i = 161$

$|I_{161}| = 4$



Attack – Success Rate Model – bit $i = 162$

$|I_{162}| = 7$



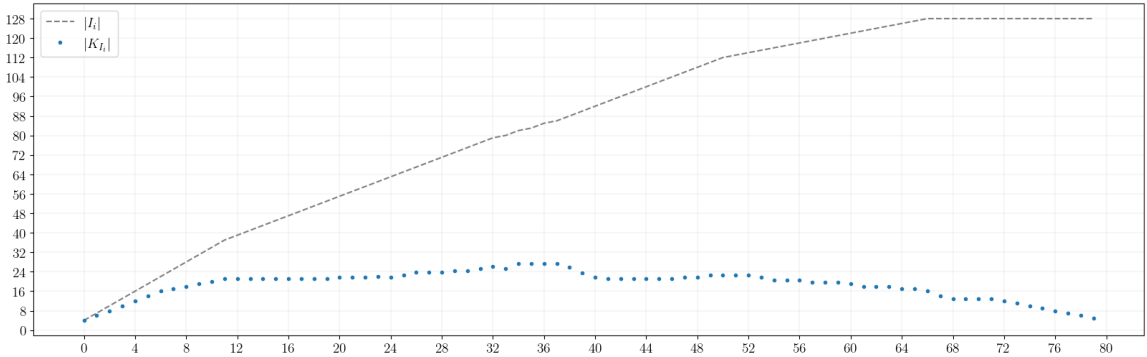
Attack – Success Rate Model – Iterative Approach

IDEA: Use results from bits i for which I_i is small, to eliminate some portion of keys to be able to attack also bits for which I_i is big.

1. Start with bit $j = 161$
2. Attack, get $\widehat{K}_{I_j}$ as the keys for which $R = 1$ or 0.
3. Find next bit i, for which $|I_i \setminus I_j|$ is small.
4. Extend the subkey space with the new key bits $I_i \setminus I_j$
5. Attack, get $\widehat{K}_{I_i}$
6. Continue with step 3

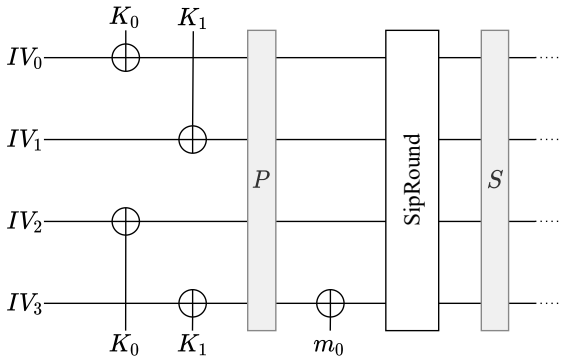
Attack – Success Rate Model – Results

Maximum of keys in single iteration: $2^{27.3}$
Total key hypotheses: 2^{30}



Attack – HD vs. HW

Compute $HD(P_i, S_i) = P_i \oplus S_i$
To attack bit i, we need to compute not only S_i, but also P_i.



PROBLEM: We need to know bit of K to correctly compute P_i.

Attack – HD vs. HW

IDEA: The value of P depends only on IV and $K \Rightarrow$ constant

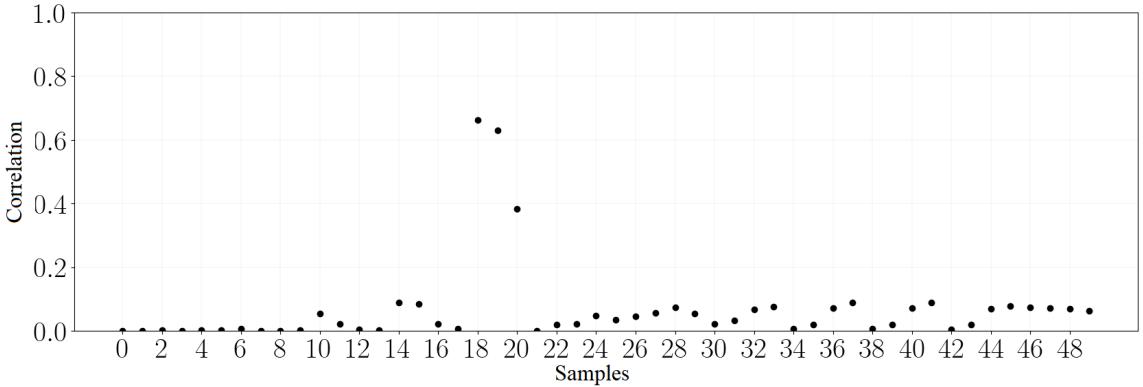
SOLUTION: Just put $P_i = 0$.

1. Real $P_i = 0$: All nice and sunflowers $\Rightarrow R = 1$
2. Real $P_i = 1$: Real $HD = (1 \oplus S_i) = \overline{S_i} \Rightarrow R = 0$

RESULT: Since we have to accept $R = 0$ too, the HD model is equivalent to HW model on single bit.

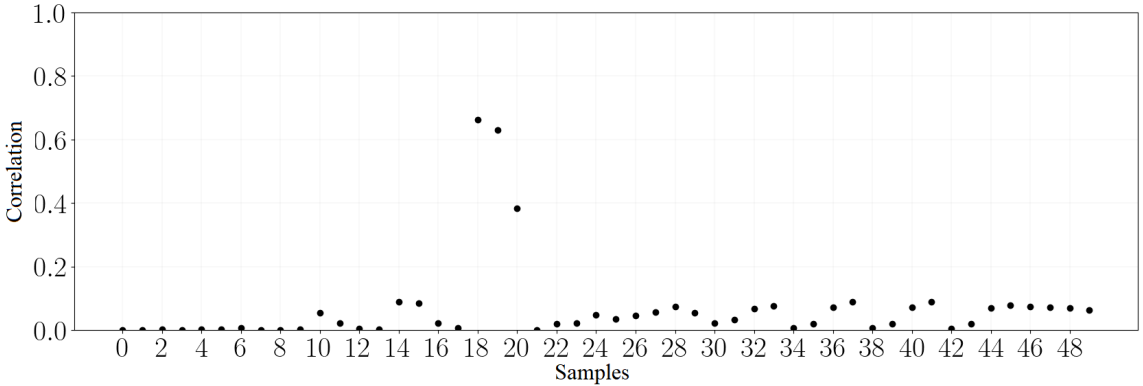
Attack – Full State Model

Model the power consumption as $HD(P, S)$.
(correlation with real measurements > 0.6)
Apply classic DPA attack and our iterative approach.

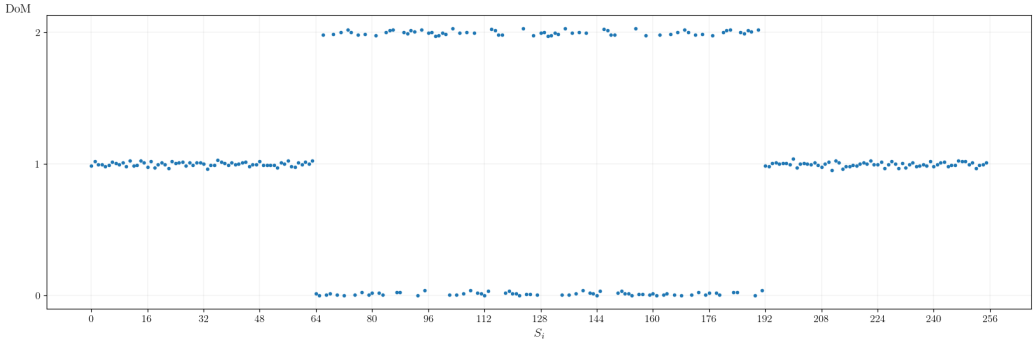


Attack – Full State Model

Model the power consumption as $HD(P, S)$.
(correlation with real measurements > 0.6)
Apply classic DPA attack and our iterative approach.
RESULT: It does not work.



Attack – Full State Model – Plot the DoMs

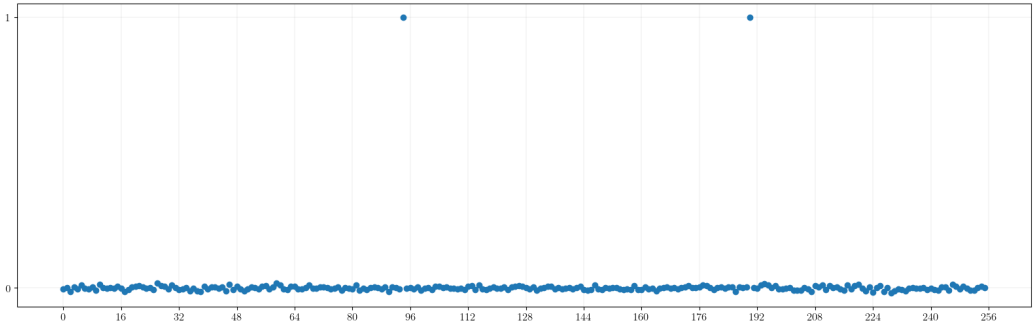


FINDING: The DoM converge to 0 or 2 for some bits.
(expected to be 1)

IDEA: Check for correlation between the bits.

Attack – Full State Model – “bad bits”

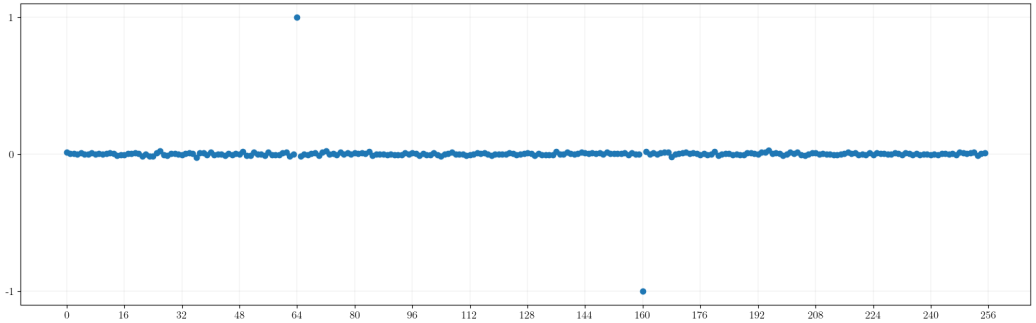
FINDING: The values $HD(P_i, S_i)$ and $HD(P_j, S_j)$ is correlated for some bits (i, j).



Positive correlation $\Rightarrow$ support each other in the leakage for DPA.

Attack – Full State Model – “bad bits”

FINDING: The values $HD(P_i, S_i)$ and $HD(P_j, S_j)$ is correlated for some bits (i, j).



Negative correlation $\Rightarrow$ mask each other in the leakage for DPA.

Attack – Full State Model – “bad bits”

PROBLEM: How two bits (i, j) are correlated depends on the secret key $\Rightarrow$ we do not know which bits leak and which do not.

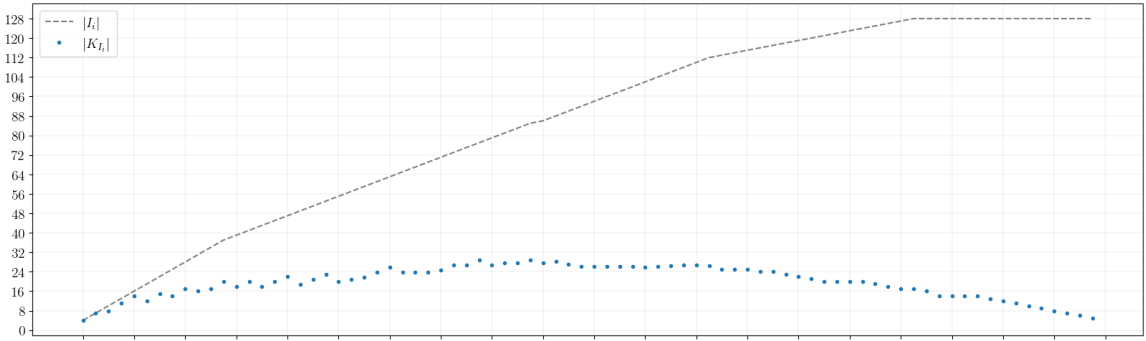
SOLUTION: Use Welch’s t-test to validate the results of each iteration.

At the end of each iteration:

1. Select one key from the results (highest absolute DoM).
2. Partition the traces as in DPA and compute the t-statistic.
3. If the t-statistic is below a critical value, scrap the iteration and find new bit i to attack.

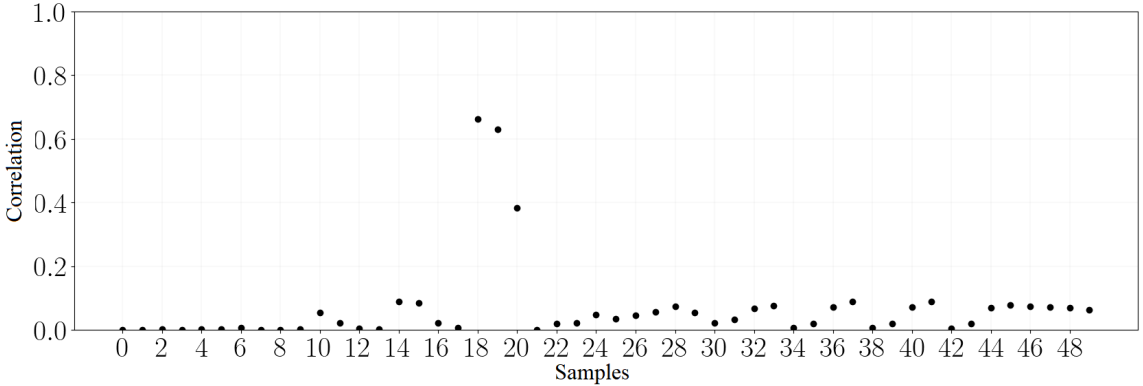
Attack – Full State Model – Results

Maximum of keys in single iteration: $2^{28.7}$
Total key hypotheses: $2^{31.7}$
Traces (modeled as $HD(P, S)$): 8.000



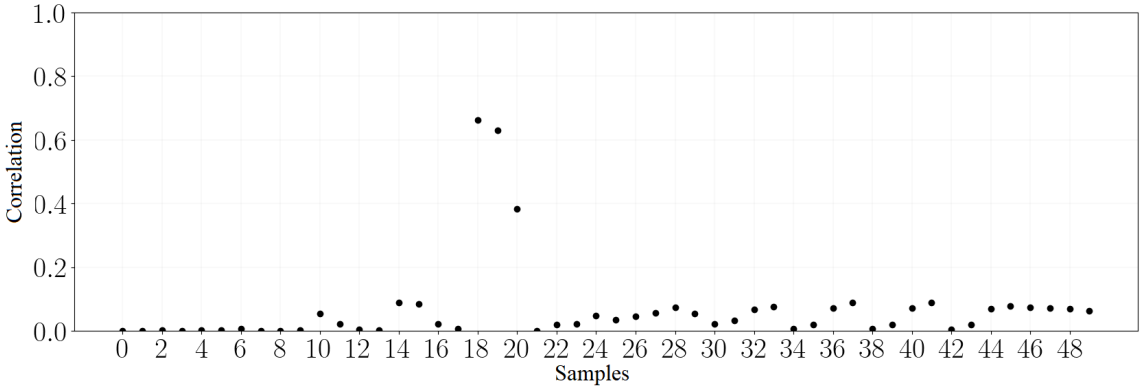
Attack – Real Traces

IDEA: Because there is a high correlation between the modeled and real traces.
⇒ We shall be able to just replace the model.



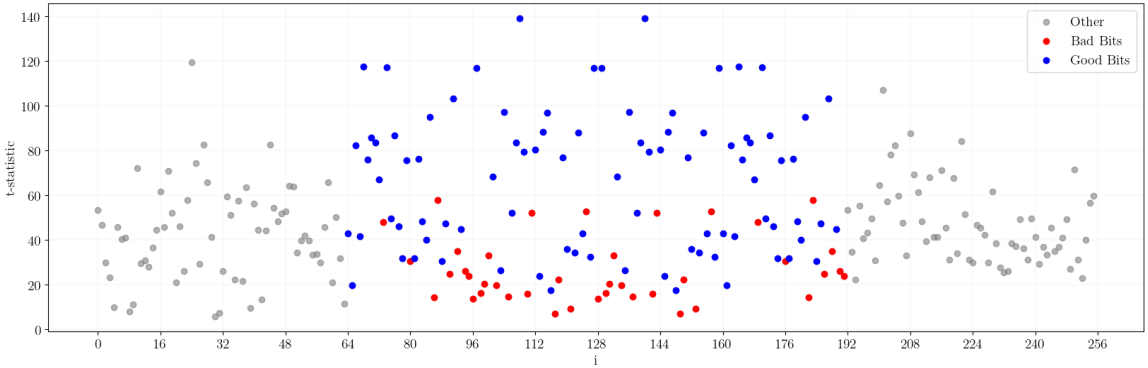
Attack – Real Traces

IDEA: Because there is a high correlation between the modeled and real traces.
$\Rightarrow$ We shall be able to just replace the model.



RESULT: It does not work... Again...

Attack – Real Traces – “bad bits”



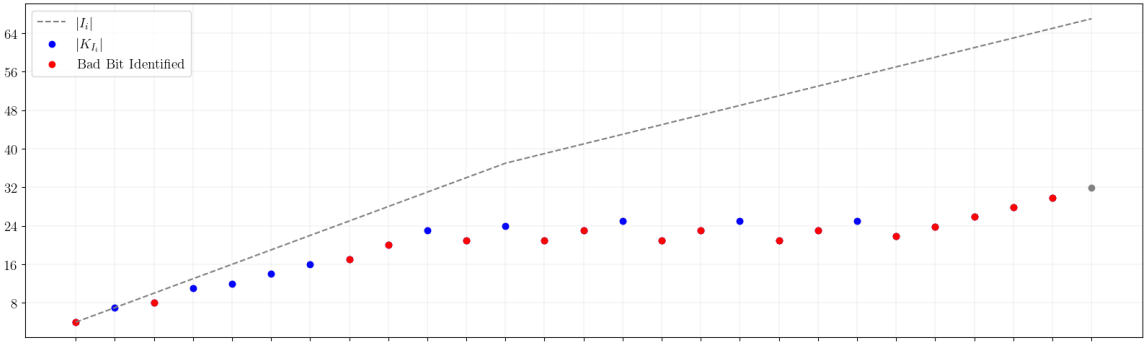
Attack – Real Traces – Adjustment

SOLUTION: Adjust the critical value for the t-test, to ensure we do not accept a bad iteration.

PROBLEM: By this adjustment, also we do not accept some good iterations.

Attack – Real Traces – Results

Traces: 100.000
Stopped when the number of keys in iteration reached $2^{31.87}$
The total key space at the time of the last successful iteration: 2^{93}



Conclusion

The proposed attack method retrieves the SipHash key based on Hamming distance of the initial state and the result of the first SipRound.

Full key recovery using real-world traces was not fully successful due to:

- Inefficient identification of the “bad bits”
- Big computational complexity
- Possibly not the best quality of traces

Despite that, the results challenge the widely held assumption that ARX designs are inherently resistant to DPA style of attacks.

Full paper will be presented at the DSD 2025 conference in September.