

# A FINER-GRAIN ANALYSIS OF THE (NON) LEAKAGE RESILIENCE OF OCB

Francesco Berti, Shivam Bhasin, Jakub Breier, Xiaolu Hou, Romain Poussier\*  
François-Xavier Standaert et Balasz Udvarhelyi

Université Catholique de Louvain (Belgium)  
Nanyang Technological University (Singapore)

\*Now working at ANSSI (France)

# OUTLINE

- **Introduction and trivial attack paths**
  - OCB
  - Trivial attacks
- **First level of protection**
  - Baseline second-order attack
  - Improved attack
- **Second level of protection**
  - SPA on whitening values
  - Breaking confidentiality and integrity
- **Conclusion**

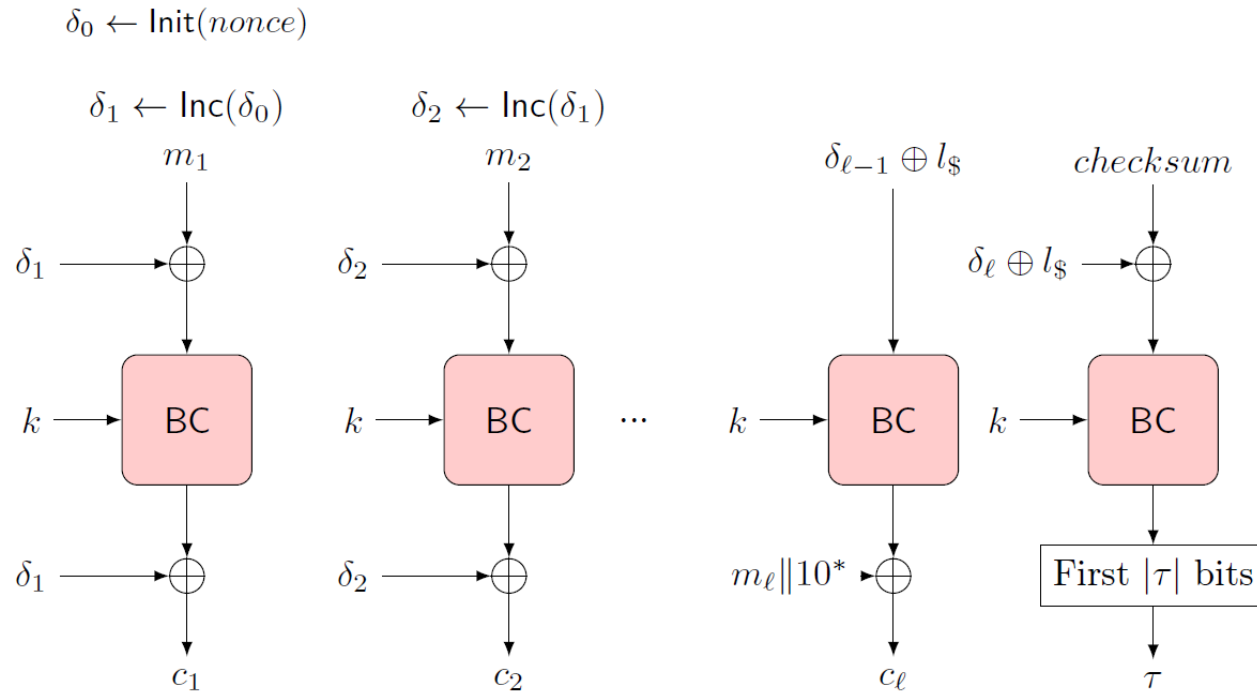
# OCB

- OCBv3 is an authenticated mode of encryption, finalist of the NIST CAESAR competition. It does not provide misuse resistance.
- By design, it does not claim any inherent protection against side-channel attacks, suggesting that all part of OCB should be implemented using strong SCA countermeasures. However, targetting a mode of encryption can be more complex than directly attacking the primitive itself.
- This study: finer grain analysis of the whole OCB mode against SCA, exhibiting the different attack paths and potentially mitigate the requirement to strongly protect all parts of OCB.

**DPA:** any side-channel attack making use of varying plaintext value (kocher's DPA, CPA, MIA, template, ML...)

**SPA:** any side-channel attack making use of a single plaintext value, potentially repeated (template, ML...)

# OCB OVERVIEW



Two functions **init**  
and **inc**

Padding for last  
message if  
incomplete

Checksum is the  
sum of the  
messages

The Associated Data (AD) is processed similarly replacing the message blocks by the AD ones. However, it uses  $\delta_0 = 0$

# INITIALIZATION

## 1. **Init.** Computation of $\delta_0 \simeq BC_k(\text{nonce})$

- Only unknown is the master key  $k$ ,  $\text{nonce}$  is known

## 2. **Inc.** Computation of $\delta_i$

- $l_0$  only depends on the master key  $k$
- **Double** is the multiplication by two over  $\mathbb{F}_{256}$
- $\text{ntz}(i) =$  the number of remaining (LSBs) 0 in the decomposition of  $i$  in base 2

---

**Algorithm 2** Schematic of OCB initialization and offset updating.

---

- 1:  $l_* \leftarrow BC_k(0)$
  - 2:  $l_\$ \leftarrow \text{double}(l_*)$
  - 3:  $l_0 \leftarrow \text{double}(l_\$)$
  - 4:  $l_i \leftarrow \text{double}(l_{i-1})$ , for  $i \geq 2$
  - 5:  $\delta_1 \leftarrow \llcorner_L \delta_0 \oplus l_0$
  - 6:  $\delta_2 \leftarrow \llcorner_L \delta_1 \oplus l_1$
  - 7:  $\delta_3 \leftarrow \llcorner_L \delta_2 \oplus l_0$
  - 8:  $\delta_4 \leftarrow \llcorner_L \delta_3 \oplus l_2$
  - 9: ...
  - 10:  $\delta_i \leftarrow \llcorner_L \delta_{i-1} \oplus l_{\text{ntz}(i)}$
-

# TRIVIAL ATTACKS

While OCB does not claim any protection against SCAs, attacking a mode can be harder than directly attacking the primitive itself (AES in this case).

We denote by **trivial** any attack on OCB that is « equivalent » to directly attacking the primitive

- 1. DPA against OCB decryption:** in this case, the adversary is able to query OCB-AES with the same nonce several times.  $\delta_1$  would thus be fixed, and the attacker targets the first decryption block  $c_1$ 
  1. Schoolbook DPA on the first AES round using  $k \oplus \delta_1$  as *key material* to recover
  2. Predict the state value at the second round, and target the second AES round key

# TRIVIAL ATTACKS

While OCB does not claim any protection against SCAs, attacking a mode can be harder than directly attacking the primitive itself (AES in this case).

We denote by **trivial** any attack on OCB that is « equivalent » to directly attacking the primitive

- 2. DPA against OCB initialization:** during the initialization procedure, it computes  $AES_k(\textit{nonce})$  where *nonce* is known. This is a trivial known plaintext SCA scenario on the primitive
- 3. DPA against the associated data:** when processing AD,  $\delta_1$  is always unknown but fixed. One can thus apply the same attack as for the decryption case (first  $k \oplus \delta_1$  then  $k$ )

# TRIVIAL ATTACKS

While OCB does not claim any protection against SCAs, attacking a mode can be harder than directly attacking the primitive itself (AES in this case).

We denote by **trivial** any attack on OCB that is « equivalent » to directly attacking the primitive

- 4. DPA against incomplete messages:** the last ciphertext block, if incomplete, is computed as  $c_i = AES_k(\delta_i) \oplus m_i$ . This is a trivial known ciphertext attack scenario, as  $\delta_i$  is not added before outputting the ciphertext



# OUTLINE

- Introduction and trivial attack paths
  - OCB
  - Trivial attacks
- **First level of protection**
  - **Baseline second-order attack**
  - Improved attack
- **Second level of protection**
  - SPA on whitening values
  - Breaking confidentiality and integrity
- Conclusion

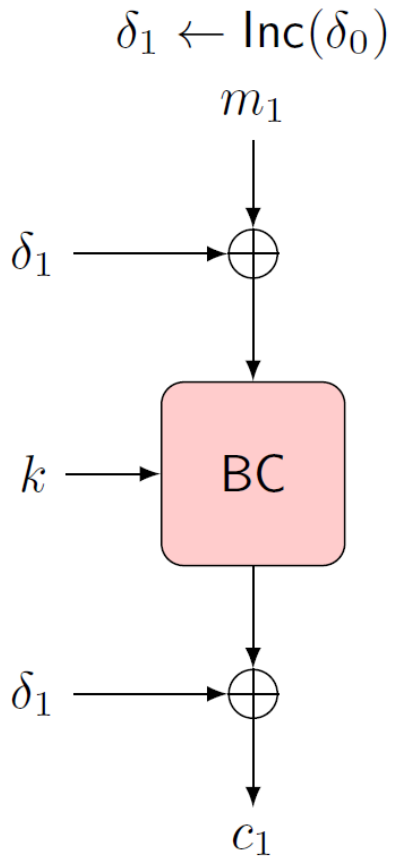
# PROTECTION LEVEL 1

Having shown some trivial attack paths, we assume that the designer has strongly protected the corresponding area with SCA countermeasures. We thus now ignore the leakages corresponding to:

- OCB decryptions
- AD processing
- The initialization procedure
- The processing of last message block (if incomplete)

The adversary is now left with leakages corresponding to the main part of the OCB encryption

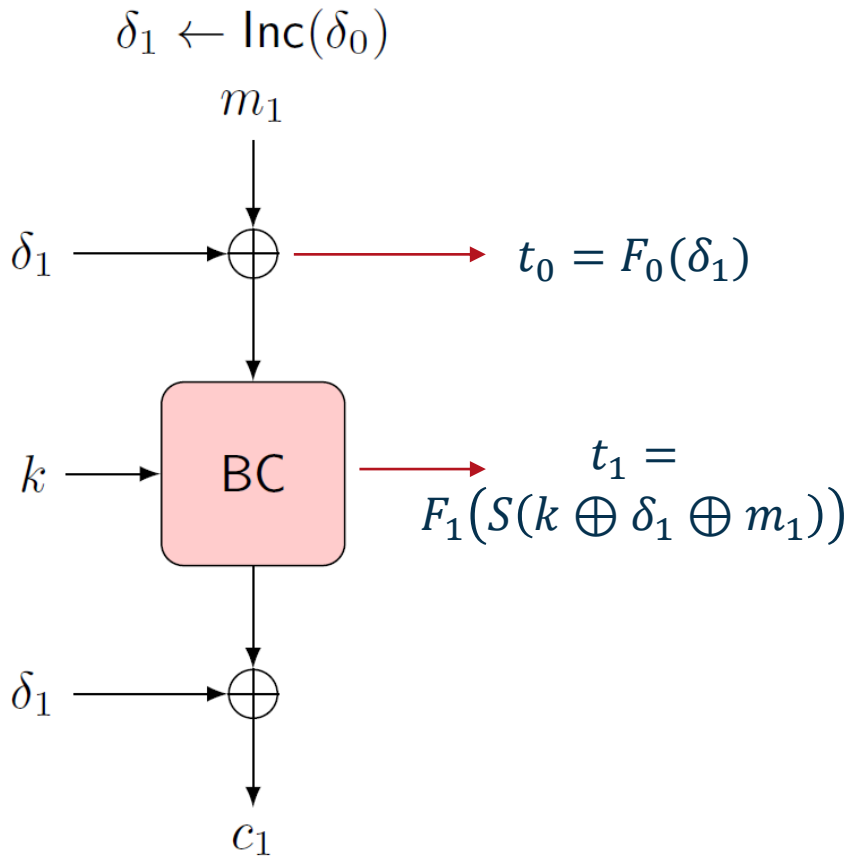
# AVAILABLE LEAKAGES



The available leakages are similar to the decryption or AD processing case, except for the fact that OCB is not misuse resistant

The nonce changes at each encryption, and so does  $\delta_i$ . The « message » input is unknown

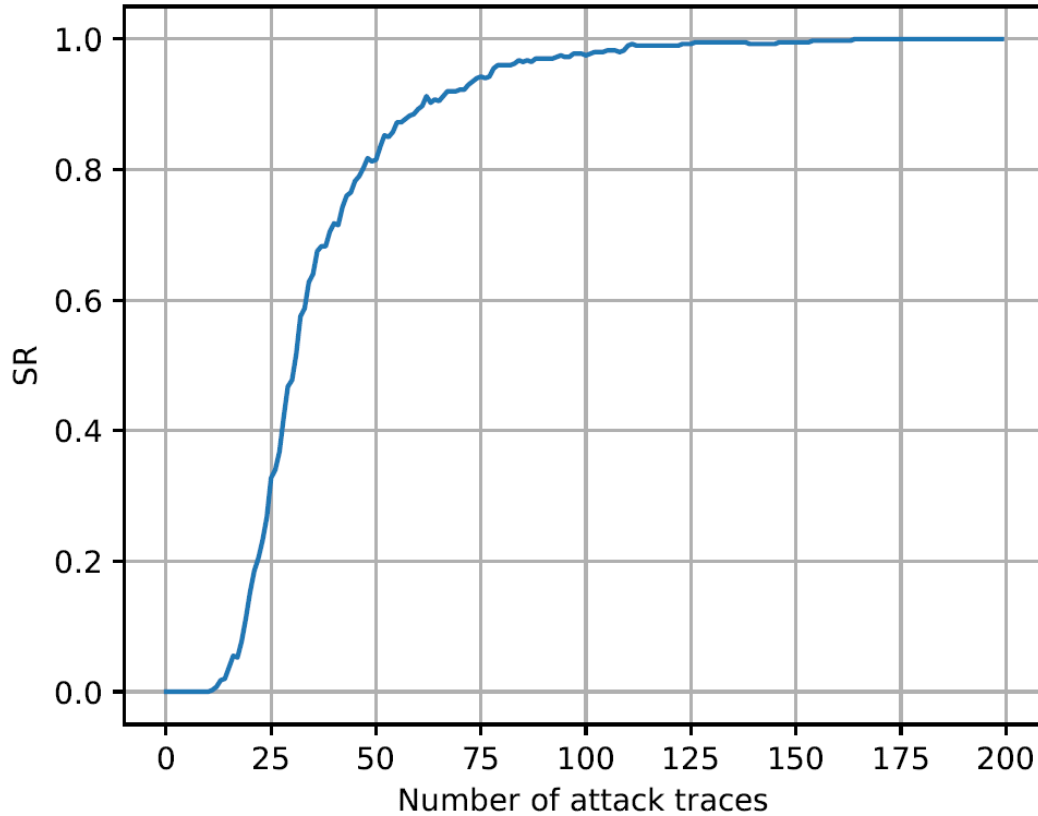
# AVAILABLE LEAKAGES



Idea 1: apply a second-order attack, considering as if  $\delta_1$  would be a *mask*

Use a leakage on  $\delta_1$  and on the Sbox output of the first AES round.

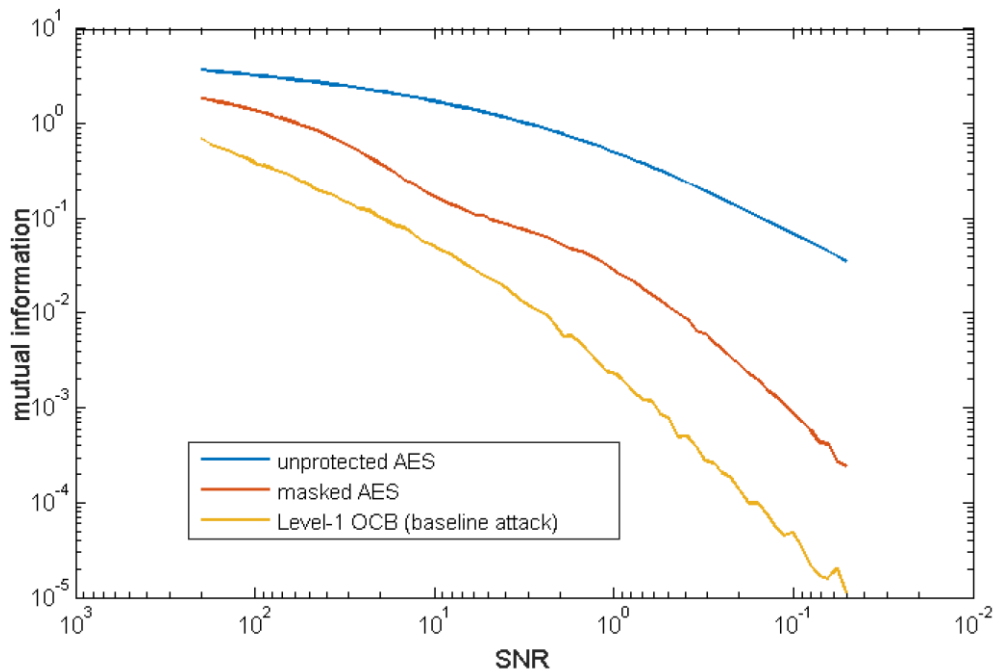
# BASELINE ATTACK RESULT



- Cortex-m0
- SNR, LDA then template attack
- Profiling: 800K traces
- Attack: 20 plaintexts of 80.000 128-bit blocs each (total 1.6M AES traces)

# LINK WITH MASKING: INFORMATION THEORY ANALYSIS

Simulations



- Red and orange curve have the same slope for high noise levels, confirming that it exploits a statistical moment of order 2 as for a 1<sup>st</sup> order masking scheme.
- However, the vertical offset, showing that it is harder to attack OCB than a regular masked AES (required traces  $\simeq \frac{c}{MI}$ )
- Can be seen as more algebraic complexity as for inner product masking:

$$\text{Masked aes: } S(k \oplus m_1) \oplus \delta_1$$

$$\text{baseline: } S(k \oplus \delta_1 \oplus m_1)$$

# OUTLINE

- Introduction and trivial attack paths
  - OCB
  - Trivial attacks
- **First level of protection**
  - Baseline second-order attack
  - **Improved attack**
- Second level of protection
  - SPA on whitening values
  - Breaking confidentiality and integrity
- Conclusion

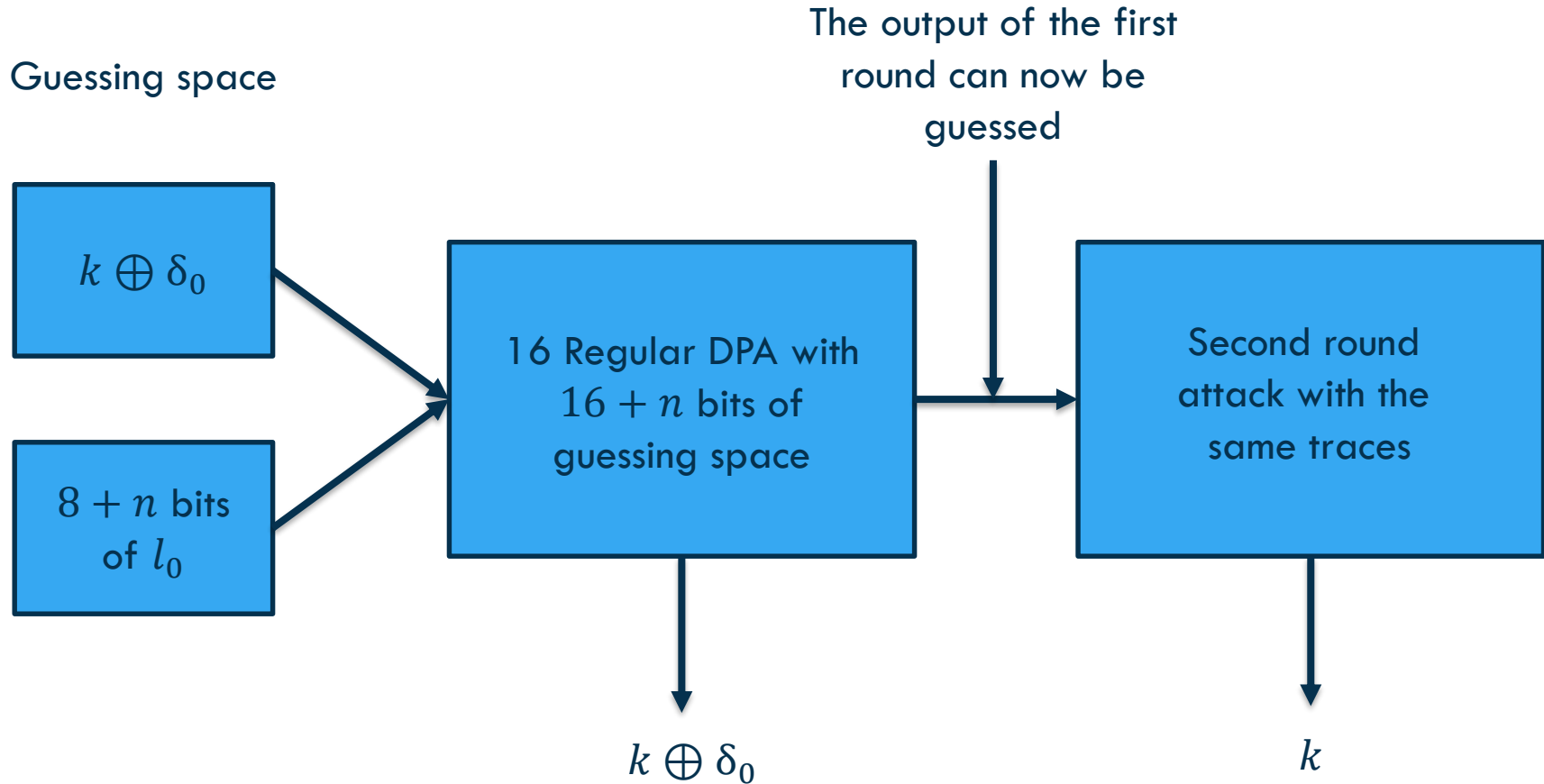
# SOME THOUGHTS ON OCB SECURITY

At first glance, it seems that the inner part of OCB is inherently protected against SCA by implementing some kind of free first-order masking

The baseline attack is thus an « easy » path to defeat OCB, but suffers from the same asymptotic trace complexity as when attacking a first-order masked implementation. For that reason, it is not efficient for high-noise scenario

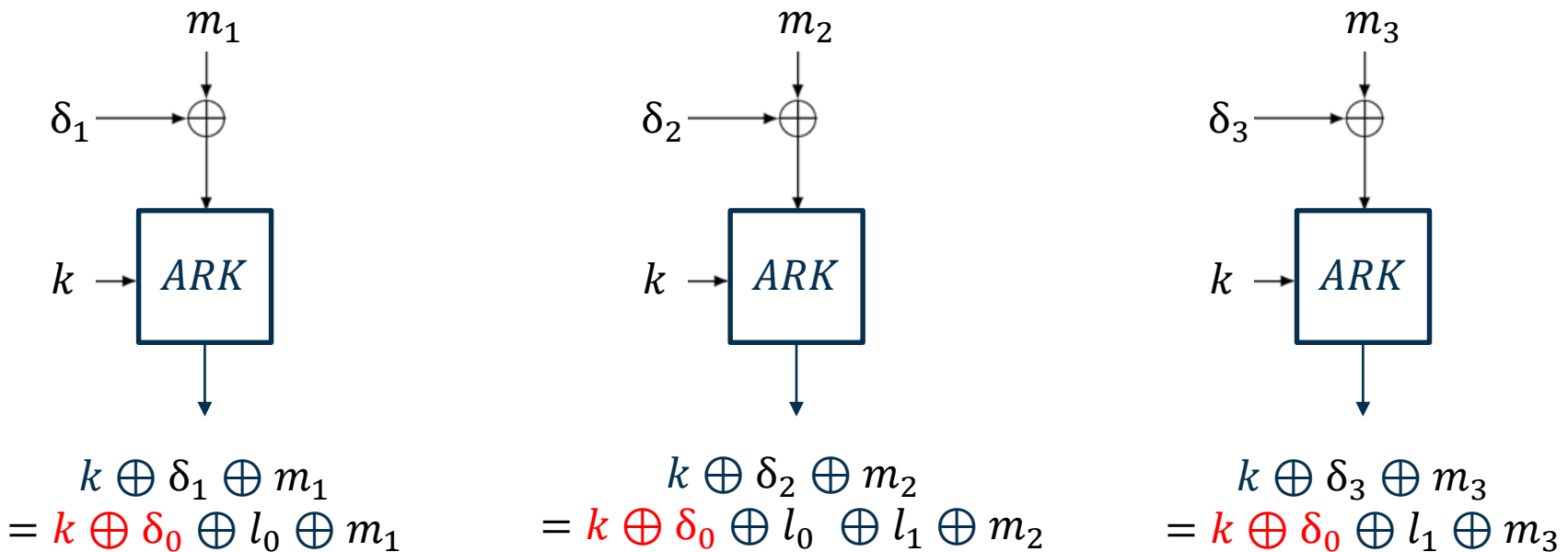


# IMPROVED ATTACK OVERVIEW



## INFORMATION FROM WHITENING VALUES

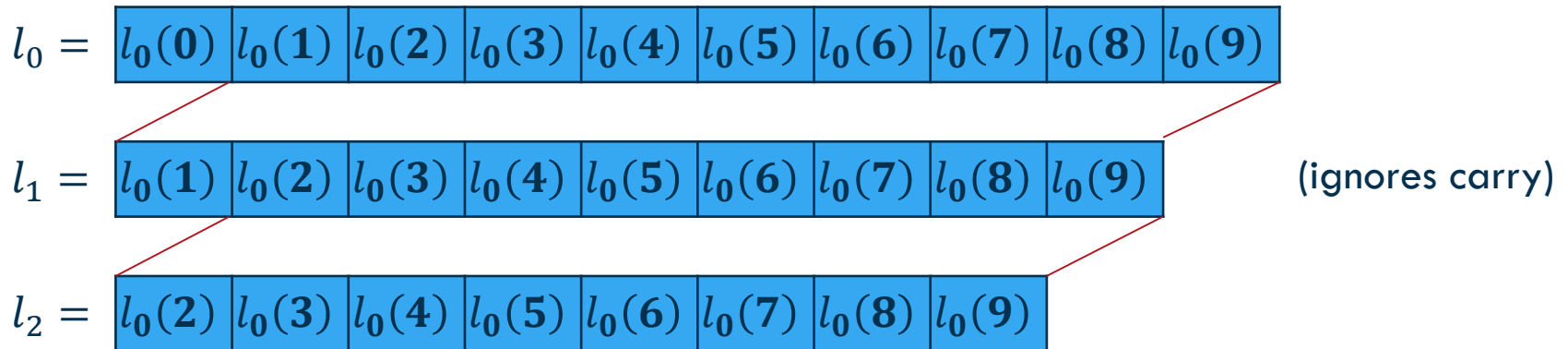
The baseline attack only considers the key as *guessing space*. We can design a more efficient « horizontal » attack by including bits of  $l_i$  into the guessing space.



If we know the  $l_i$ , the attacker can target  $k \oplus \delta_0$  as the new « key material »

# GUESSING WHITENING VALUES

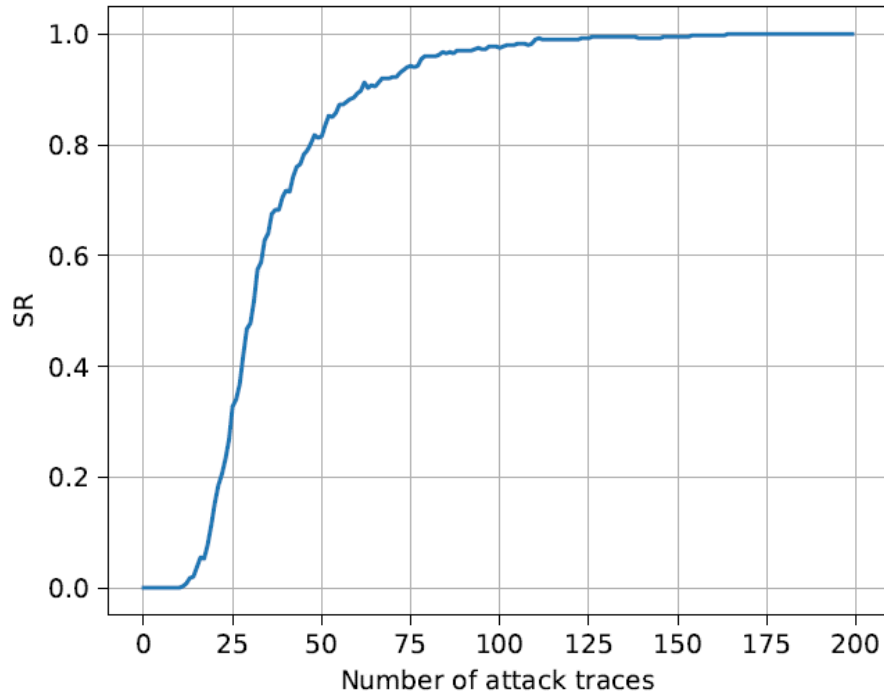
- $\delta_i = \delta_{i-1} \oplus l_{ntz(i)}$ : The knowledge of  $l_i$  translates into the knowledge of  $\delta_i$



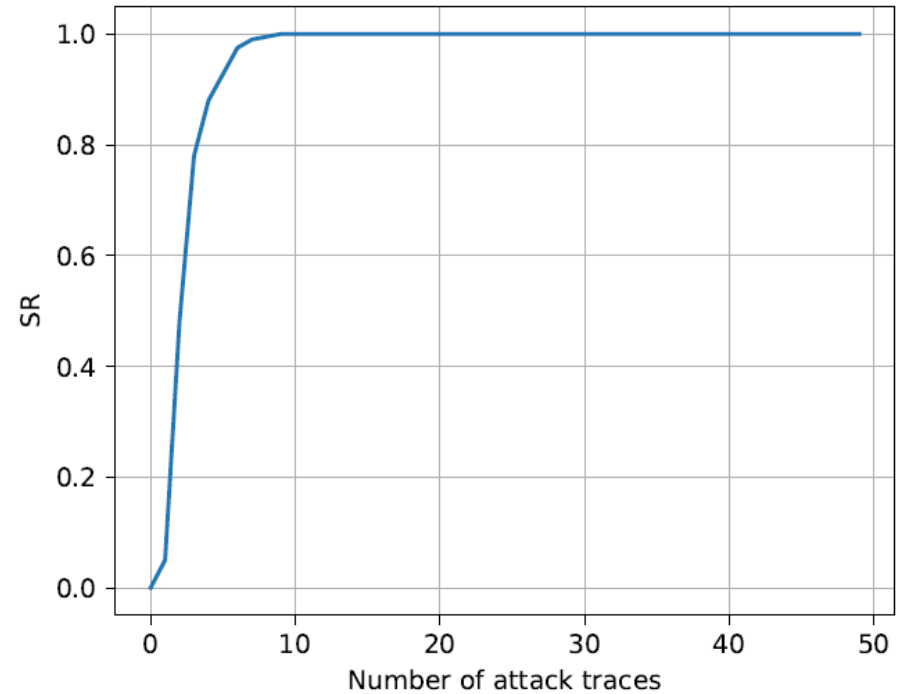
- Adding  $8 + n$  bits of guessing space on  $l_0$  allows knowing the first byte of  $l_i$  for  $i \in [1, n]$ .
- The attack has a computational complexity of  $2^{16+n}$  in order to use  $2^{n+1} - 1$  traces. However, when feasible, it is as trace-efficient as a regular unprotected DPA.

# IMPROVED ATTACK RESULTS

Baseline



Improved



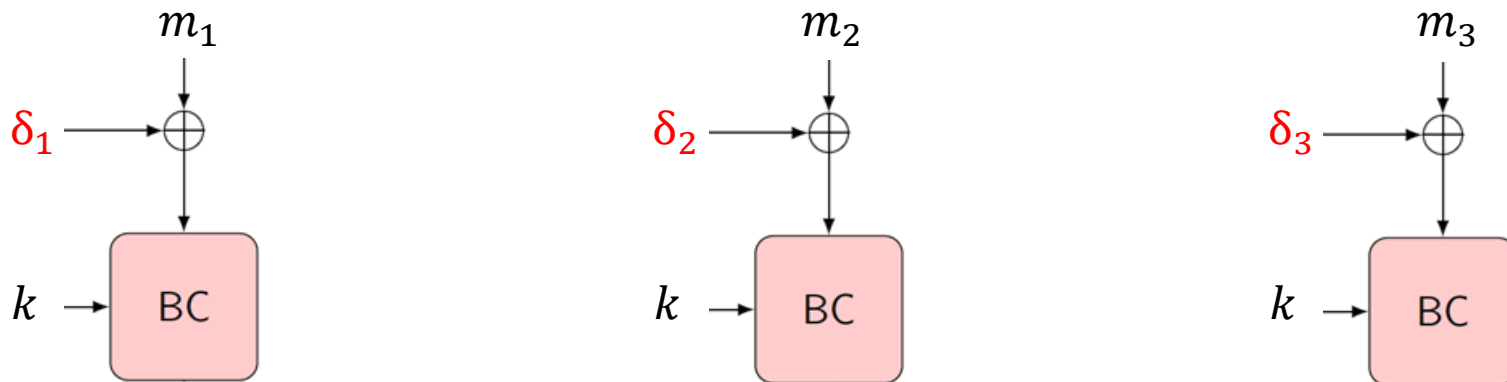
We set the complexity parameter  $n$  such that the complexity was feasible

# OUTLINE

- Introduction and trivial attack paths
  - OCB
  - Trivial attacks
- First level of protection
  - Baseline second-order attack
  - Improved attack
- **Second level of protection**
  - SPA on whitening values
  - Breaking confidentiality and integrity
- Conclusion

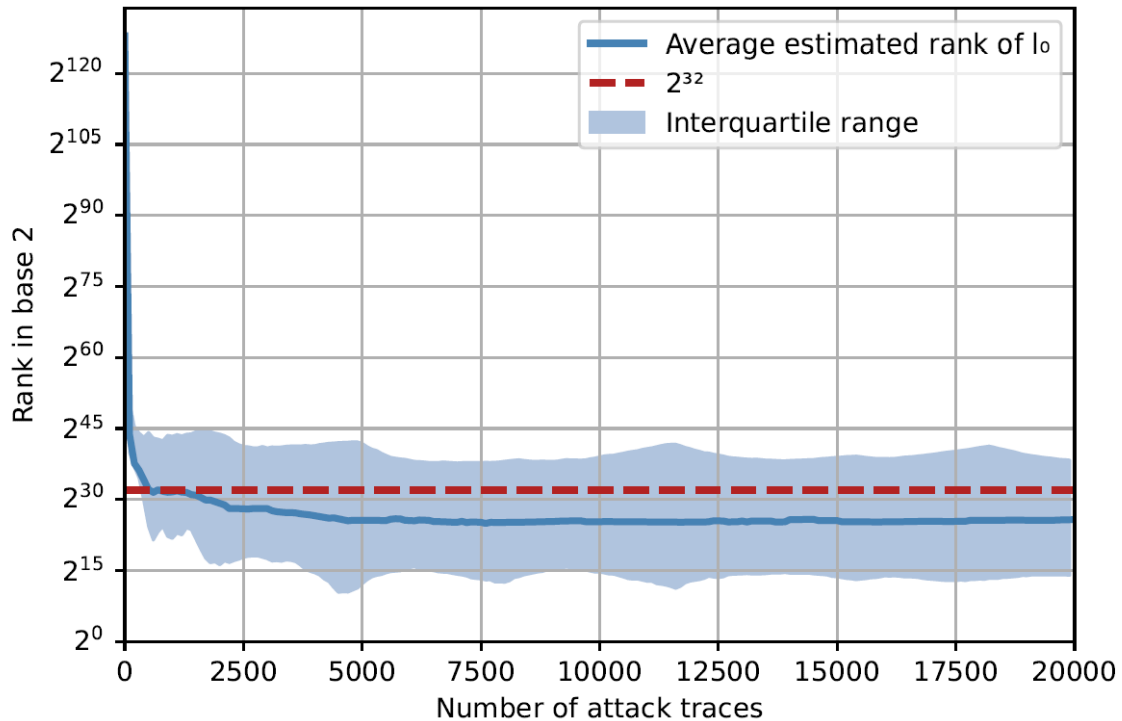
## PROTECTION LEVEL 2

- We now consider an implementation where all the block cipher calls are well protected against SCA



- Only the computations of the  $\delta_i$  are exploitable leakages. Since  $\delta_i = \delta_i \oplus l_{ntz(i)}$ , given a plaintext of  $n$  128-bit blocks,  $l_0$  is loaded  $\frac{n}{2}$  times.
  - We thus perform an SPA on  $l_0$

# SPA ON WHITENING VALUES: RESULTS



Attack performed on twisted (weaker) 8-bit fashion of the implementation

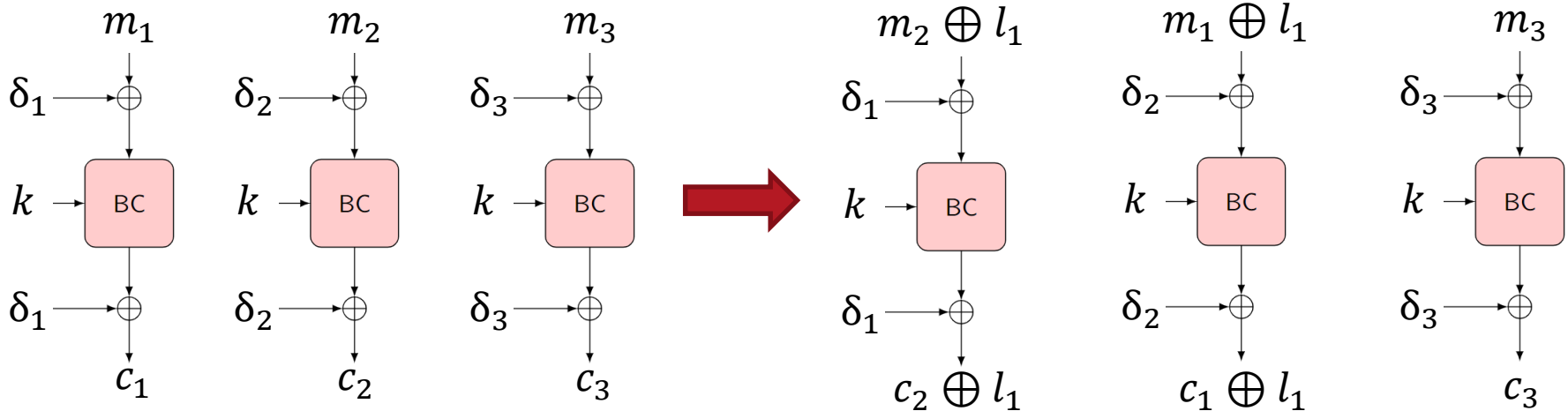
Exploiting the relations between the different  $l_i$  and the use of belief propagation only provided marginal results

## PROTECTION LEVEL 2: IMPACT

- In this scenario, we now assume that the adversary is able to recover the  $l_i$  as it is the only part that is left unprotected.
- These leakages are not enough for key recovery. Yet, they allow breaking **integrity** and **confidentiality** of OCB. Example with integrity:

$(c_2 \oplus l_1, c_1 \oplus l_1, c_3, \tau)$  is a valid forgery  
 $E(m_2 \oplus l_1, m_1 \oplus l_1, m_3, N)$

Query  $E(m_1, m_2, m_3, N) = (c_1, c_2, c_3, \tau)$

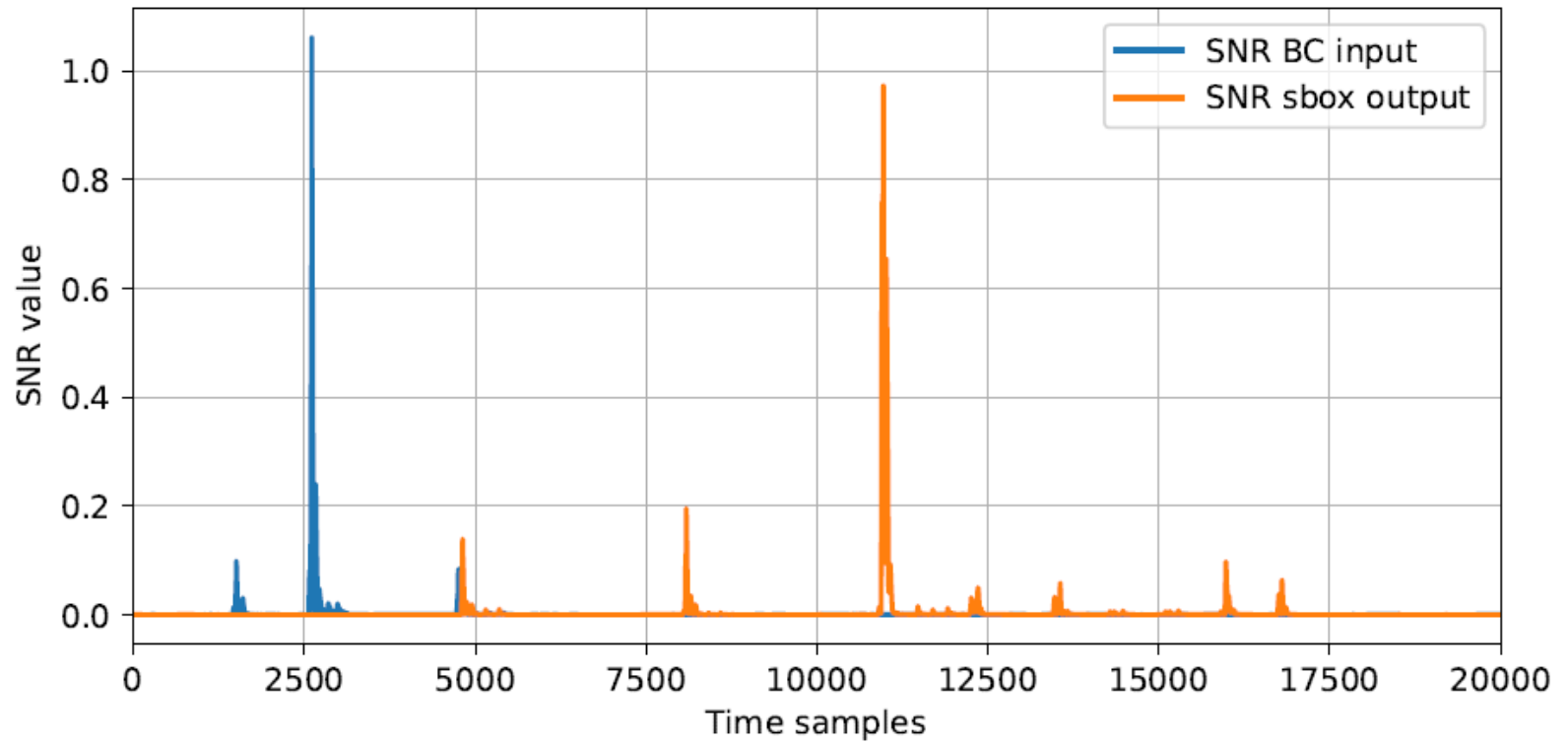




# CONCLUSION

- While OCB does not provide inherent resistance against SCA, a finer-grain analysis balances that statement
  1. Some parts can be attacked with trivial schoolbook DPAs: initialization, processing of AD incomplete message block and decryption
    - Require strong protections against SCA
  2. The inner block cipher calls be targeted using a second-order DPA (costly in traces) or using an extended key space guess (tradeoff data v.s. computation/memory)
    - **Might** be sufficient to use weaker countermeasures (e.g. less shares for masked implementations)
  3. Whitening values outside the block cipher calls can be attacked using SPA. Probably feasible against 8-bit implementations, but harder against larger architectures
    - Recovering the whitening values allows breaking confidentiality and integrity
    - Weak countermeasure might be sufficient, but should not be ignored.
- Other modes might be preferred regarding inherent SCA protection

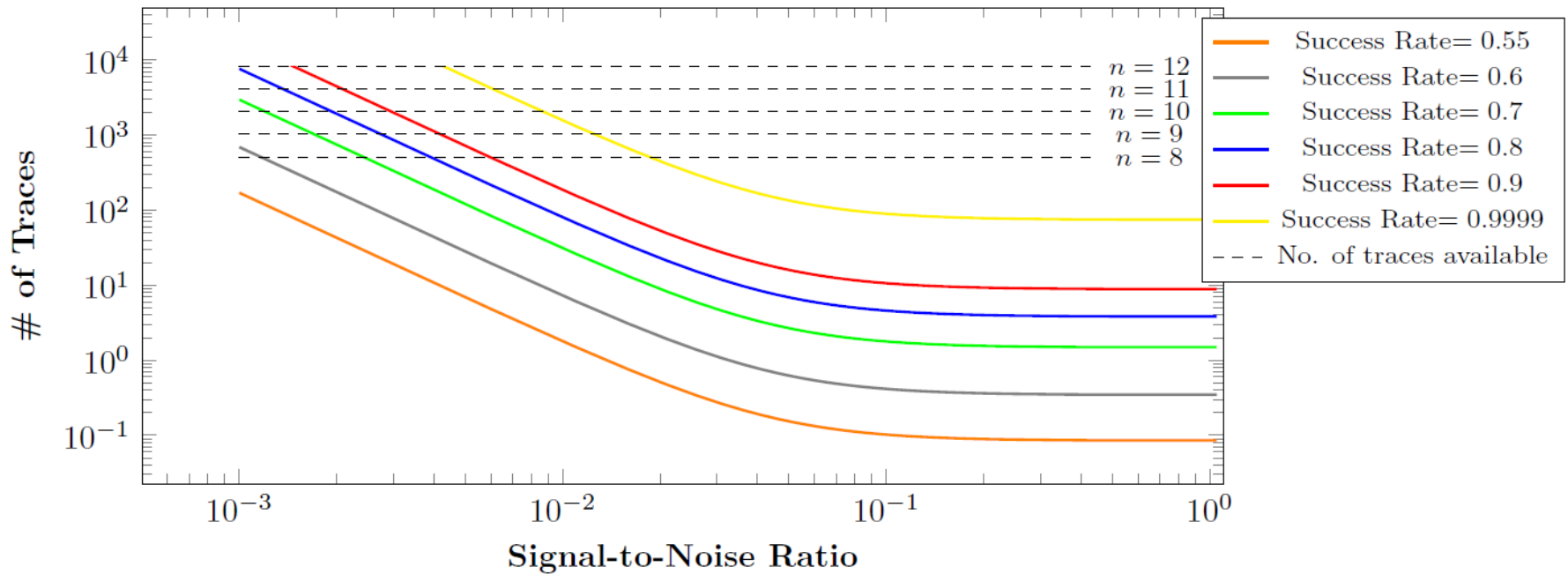
# SNR ANALYSIS



# EXPERIMENTAL SETUP

- **Target:** ARM cortex-m0 running at 48 MHz
- **Implementation of OCB:** C code optimized to use 32-bit values when possible. Sbox using lookup tables for the AES.
- **Scope:** Picoscope 5244D sampled at 500MS/s.
- **Test platform:** STM32F0308 discovery board, power leakages.
- **Testing methodology:** all attacks follow the same procedure
  1. SNR testing and identification of POI
  2. Linear discriminant analysis based on (1)
  3. Gaussien template attack based on (2)
- **Traces sets:**
  - Profiling set: 800K traces of one 128-bit random plaintext block
  - Attack set: 20 plaintexts of 80.000 128-bit blocs each (total 1.6M AES traces)

# IMPROVED ATTACK: THEORETICAL EVALUATION



This attack has a computational complexity of  $2^{16+n}$  in order to use  $2^{n+1} - 1$  traces.

## PROTECTION LEVEL 2: IMPACT

In this scenario, we now assume that the adversary is able to recover the  $l_i$  as it is the only part that is left unprotected.

These leakages are not enough for key recovery. Yet, they allow breaking **integrity** and **confidentiality** of OCB.

While we couldn't exhibit any key recovery attack using the sole knowledge of  $l_i$ , we still showed the following vulnerabilities:

- **Attack against confidentiality:** *for certain messages, we are able to distinguish between their ciphertexts made via OCB from random ones.*
- **Attack against integrity:** *We forge a valid pair  $(c^*, tag)$  using the knowledge of the  $l_i$  and a valid pair  $(c, tag)$*

## PROTECTION LEVEL 2: BREAKING INTEGRITY (DETAILS)

For a n-bloc message, tag  $\tau = BC_k(\delta_n \oplus \sum_i m_i)$

Encryption query of 3 128-bit message m:  $E(m_1, m_2, m_3, N) = (c_1, c_2, c_3, \tau)$

$(c'_1, c'_2, c'_3, \tau) = (c_2 \oplus l_1, c_1 \oplus l_1, c_3, \tau)$  is a valid forgery, and would correspond to the encryption of  $m'$ :  $E(m_2 \oplus l_1, m_1 \oplus l_1, m_3, N)$

$$\begin{aligned}c'_1 &= BC_k(m_2 \oplus l_1 \oplus l_0 \oplus \delta_0) \oplus l_0 \oplus \delta_0 = c_2 \oplus l_1 \\c'_2 &= BC_k(m_2 \oplus \cancel{l_1} \oplus l_0 \oplus \cancel{l_1} \oplus \delta_0) \oplus l_0 \oplus l_1 \oplus \delta_0 = c_1 \oplus l_1 \\c'_3 &= c_3\end{aligned}$$

And we have  $c'_1 \oplus c'_2 \oplus c'_3 = c_2 \oplus l_1 \oplus c_1 \oplus l_1 \oplus c_3 = c_1 \oplus c_2 \oplus c_3$

# IMPROVED ATTACK OVERVIEW

The output of the first round can now be guessed

