23rd International Conference on Fast Software Encryption (FSE 2016)

# There is Wisdom in Harnessing the Strengths of your Enemy: Customized Encoding to Thwart Side-Channel Attacks*

Houssem MAGHREBI, Victor SERVANT, Julien BRINGER

*Partially funded by the ANR project SERTIF



SAFRAN Morpho

# OUTLINE

1. Introduction
2. Towards a New Encoding Procedure for Leakage Balancing
3. Theoretical Analysis
4. Security Evaluation
5. Practical Evaluation
6. Conclusion and Future Works

# Introduction

# SIDE-CHANNEL COUNTERMEASURES

- **Masking**
  - Secret sharing and variants
  - A random leakage $Z = (Z \oplus M, M)$
    - Vulnerable to Higher-order attacks

- **Hiding**
  - Dual-Rail
  - A constant leakage
    - Strongly depends on the hardware leakage

- **Others**
  - Shuffling
  - Noise Generators
  - Jitters
  - …
    - Not sufficient to counteract practical attacks

# LEAKAGE HIDING

## In Hardware

- Dual-Rail : balancing of pairs of wires
  - Depending on cell design, could be vulnerable to glitches

## In Software

- Lightweight implementations (PRESENT)
  - Hoogvorst et al. (2011) : Dual-Rail encoding
  - Han et al. (2012) : *idem* with XOR simplification
  - Rauzy et al. (2014): Dual-rail with Precharge Logic (DPL) encoding
  - Each n-bit variable is encoded as 2n-bit variable of Hamming weight n

| Bit value | Codeword |
|---|---|
| 0 | 01 |
| 1 | 10 |

- Block cipher implementations
  - Servant et al. (2014): Constant weight implementation of AES
  - (m, n)-codes : words of Hamming weight equal to *m* of *n*-bit length
  - AES case: use the (3,6)-code
  - Each 4-bit variable is encoded as 6-bit variable of Hamming weight 3

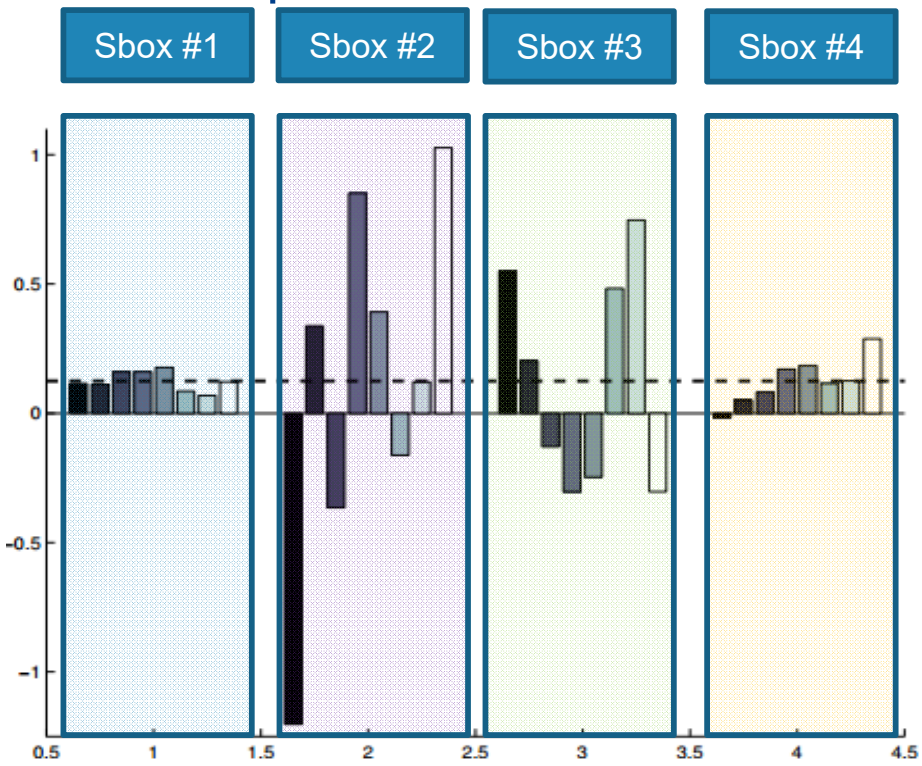
| | | | |
|---|---|---|---|
| $0 \rightarrow 000111$ | $4 \rightarrow 010011$ | $8 \rightarrow 011010$ | $12 \rightarrow 100110$ |
| $1 \rightarrow 001011$ | $5 \rightarrow 010101$ | $9 \rightarrow 011100$ | $13 \rightarrow 101001$ |
| $2 \rightarrow 001101$ | $6 \rightarrow 010110$ | $10 \rightarrow 100011$ | $14 \rightarrow 101010$ |
| $3 \rightarrow 001110$ | $7 \rightarrow 011001$ | $11 \rightarrow 100101$ | $15 \rightarrow 101100$ |

# CONSTANT WEIGHT COUNTERMEASURES

- A constant weight implementation is a leak-free countermeasure under a Hamming weight leakage model assumption
- Is the Hamming weight leakage model a realistic assumption?
- So, the claimed security guarantee of

constant weight countermeasures vanishes!

How to balance the leakage in realistic scenarios?




Bit weights of 4 AES SBOX outputs (90nm CMOS technology)

V. Lomné, E. Prouff, and T. Roche. Behind the scene of side channel attack. ASIACRYPT 2013 -, Bengaluru, India, December 1-5, 2013, Proceedings, Part I, volume 8269 of LNCS, pages 506-525. Springer, 2013.

# Towards a New Encoding Procedure for Leakage Balancing

# PROPOSAL STRATEGY

➔ Our framework is composed of two steps

1. Stochastic characterization of the leakage function
   - Independent Bit Leakage assumption
   - The leakage function satisfies: $L(Z) = \sum_{i=1}^{n} \alpha_i Z[i] + N(0, \sigma)$ (i.e. linear basis)
   - The bit leakage weights ($\alpha_i$) are recovered with a linear regression analysis

2. Encoding function selection

**Algorithm 1** Selection of the optimal encoding function

**Input:** $m$: the codeword bit-length, $n$: the sensitive variable bit-length and $\alpha_i$: the leakage bit weights, where $i$ in $[\![1, m]\!]$

**Output:** $2^n$ codewords of $m$-bit length

1: **for** $X$ in $[\![0, 2^m - 1]\!]$ **do**
2: Compute the power consumption for each codeword $X$ and store the result in table $D$: $D[X] = \sum_{i=1}^{m} \alpha_i X[i]$
3: Store the corresponding value of the codeword in the index table $I$: $I[X] = X$
4: **end for**
5: Sort the power consumption stored in table $D$ and the index table $I$ accordingly
6: **for** $j$ in $[\![0, 2^m - 2^n]\!]$ **do**
7: Find the $argmin$ of $|D[j] - D[j + 2^n]|$
8: **end for**
9: **return** $2^n$ codewords corresponding to $[\![I[argmin], I[argmin + 2^n]]\!]$

Minimize the variance (the SNR)

Algorithm 1 is still applicable if the IBL assumption is not respected

# Theoretical Analysis of our Proposal

# SECURITY METRICS

➔ To theoretically analyze our proposal we have considered two security metrics:

1. The Signal-to-Noise Ratio: $\mathrm{SNR} = Var\left(\sum_{i=1}^{n} \alpha_i Z[i]\right)/\sigma^2$

2. The minimum number of traces to disclose the key with 90%: $N_{90\%} \simeq 8\frac{Z^2{}_{90\%}}{4\rho^2}$

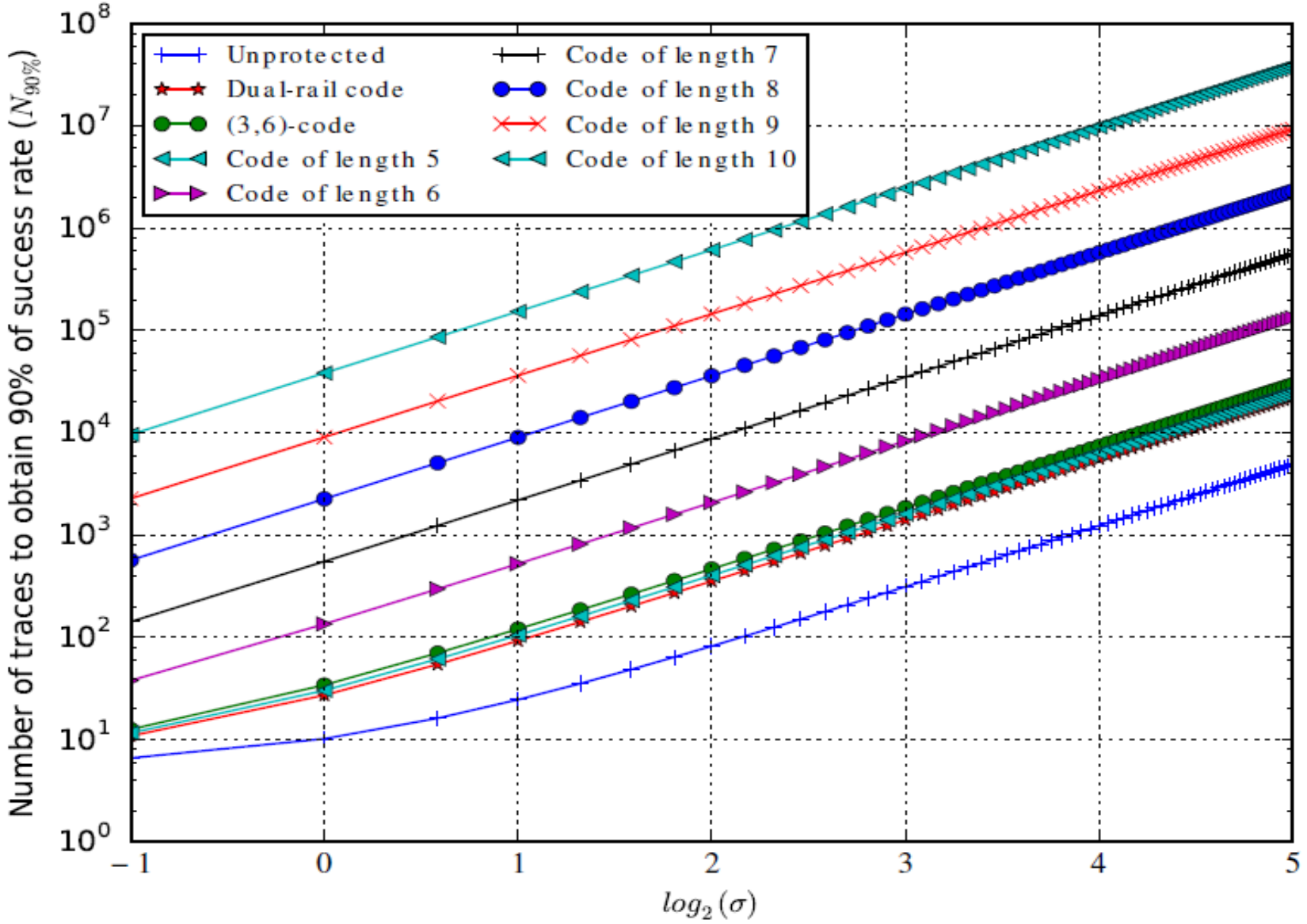
➔ Since $\rho = \sqrt{\frac{1}{1+\frac{1}{SNR}}}$ then we can exhibit a relationship between $N_{90\%}$ and $\mathrm{SNR}$:

$$N_{90\%} \simeq 8\frac{Z^2{}_{90\%}}{SNR}$$

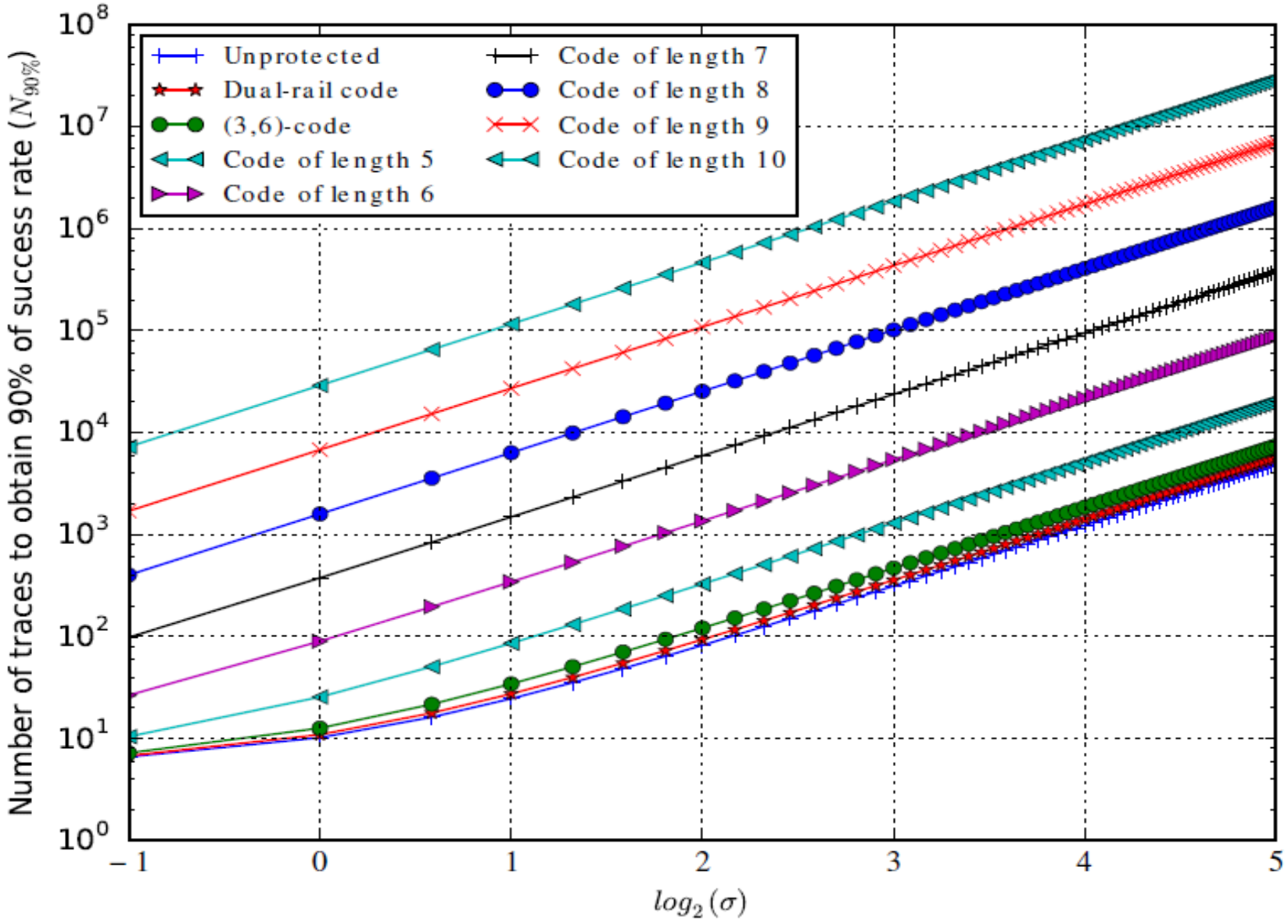
➔ For the sake of comparison, we targeted other implementations:

- Unprotected: $L(Z) = \sum_{i=1}^{n} \alpha_i Z[i] + N(0,\sigma)$
- Hardware constant weight: $L(Z) = \sum_{i=1}^{n} \alpha_i C_{HW}(Z[i]) + N(0,\sigma)$, $C_{HW}$ is the dual-rail code
- Software constant weight: $L(Z) = \sum_{i=1}^{n} \alpha_i C_{SW}(Z[i]) + N(0,\sigma)$, $C_{SW}$ is the (3,6)-code
- Our proposal: $L(Z) = \sum_{i=1}^{n} \alpha_i C_{cust}(Z[i]) + N(0,\sigma)$, $C_{cust}$ our customized encoding

# EVOLUTION OF $N_{90\%}$ ACCORDING TO $\sigma$ WHEN $\sigma_\alpha = 0.05$



# EVOLUTION OF $N_{90\%}$ ACCORDING TO $\sigma$ WHEN $\sigma_\alpha = 0.5$



# Security Evaluation of our Proposal

# INFORMATION THEORETIC ANALYSIS

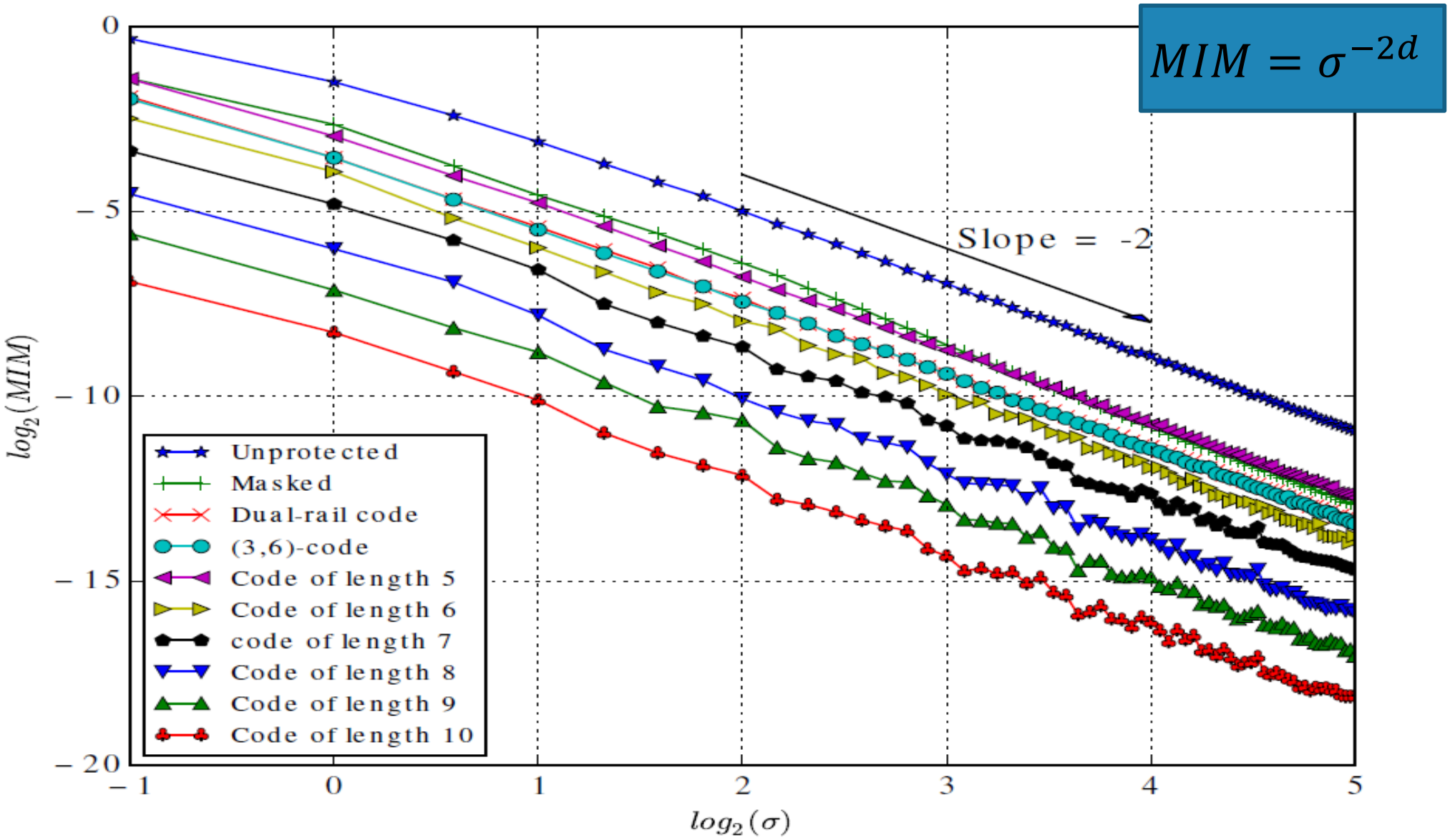
- To evaluate the information revealed by our proposed encoding functions, we compute the Mutual Information Metric $\mathrm{I}[\mathrm{L}(\mathrm{Z});\mathrm{Z}] = H[L(Z)] - H[L(Z)|Z]$

- For the sake of comparison, we compute the MIM for several implementations:
  - Unprotected: $L(Z) = \sum_{i=1}^{n} \alpha_i Z[i] + N(0,\sigma)$
  - First-order masking: $L(Z) = (\sum_{i=1}^{n} \alpha_i Z[i] \oplus M[i]) * (\sum_{i=1}^{n} \beta_i M[i]) + N(0,\sigma)$
  - Hardware constant weight: $L(Z) = \sum_{i=1}^{n} \alpha_i C_{HW}(Z[i]) + N(0,\sigma)$, $C_{HW}$ is the dual-rail code
  - Software constant weight: $L(Z) = \sum_{i=1}^{n} \alpha_i C_{SW}(Z[i]) + N(0,\sigma)$, $C_{SW}$ is the (3,6)-code
  - Our proposal: $L(Z) = \sum_{i=1}^{n} \alpha_i C_{cust}(Z[i]) + N(0,\sigma)$, $C_{cust}$ our customized encoding

- The standard deviations of the bit leakage weights varies in [0.05, 0.5]

- The mutual information is proportional to $\sigma^{-2d}$, where d denotes the order of the smallest statistical moment in the leakage distribution depending on the secret key

# MIM RESULTS WHEN $\sigma_\alpha = 0.05$

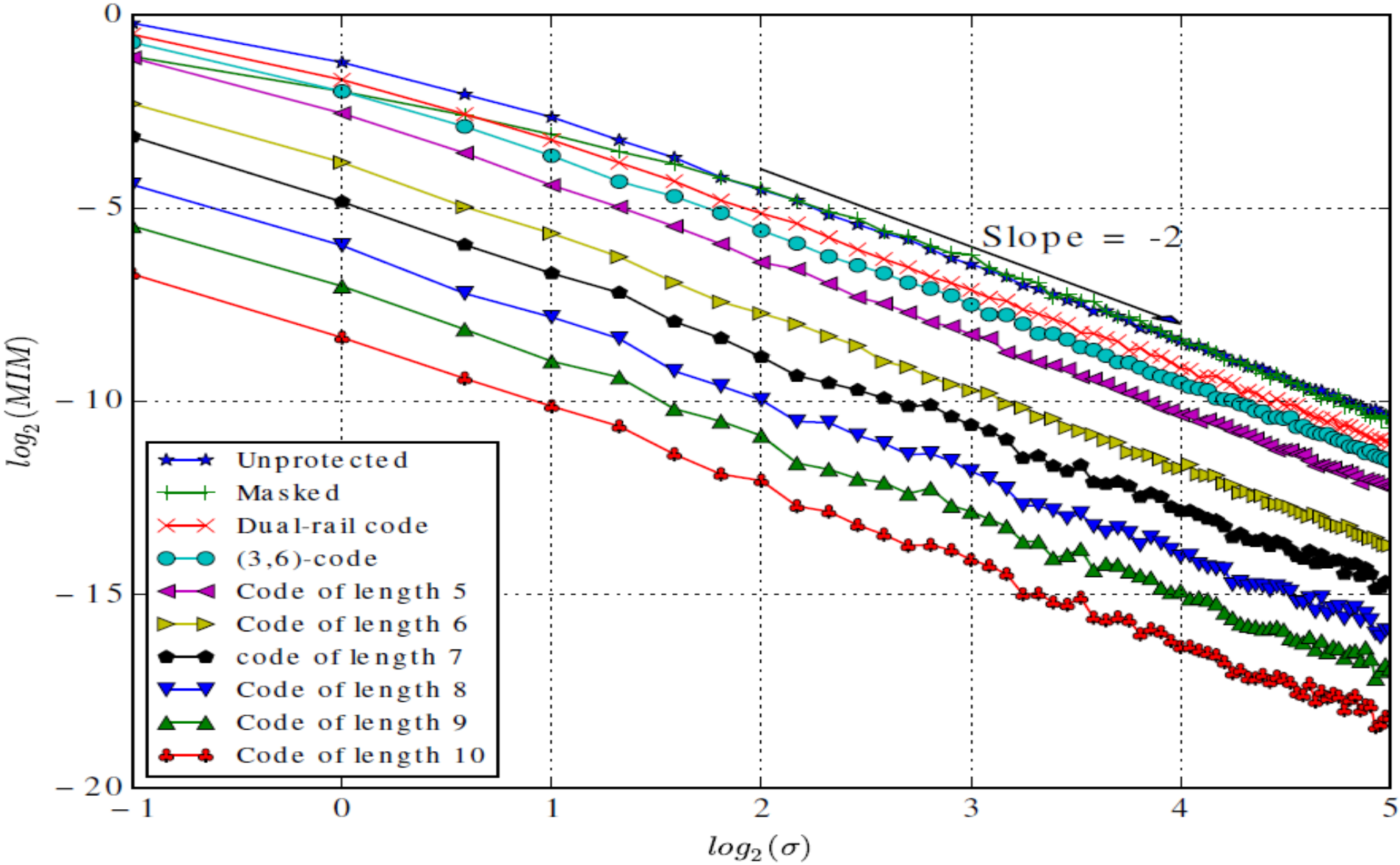







SAFRAN Morpho

# MIM RESULTS WHEN $\sigma_\alpha = 0.5$








SAFRAN Morpho

# SIDE-CHANNEL ANALYSIS

## ➔ Simulation Setup

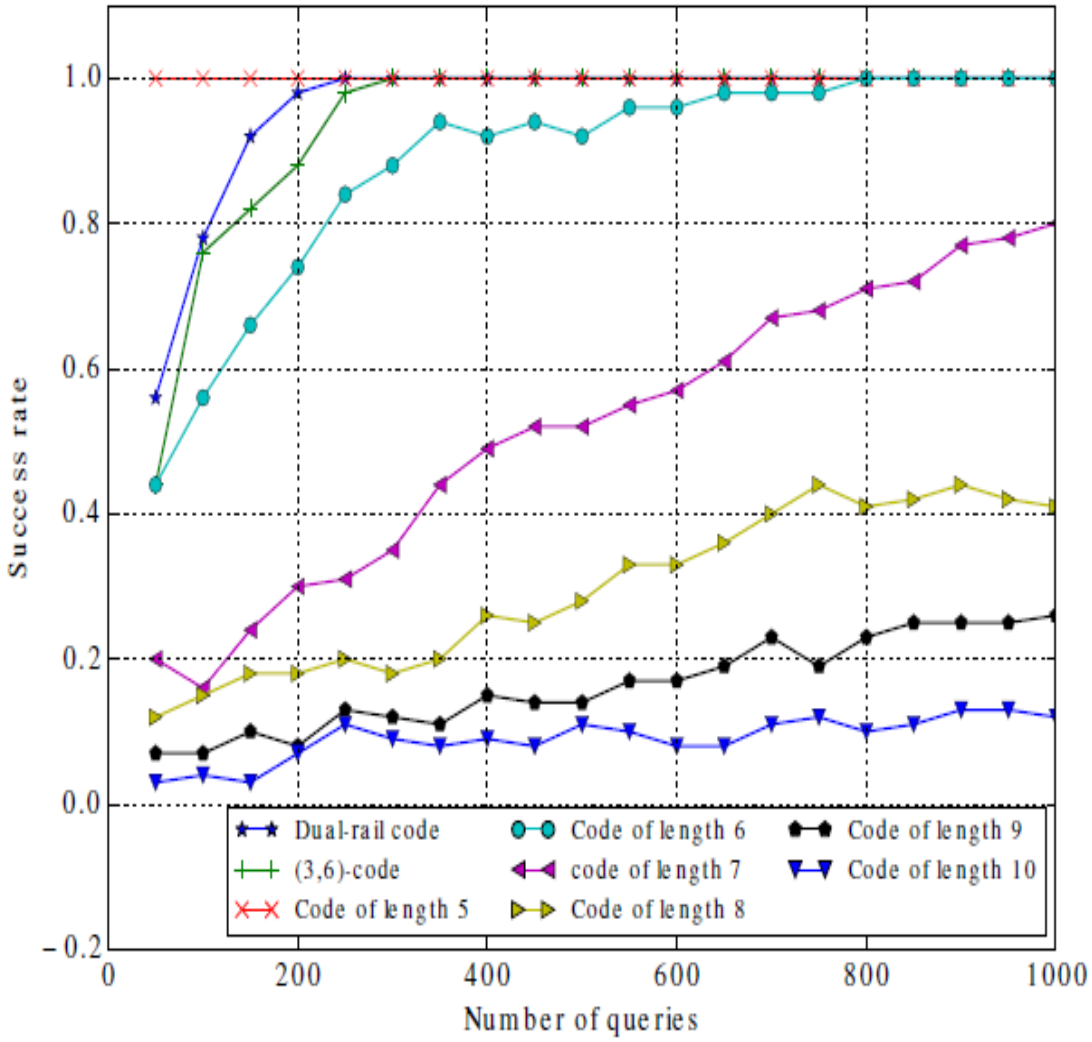
- PRESENT Sbox output: $S[X \oplus K]$
- Targeted implementations
  - Hardware constant weight $C_{HW}\ (S[X \oplus K])$
  - Software constant weight $C_{SW}\ (S[X \oplus K])$
  - Our proposal $C_{cust}\ (S[X \oplus K])$

## ➔ Two attack scenarios:

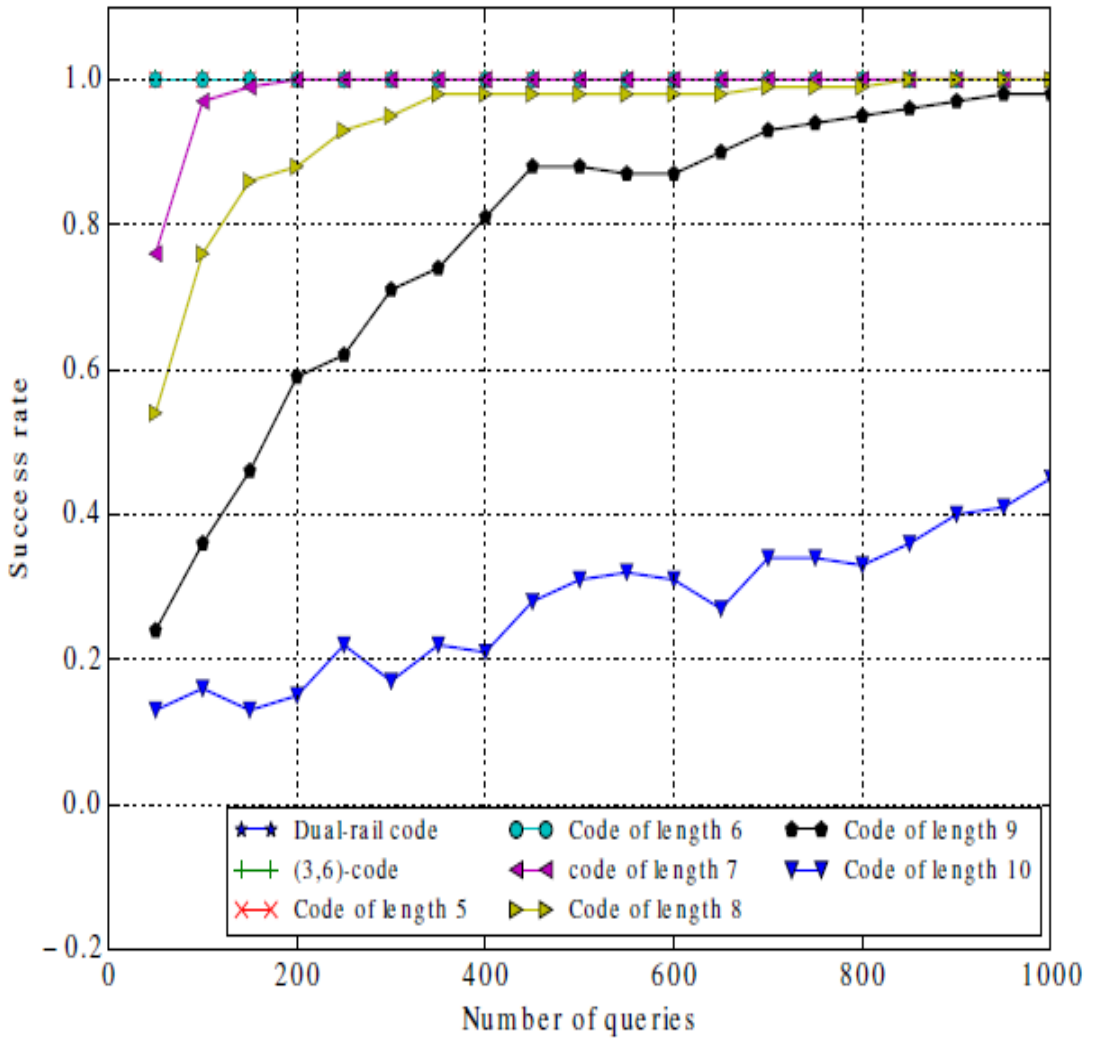
- Best-case scenario
  - We consider a powerful adversary who has access to:
    - The bit leakage weights
    - The customized encoding function
- Worst-case scenario
  - We consider a weaker adversary:
    - The bit leakage weights are unknown
    - The used customized encoding function is unknown

# ATTACK RESULTS IN THE BEST CASE SCENARIO

$\sigma_\alpha = 0.05$




$\sigma_\alpha = 0.5$



# ATTACK RESULTS

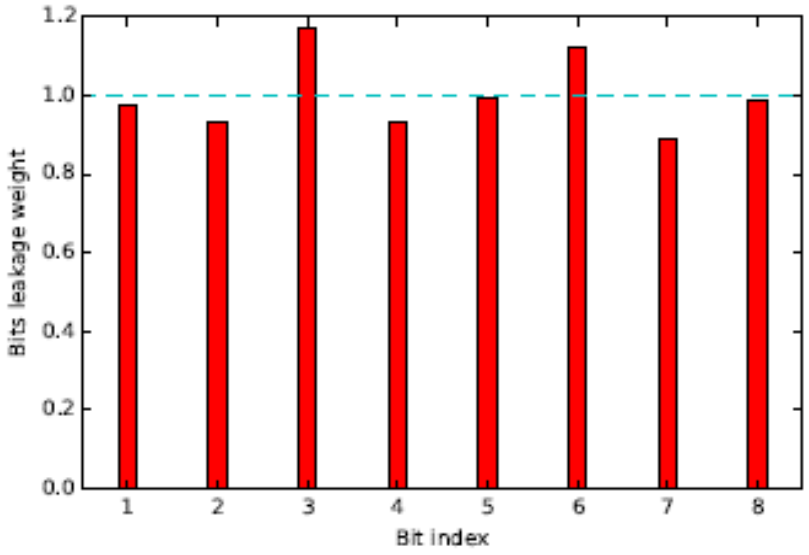
- In the worst-case scenario, the attack performs worse since the adversary does not have the control on the code length and the used subset of codewords
- The obtained results within the 4 considered security metrics (the SNR, the MTD, the information theoretic, the success rate of stochastic attack) are in-line
  - The claimed security guarantee of constant weight countermeasures vanishes
  - The longer the code is, the less information is leaked, the lowest the SNR is, and the more traces are needed to break the implementation
  - The higher deviation from the Hamming weight model is, the larger the leakage is, the longer the code is needed to protect the implementation
- What about the practice?

# Practical Evaluation of our Proposal

# EXPERIMENTAL SETUP (1)

- Take 4 different copies of the STM32F3 circuit
- Perform a stochastic profiling of the leakage and customized encoding generation on one copy
- Use the same encoding function to protect the 3 others copies without a prior profiling
- Target the PRESENT Sbox computation protected by a customized encoding function
- Compute the optimal encoding functions of length

in [5, 7] to protect the 4-bit PRESENT Sbox output





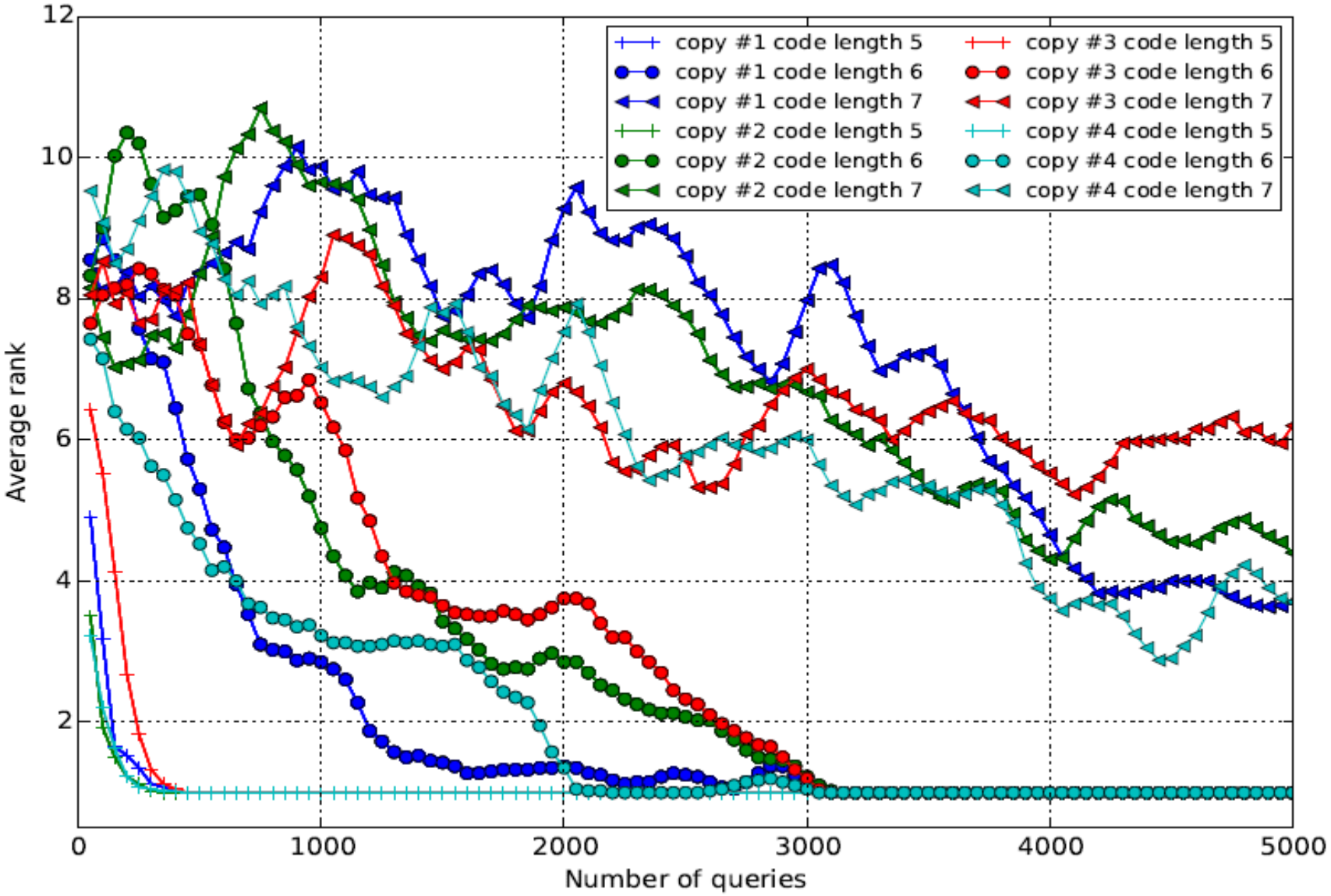



SAFRAN Morpho

# EXPERIMENTAL SETUP (2)

- Implement the protected PRESENT Sbox on each copy
- Acquire 50.000 EM traces with a fixed setup
  - same electromagnetic probe
  - same probe's position
  - same oscilloscope configuration
  - same temporal acquisition window
- For each copy and code length, conduct 10 independent CPA attacks using 10 independent set of 5.000 EM traces
- Estimate the evolution of the averaged rank of the correct key

# ATTACK RESULTS



# Conclusions & Perspectives

# CONCLUSIONS

- A new framework for building customized encoding function according to the physical leakage characteristics of the target device
- The obtained results within the four considered security metrics (i.e. the SNR, the MTD, the information theoretic, the success rate of SCA attack) are in-line
- The practical assessment of our solution have enabled us to confirm its practicability to protect cryptographic operations when applied on four different copies of the same device.
- Encoding is a promising strategy to thwart SCA attacks

# PERSPECTIVES

- Evaluate a hole block cipher protected with our proposal
- Study new encoding functions when assuming a higher-order leakage model (e.g. quadratic, cubic, . . . )
- Study new designs by combining both masking and encodings
- Study the inter-conversion of encoding functions when the registers of a circuit have different leakage mode

# THANK YOU FOR YOUR ATTENTION

23rd International Conference on Fast Software Encryption (FSE 2016)

# There is Wisdom in Harnessing the Strengths of your Enemy: Customized Encoding to Thwart Side-Channel Attacks*

Houssem MAGHREBI, Victor SERVANT, Julien BRINGER

*Partially funded by the ANR project SERTIF



SAFRAN Morpho

# IMPLEMENTATION CONSIDERATIONS

## ➔ Encoded operations

- Pre-computed tables in non-volatile memory
- Only encoded values contain an output
- The rest of the table is 0
  - Some faults could be detected for free!
- Single operand, non-linear (***SubBytes***)
  - SubByte'[ **C(**A**)** ] = **C(** SubByte[A] **)**
- Single operand, linear (***Xtimes***)
  - Xtimes'[ **C(**A**)** ] = **C(** Xtimes[A] **)**
- Two-operand, linear (***XOR***)
  - XOR'[ **C(**A**)** || **C(**B**)** ] = **C(** A ⊕ B **)**

| @ | values |
|---|---|
| 000111 | XXX |
| 001000 | 000 |
| 001001 | 000 |
| 001010 | 000 |
| 001011 | XXX |