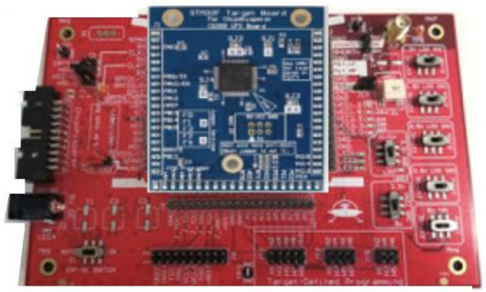
SIDE CHANNEL ANALYSIS: INSTRUCTION EXTRACTION AND INFORMATION ESTIMATION

Rennes 2022 | Valence Cristiani

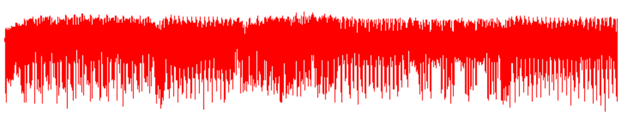
GOAL



Target board

Side channel measurments

Oscilloscope



Side channel Trace

⋮

Xor r2, r3, r2

Mov r0, r1

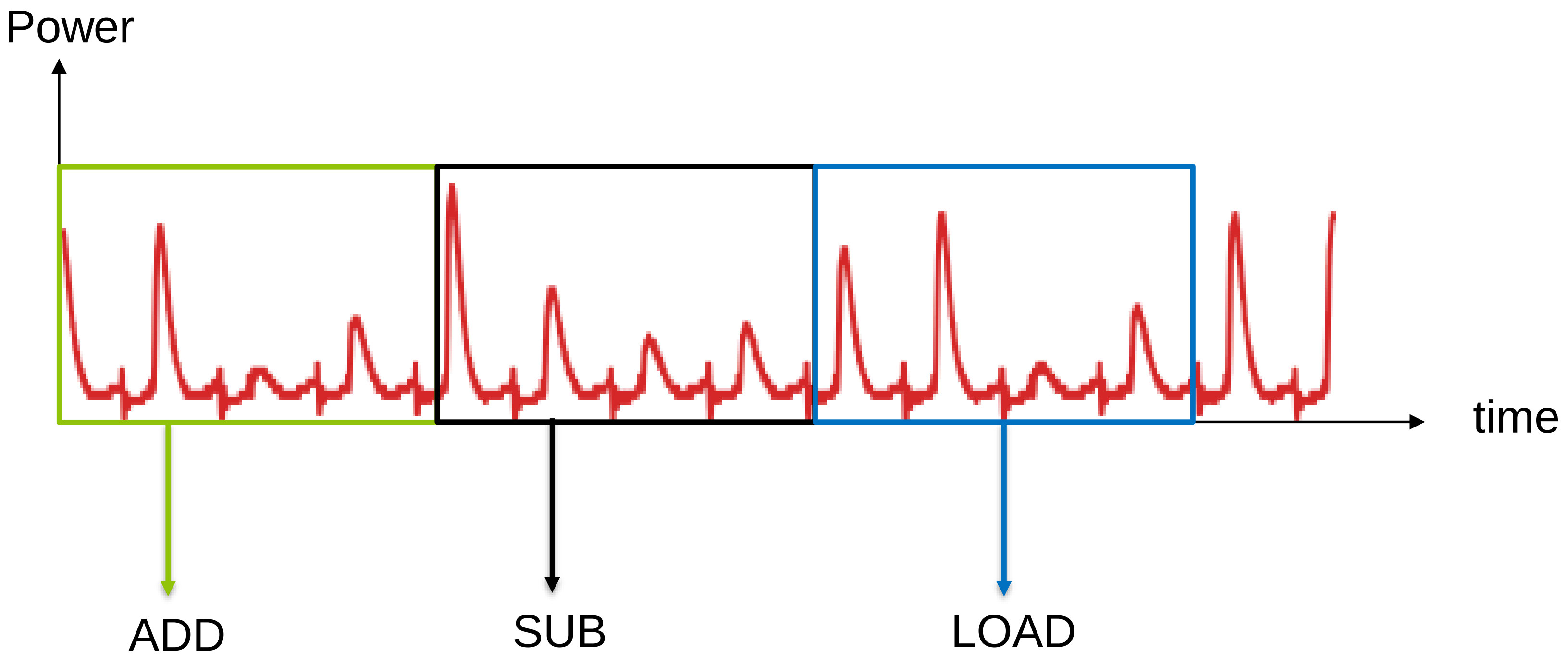
Sub r1, r2, r3

⋮

Instruction flow

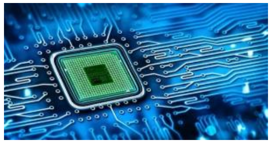
GOAL

Label a trace with executed instructions

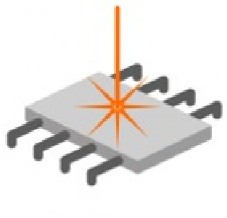


WHY IS IT INTERESTING ?

- Leakage characterization of processor architecture



- Detect interesting zones of the code
  - AES
  - Function entry/exit point
  - Combine with fault injection attack



- Detect malwares

- Reverse proprietary source code

SUMMARY

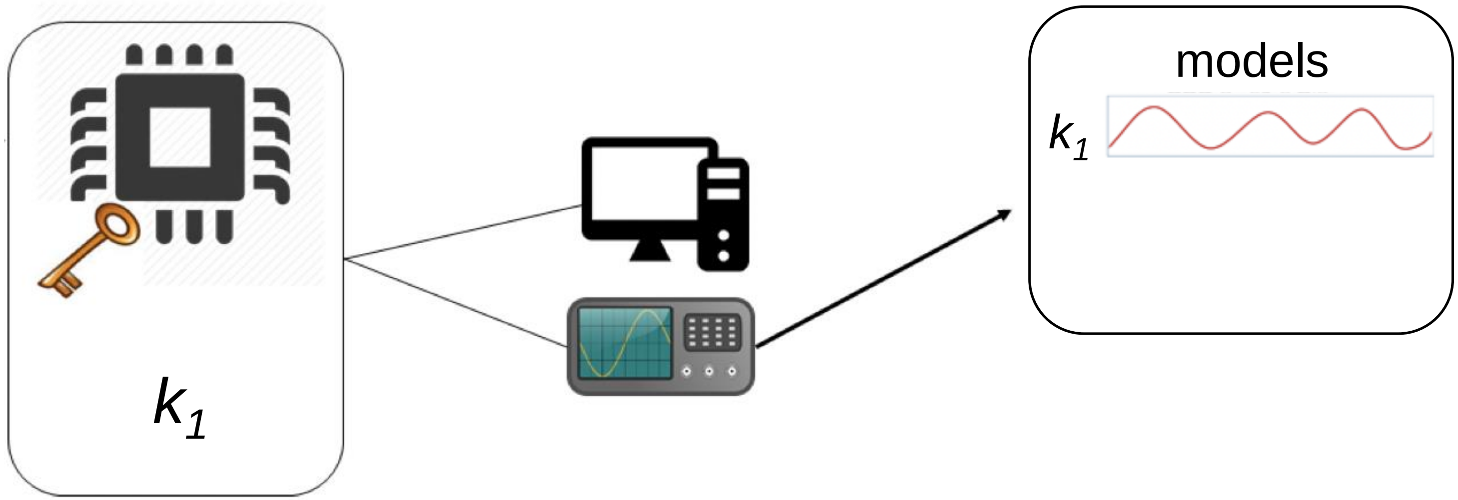
I. Template attack to recover instructions

II. New approach: a bit level reconstruction

III. Results

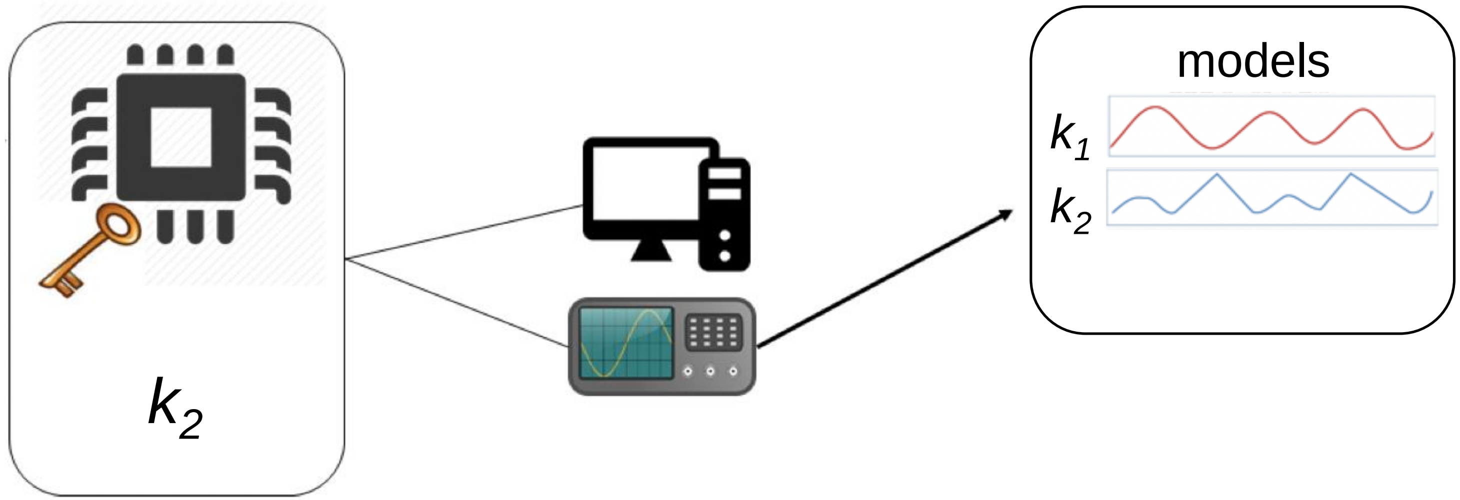
TEMPLATE ATTACK

- 3 possible secret keys : k_1, k_2 and k_3



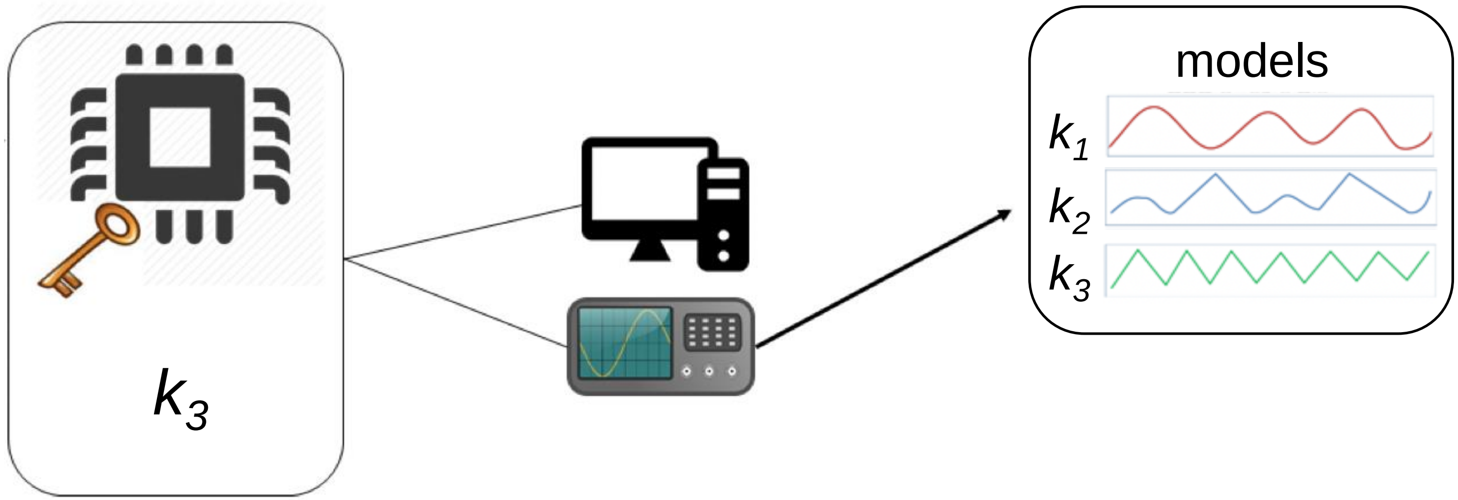
TEMPLATE ATTACK

- 3 possible secret keys : k_1, k_2 and k_3



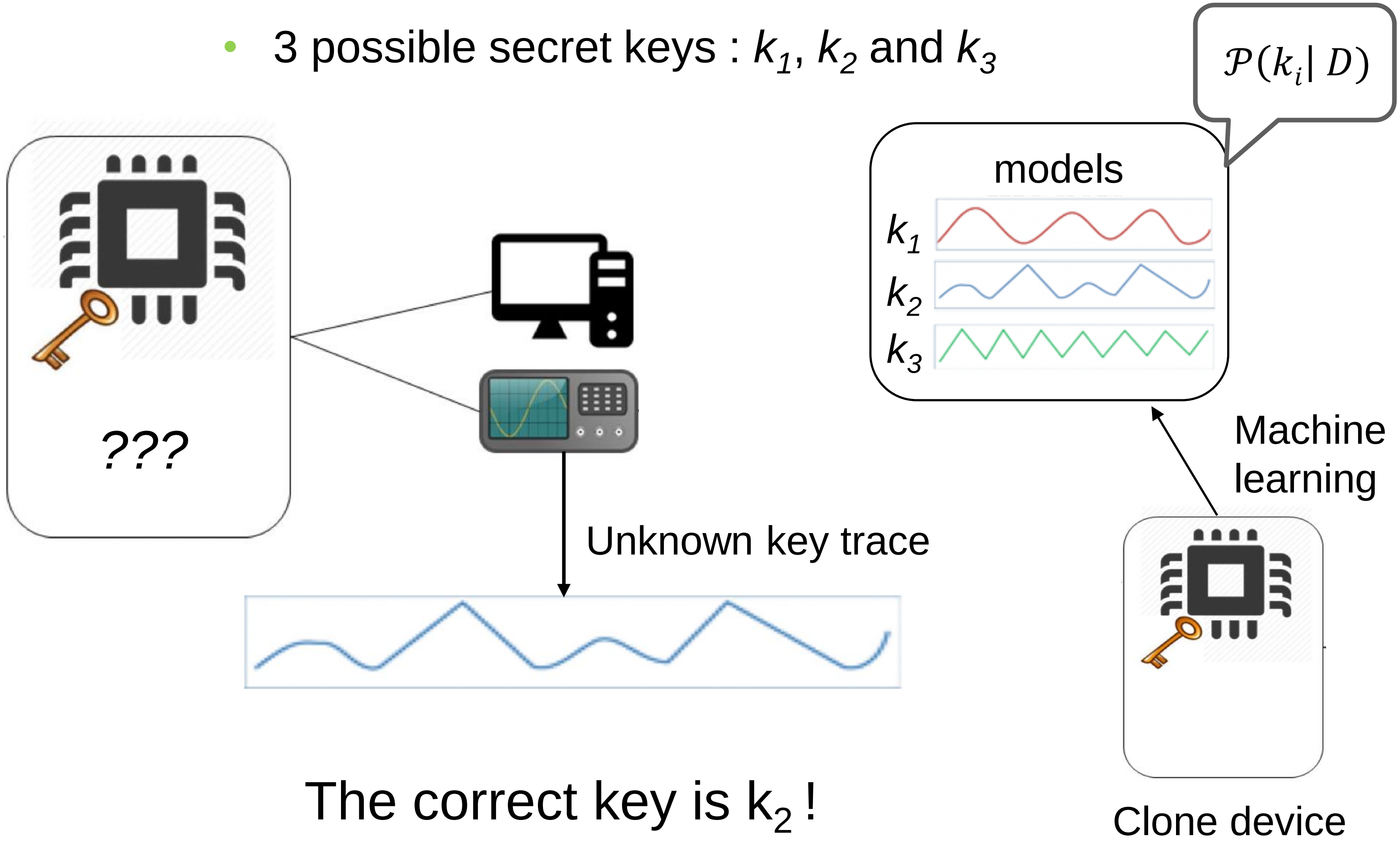
TEMPLATE ATTACK

- 3 possible secret keys : k_1, k_2 and k_3



TEMPLATE ATTACK

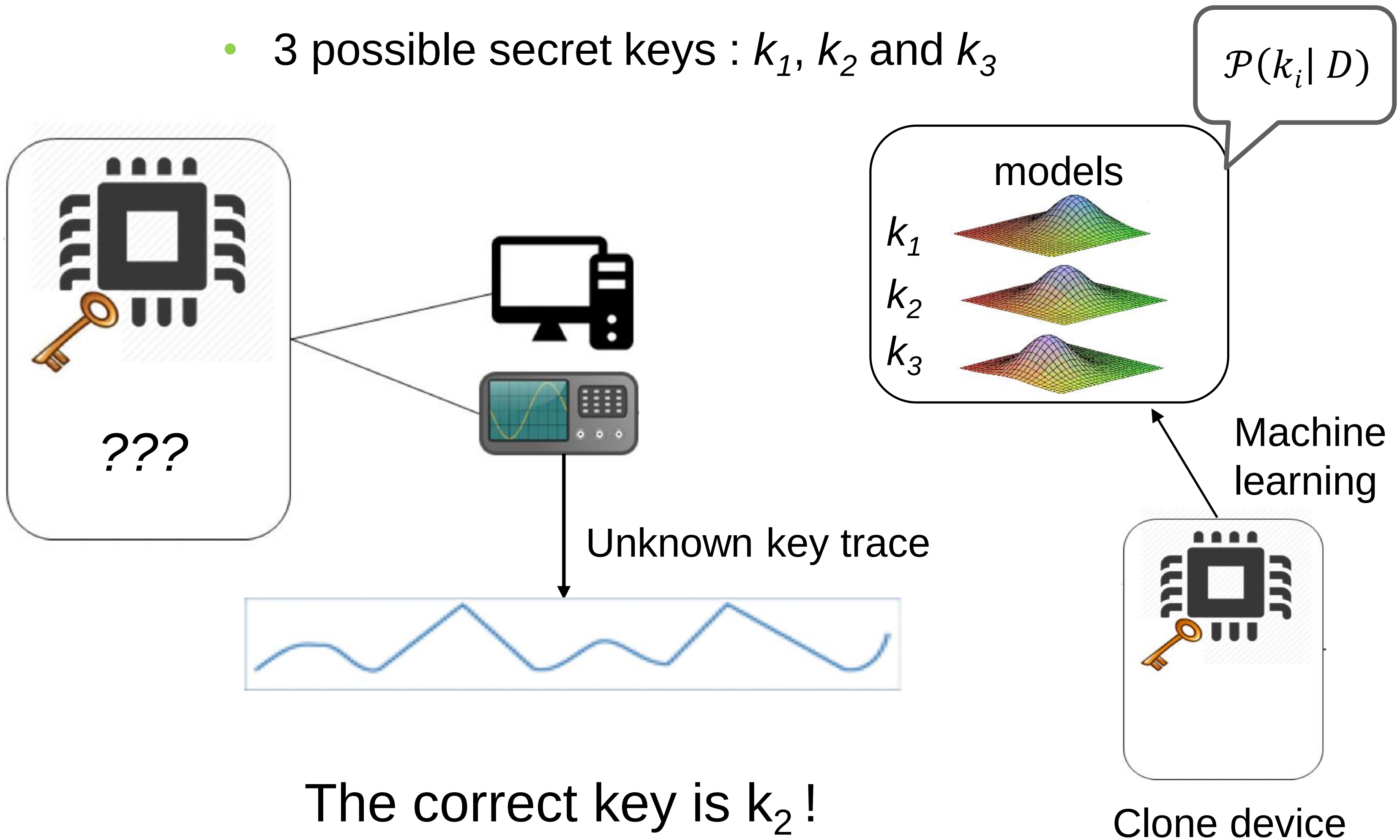
- 3 possible secret keys : k_1, k_2 and k_3



The correct key is k_2 !

TEMPLATE ATTACK

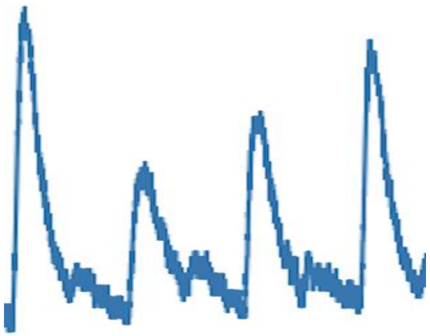
- 3 possible secret keys : k_1, k_2 and k_3



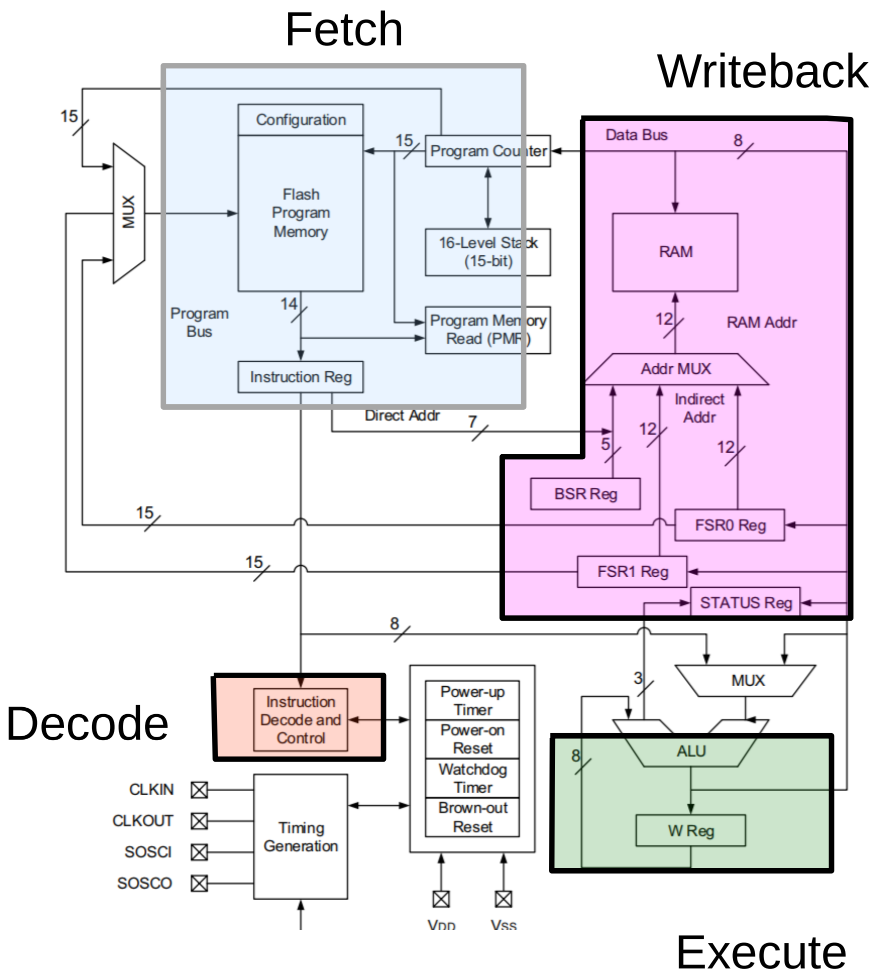
The correct key is k_2 !

PIC16F ARCHITECTURE

- Simple 8 bits microcontroller
- 14 bits instructions
- Why PIC ?
  - Most widely used target in SotA
  - Very simple
- 4 clock cycles per instruction
- 2 stages pipeline



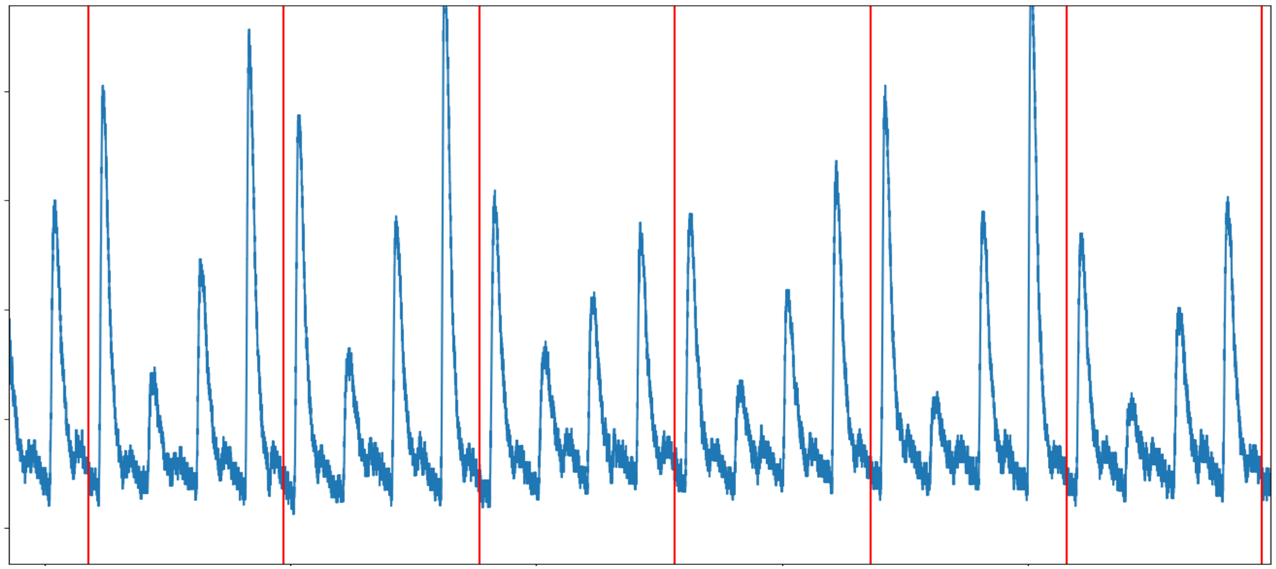
Typical instruction trace (Power)



INSTRUCTIONS EXTRACTION

A very long trace…

Power

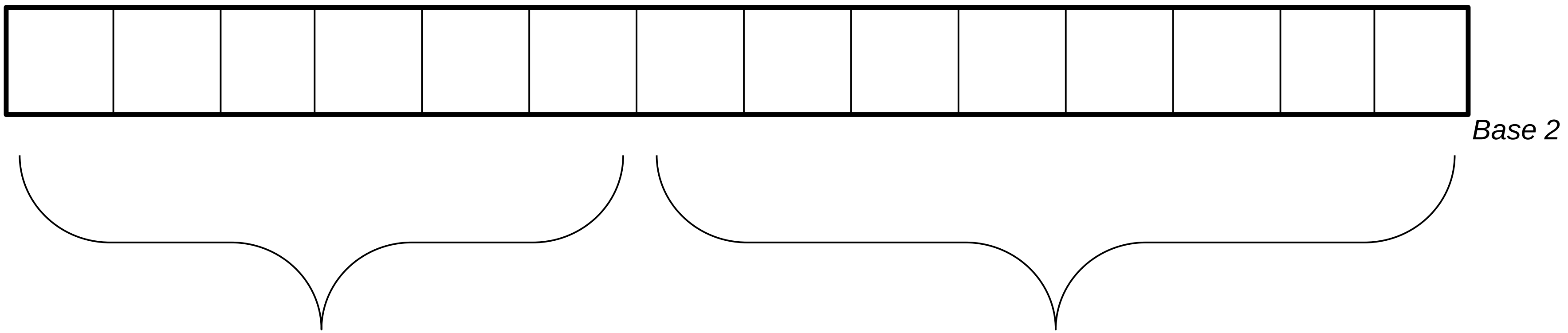


Time

- ➢ Instruction cutting and alignment

ASSEMBLY INSTRUCTIONS

- Binary word (opcode+operands) stored in the instruction register



Opcode

(ADD, MULT, MOV…)

Operands

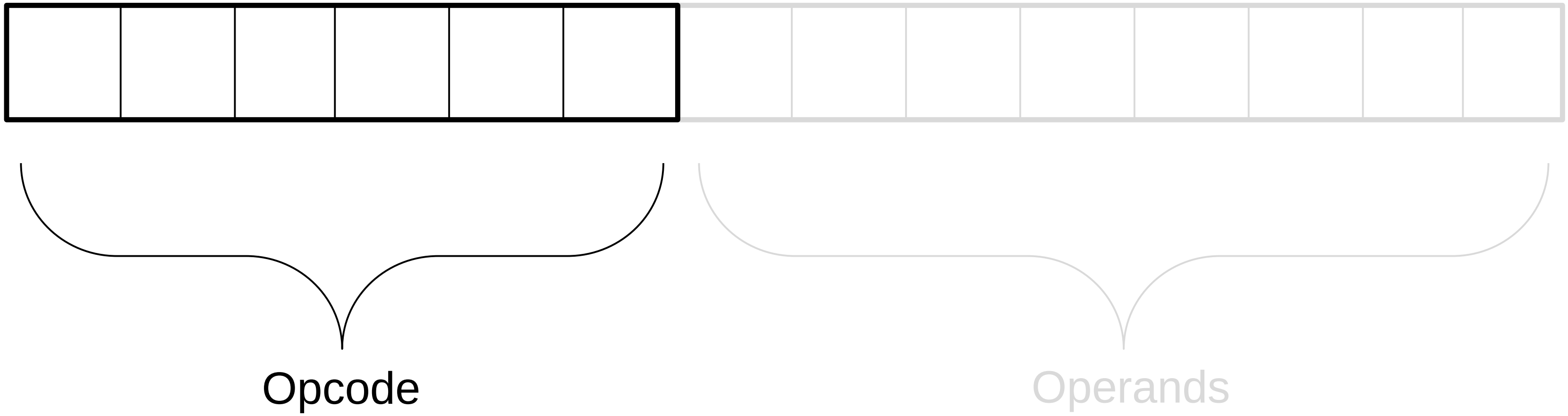
(Literal values, registers…)

- Naïve solution: model each possible combination (opcode, operands) as a class → Too many classes !

DIFFERENCES WITH KEY RECOVERING TEMPLATE

- Divide and conquer is not efficient. Where to divide ?
  - Number of operands is not fixed
  - Size of opcode and operands are not fixed

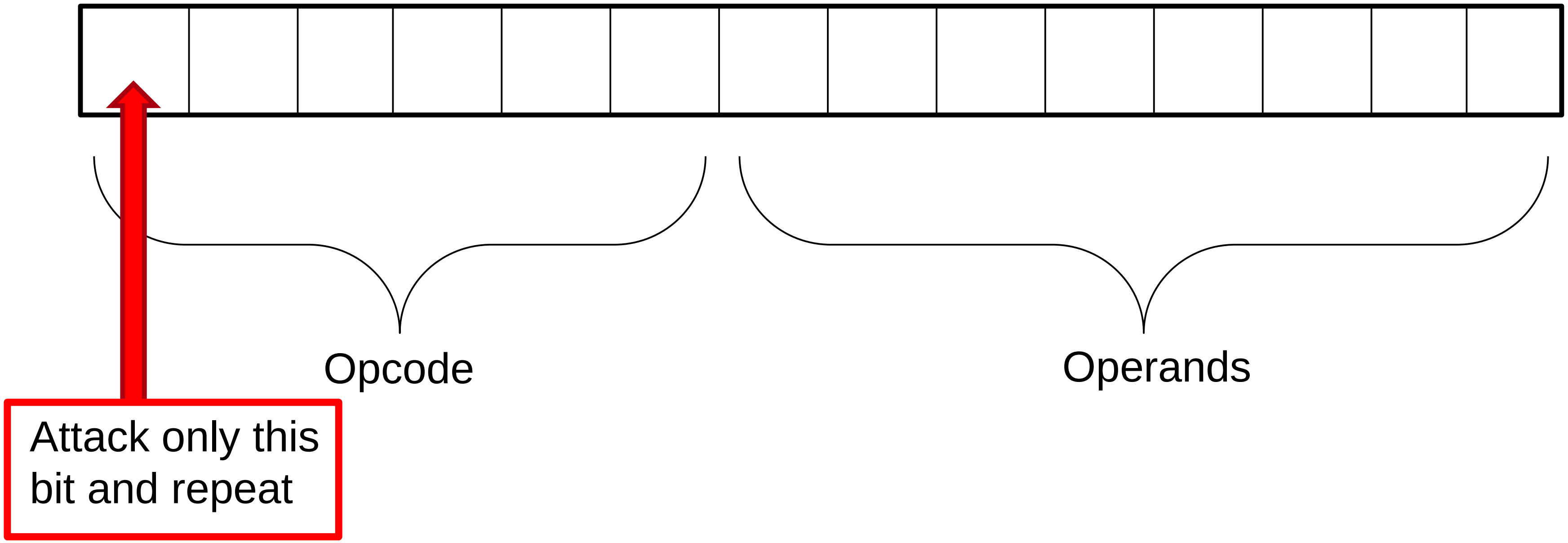
State of the art only focuses the opcodes !



STATE OF THE ART

- Reach a good success rate (90 - 95%) on the PIC (Eisenbarth et al. 2010, Strobel et al. 2015)
- Usually do not recover the operands
- Require a long profiling phase with a lot of data
- Are not scalable to more complex processors with
  - More instructions (encoded on 32 bits)
  - Deeper pipeline

BIT ORIENTED TEMPLATE ATTACK ?



- Only 2 classes by template (0 and 1)
- Profiling can be done on random instructions (correctly labelled)
- We would get the operands as a bonus !

VIABILITY QUESTIONS…

1. Distinguish bit level variation (good enough SNR) ?

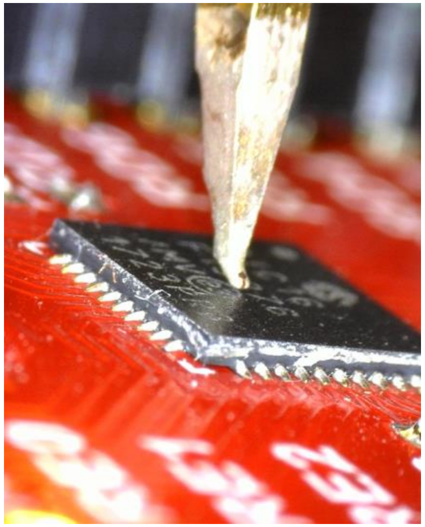
2. Does each bit have its own leakage ?

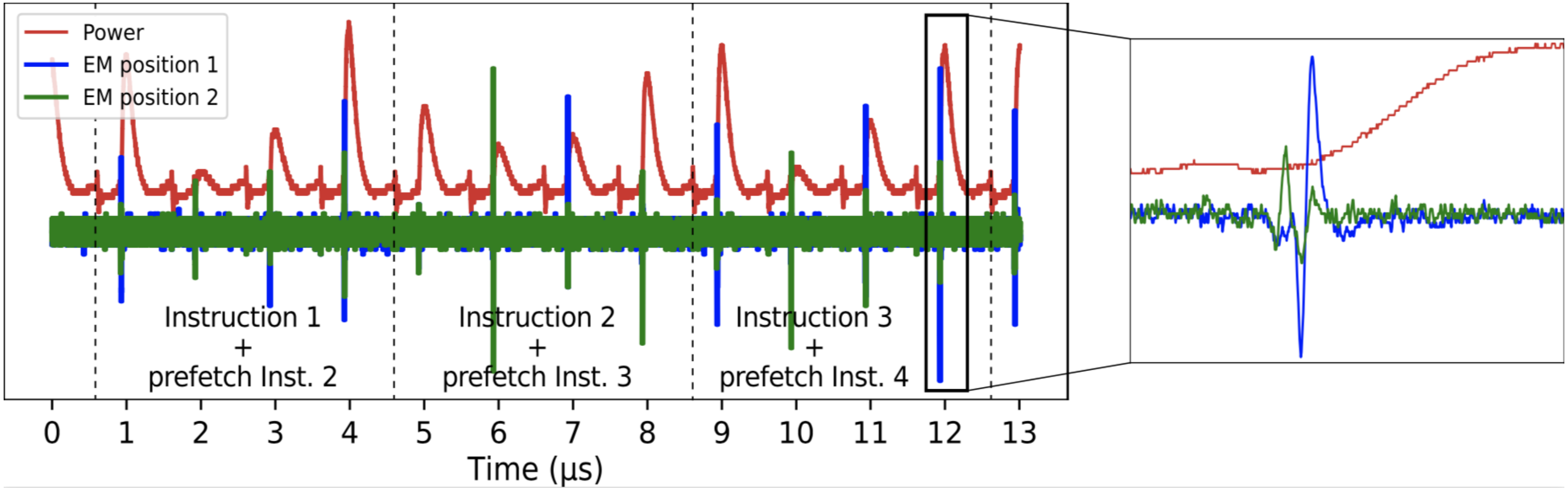
3. Does each bit leak independently ?

4. What is the leakage model ?

EM VS POWER

- ➢ EM + micro-probe → exploit local leakage.
- ➢ Leakage vary with probe position → cartography





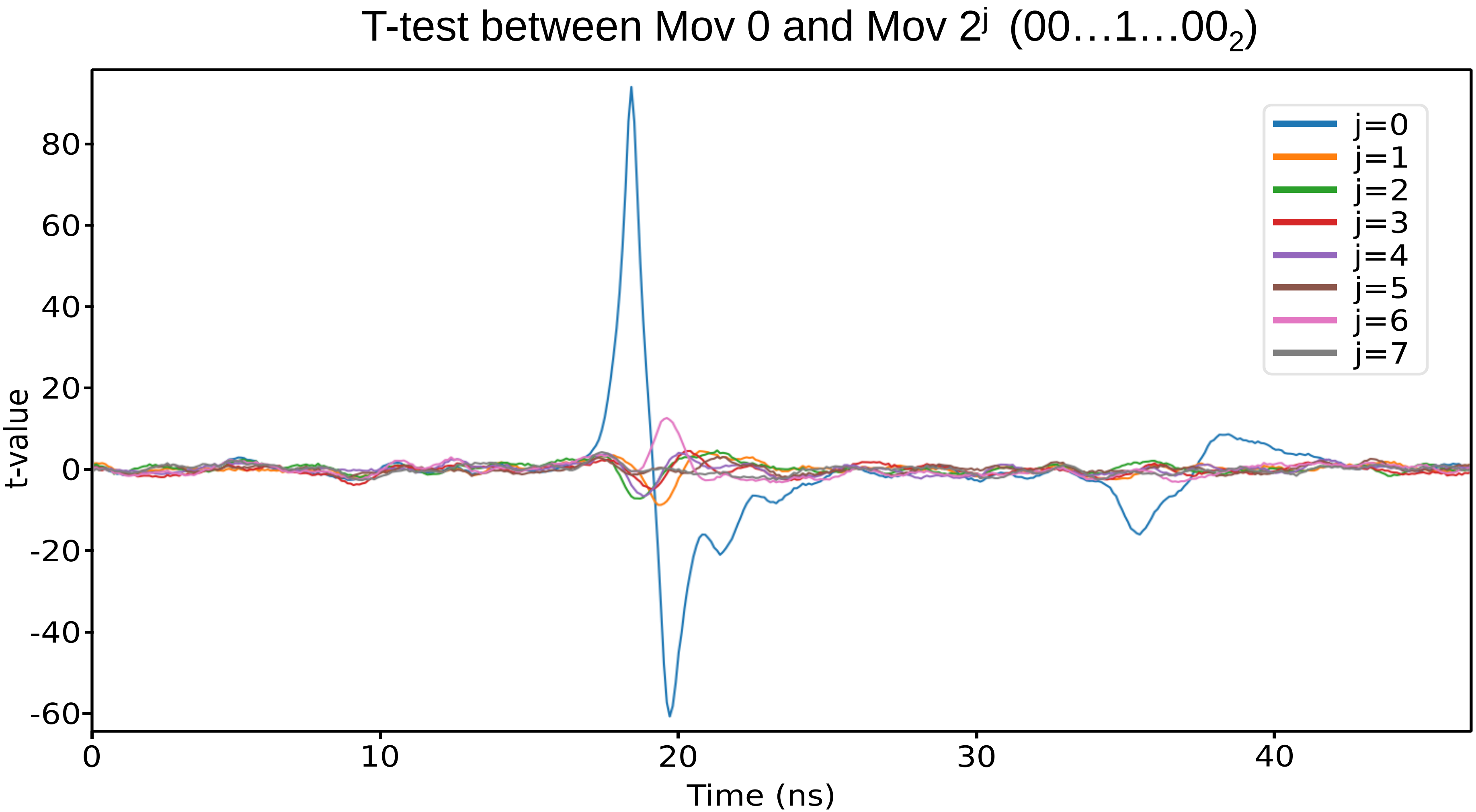
LEAKAGE DETECTION

- We can try the following test …

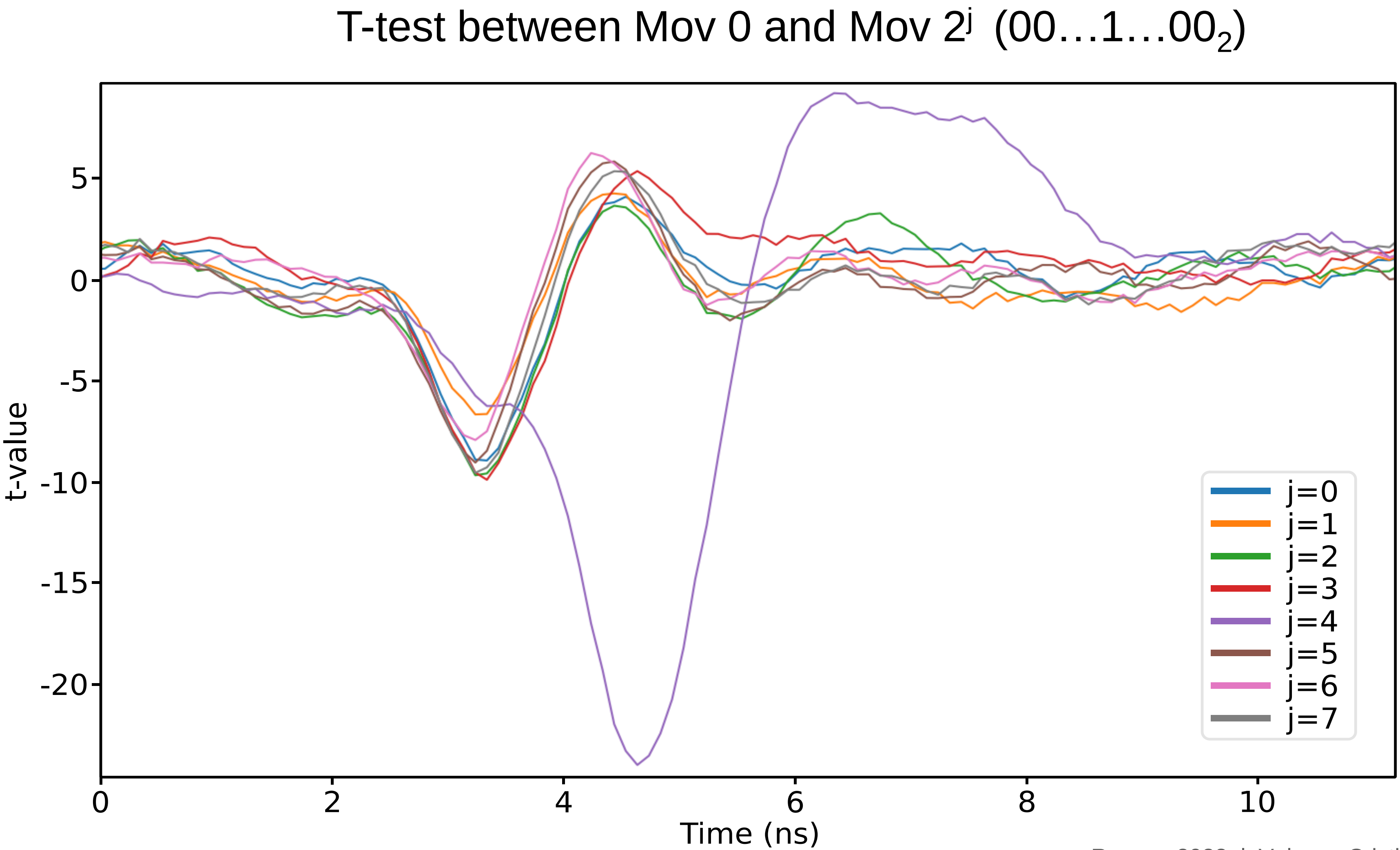
MOV $\underbrace{0\ldots1\ldots0}_{i}$ → Compare with T-test ← MOV 0…0…0

… at different probe position

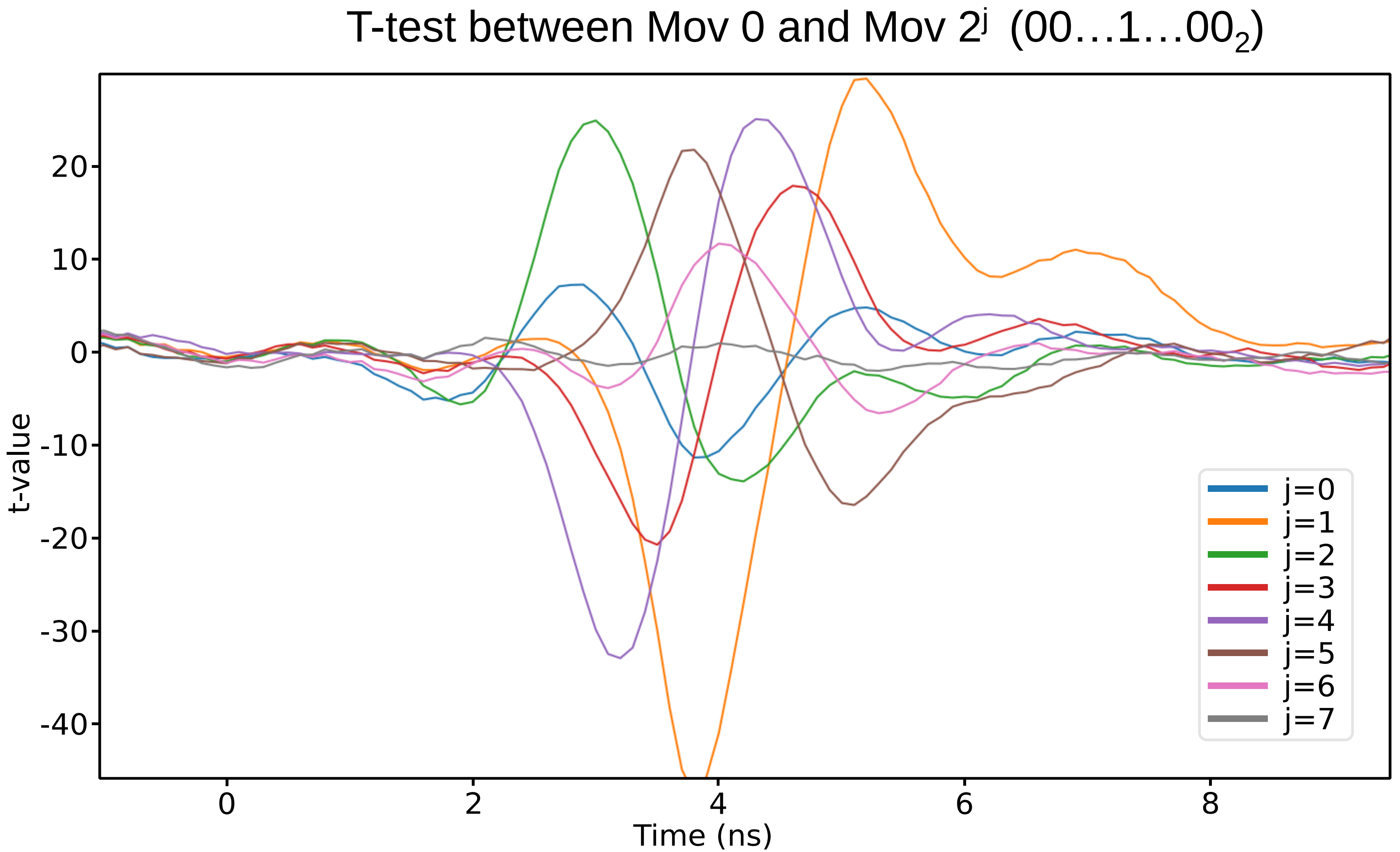
BEST PROBE POSITION FOR BIT 0



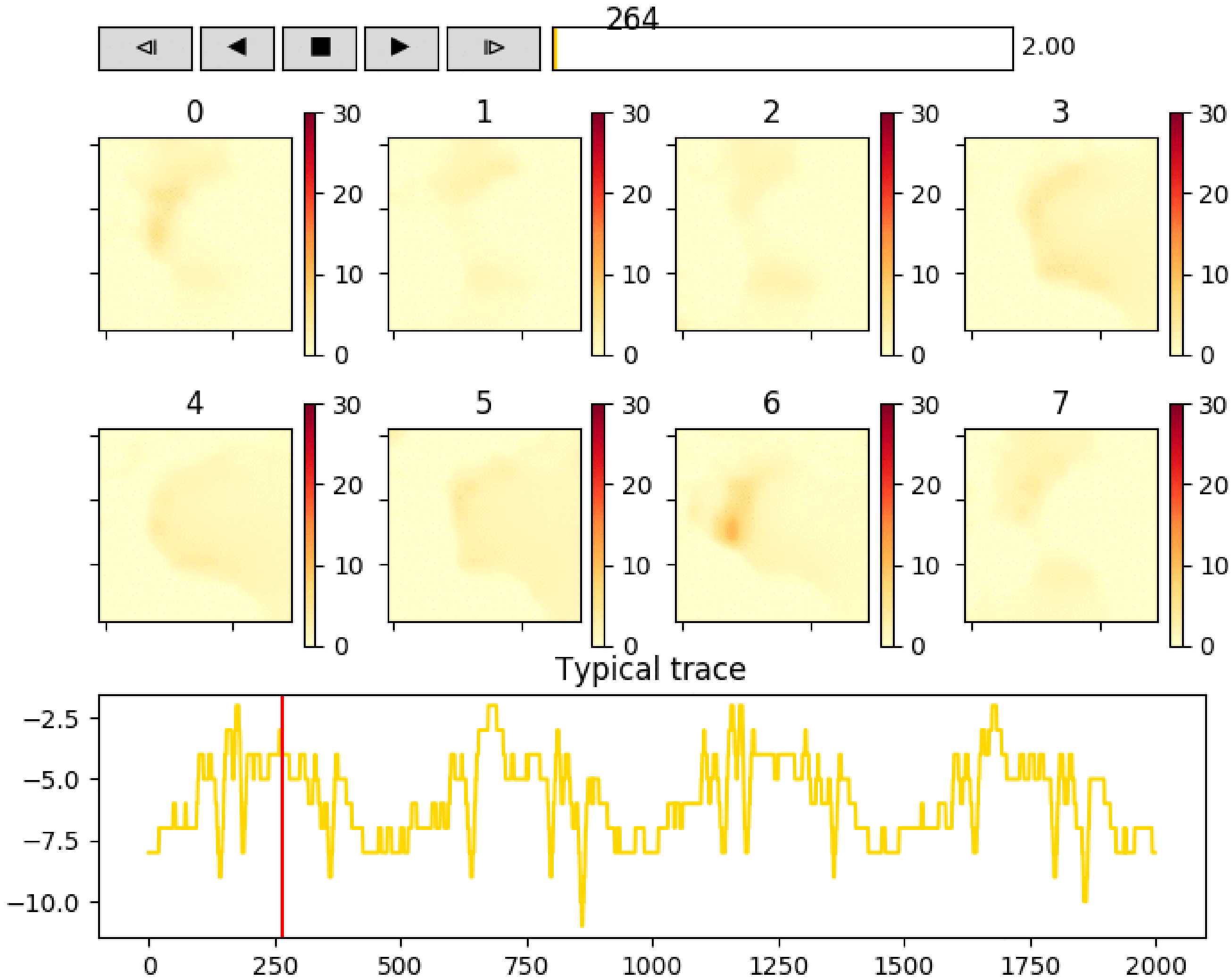
BEST PROBE POSITION FOR BIT 4



… FOR ALL THE BITS

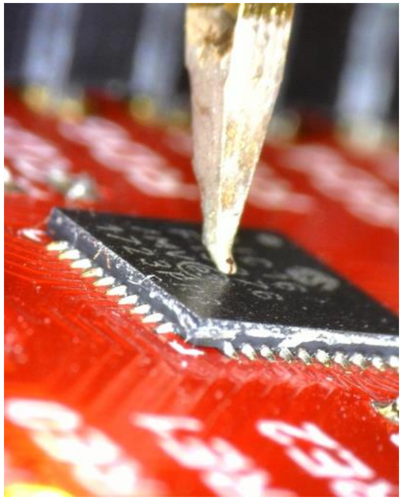


SPATIAL AND TEMPORAL LEAKAGE



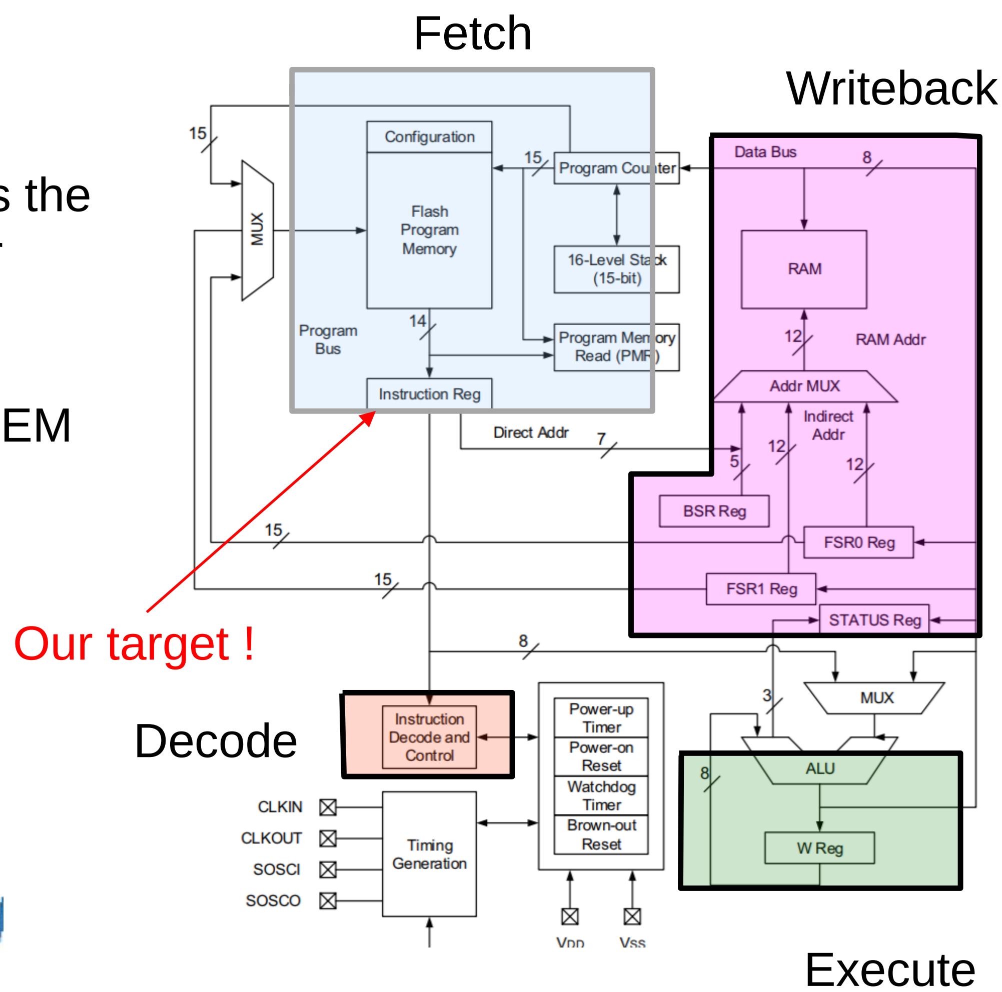
VIABILITY QUESTIONS…

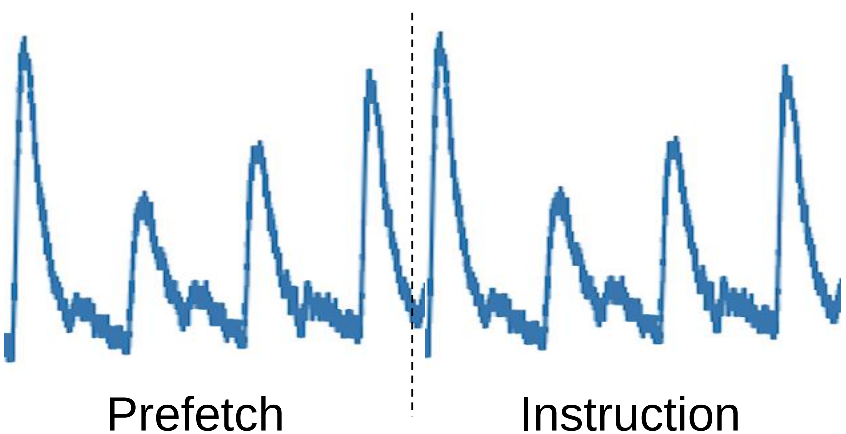
1. Distinguish bit level variation (good enough SNR) ?
2. Does each bit have its own leakage ?
3. Does each bit leak independently ?
4. What is the leakage model ?



PIC16F ARCHITECTURE

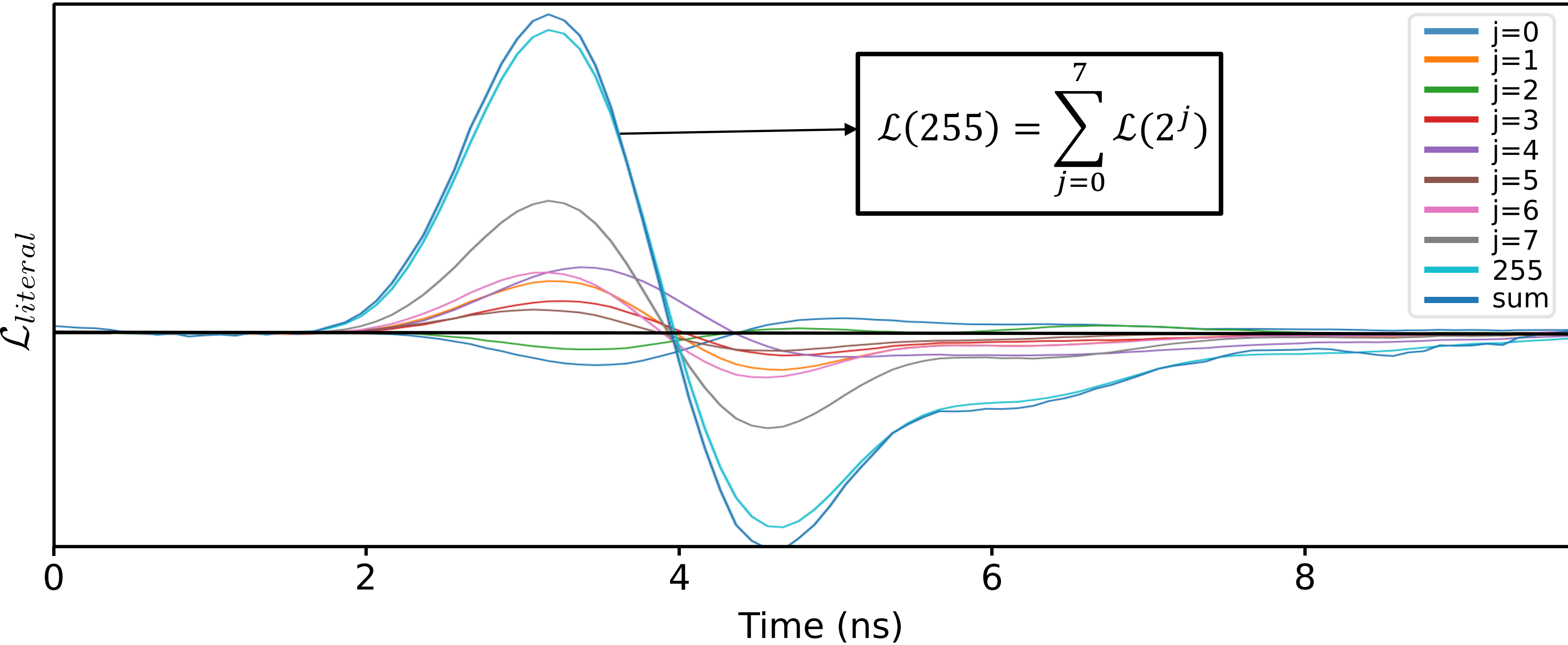
- Attack only the prefecth to focus the updtate of the intruction register
- Each bit may have its own independent contribution to the EM





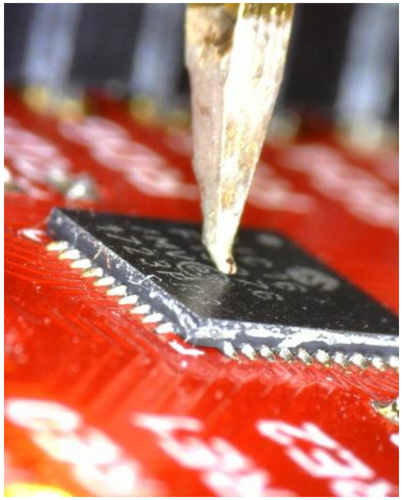
BIT INDEPENDENCE

$$\mathcal{L}(2^j) = Leakage(Mov\ 2^j) - Leakage\ (Mov\ 0)$$



VIABILITY QUESTIONS…

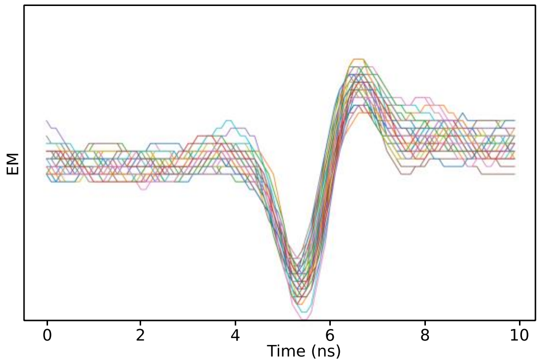
1. Distinguish bit level variation (good enough SNR) ?
2. Does each bit have its own leakage ?
3. Does each bit leak independently ?
4. What is the leakage model ?



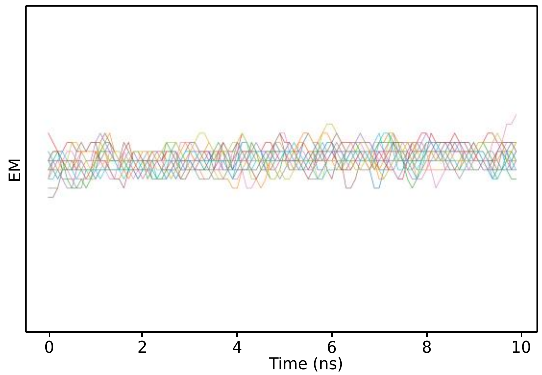
HAMMING DISTANCE LEAKAGE MODEL

- Leakage depends on the previous state of the bit. Power consumption is caused by a bit flip.

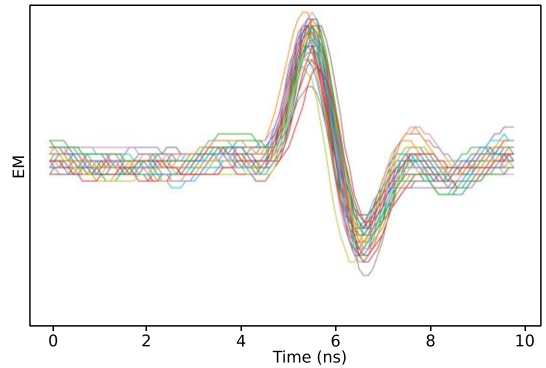
- EM allows to get the direction of the transition: 0→1 or 1→0



0 → 1



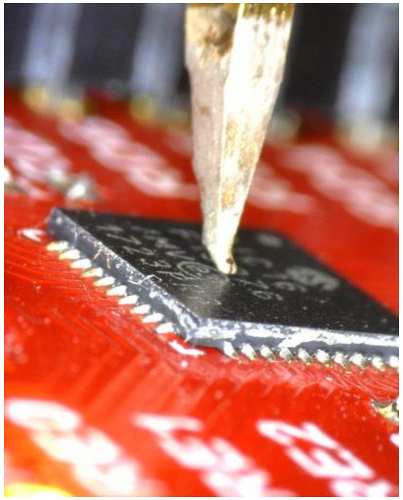
0 → 0
1 → 1



1 → 0

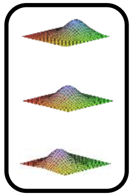
VIABILITY QUESTIONS…

1. Distinguish bit level variation (good enough SNR) ?
2. Does each bit have its own leakage ?
3. Does each bit leak independently ?
4. What is the leakage model ?



OUR METHODOLOGY

1. Build a 3 classes template (0 → 1, 1 → 0, constant)



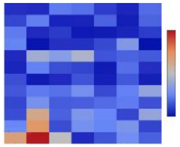
2. Apply it to get a sequence of transitions on the attack data $[t_1, t_2, \ldots, t_n]$

3. Convert it to a sequence of bits (Viterbi algorithm) $[b_1, b_2, \ldots, b_n]$

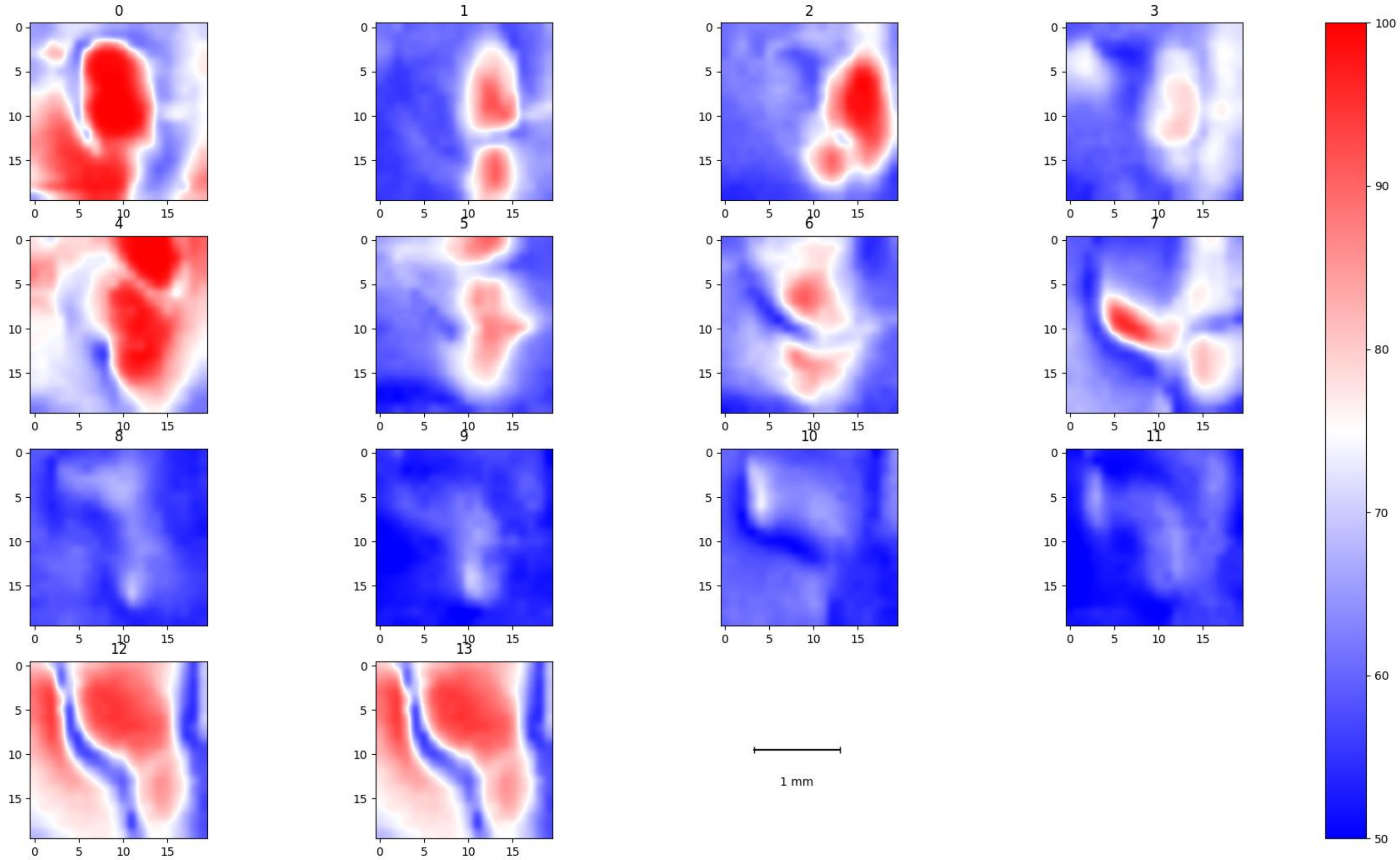
4. Measure your success rate

70%

5. Repeat for all bit at every probe position on a grid



CARTOGRAPHY FOR ALL BITS



Success rate

USE MULTIPLE POSITIONS ?

How to increase the success rate of the attack ?

Combine the information from multiple probe positions !

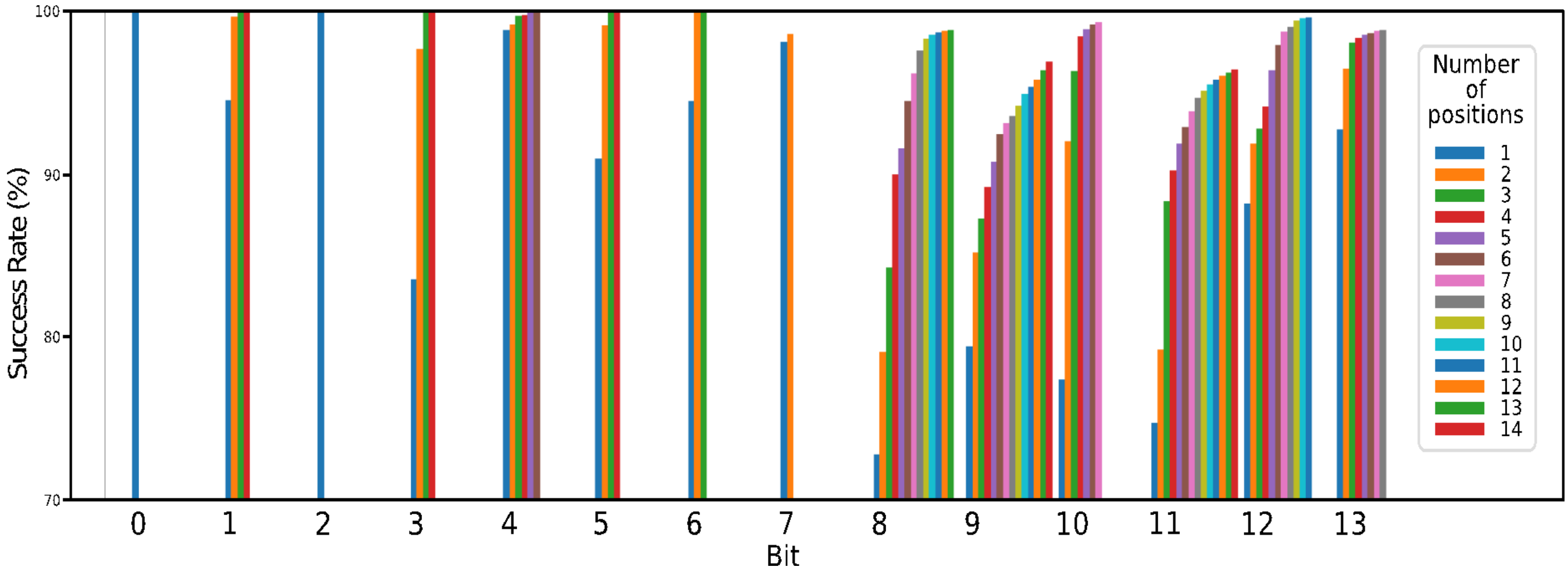
USE MULTIPLE POSITIONS ?

How to increase the success rate of the attack ?

Combine the information from multiple probe positions !

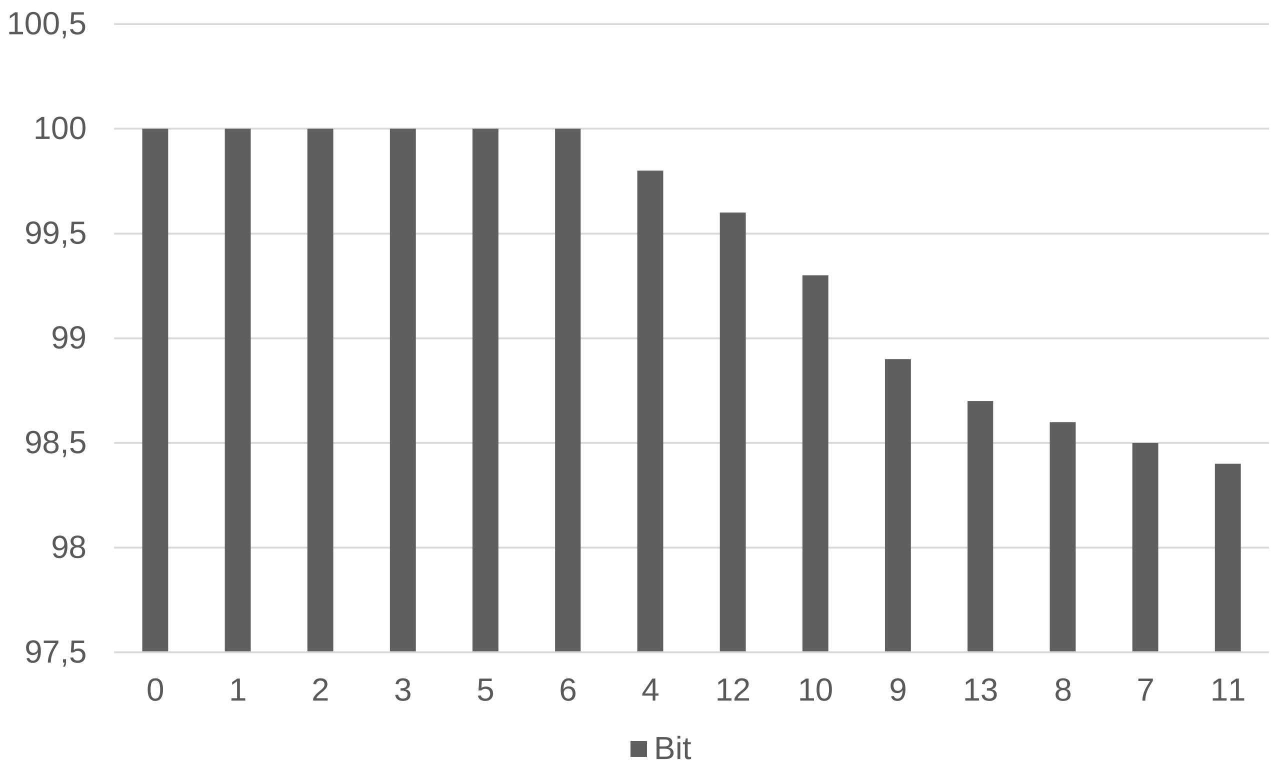
Concatenate the traces and apply the template attack as if it was a single trace.

USE MULTIPLE POSITIONS ?



- We selected a subset of up to 14 positions per bit
- Success rate converges towards 100%

RESULT ON A RANDOM PROGRAM



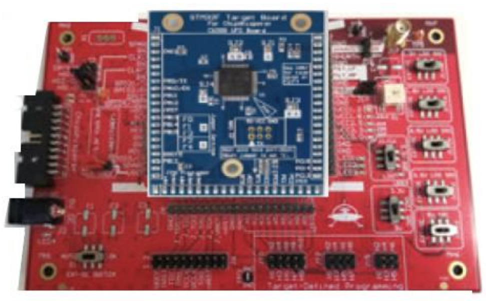
Average: 99.4%

95% of the instruction were recovered without any fault on the 14 bits !

A BIT LEVEL DISASSEMBLER…

- Monobit approach
  - Easier to train
  - Potentially scalable
  - Gives usefull information even in case of error
- Exploit local leakage
  - Different leakage between the bits
  - Find the best probe positions for each bit
  - Combine information from multiple positions
- Our attack is portable between 2 targets

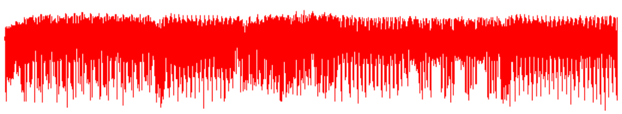
LET'S TAKE A STEP BACK…



Target board

Side channel
measurments

Oscilloscope



Side channel Trace

⋮

Xor r2, r3, r2

Mov r0, r1

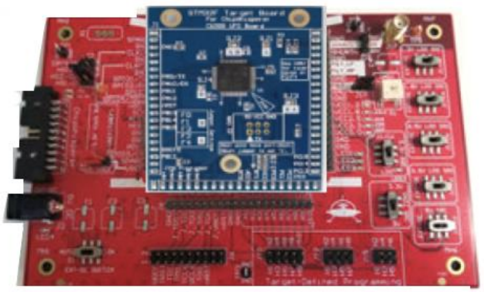
Sub r1, r2, r3

⋮

Instruction flow

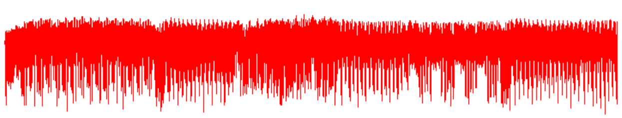
LET'S TAKE A STEP BACK…

Side channel measurments



Target board

Oscilloscope

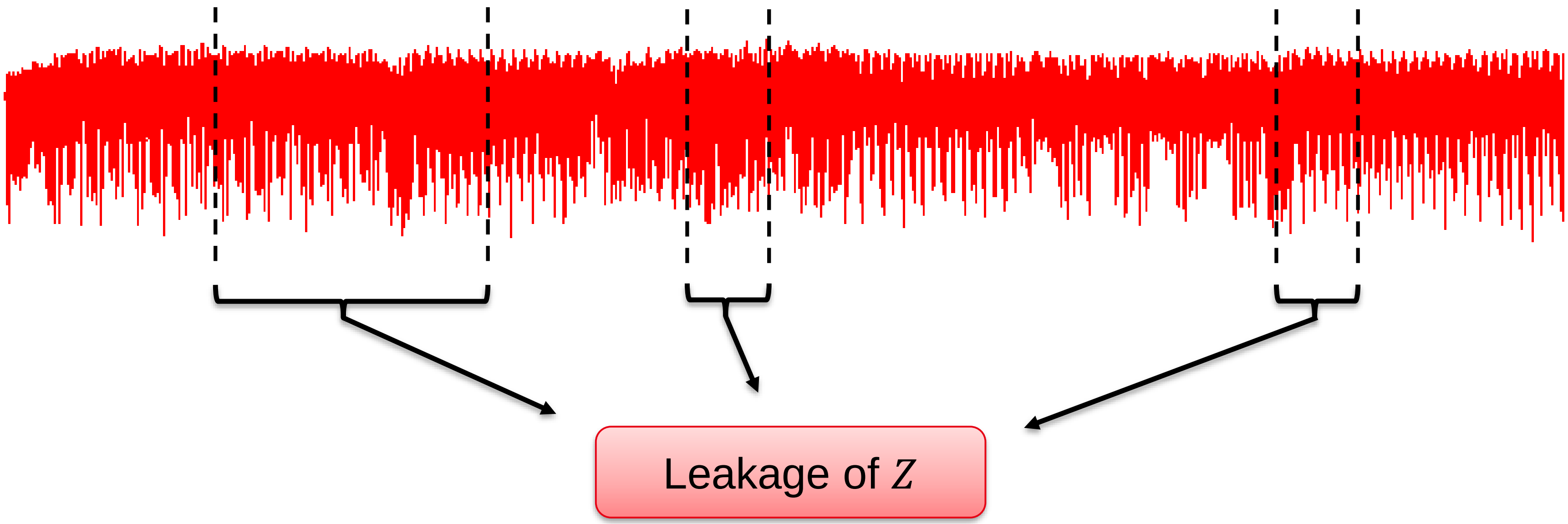


Side channel Trace

Recover information about a secret

SIDE CHANNEL GENERIC GOAL

EM trace of an AES



- Sensitive variables such as $Z = K \oplus P$ are processed during the execution
- The goal is to exploit this leakage to extract information about these variables

Being fundamentally bounded by the existing information…

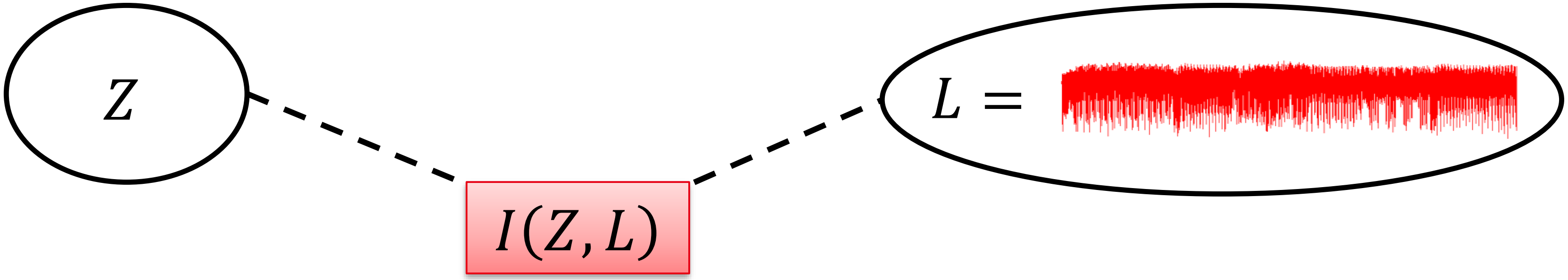
MUTUAL INFORMATION IN SIDE CHANNEL

Z = sensitive variable
L = Leakage (trace)

$$I(Z,L) = \mathcal{H}(Z) - \mathcal{H}(Z \mid L)$$

- Mutual information between Z and L can be seen as an absolute leakage quantification

- It can also be seen as an upper bound of information an attacker could potentially retrieve about the secret



GOALS OF THE DIFFERENT ACTORS IN TERMS OF MI

Designers

Aim at implementing countermeasures to decrease $I(Z, L)$ as far as possible with efficiency constraints

Evaluators

Aim at estimating $I(Z, L)$ to get an objective leakage metric in a worst case scenario

Attackers

Aim at exploiting the maximum information from $I(Z, L)$ about a secret to retrieve it

But …. We don't know how to compute $I(Z, L)$ because the current estimation techniques does not scale to higher dimension variables such as L

ESTIMATE MI USING DEEP LEARNING ?

MINE is a new technique that comes from the pure machine learning community. It uses deep learning to estimate MI in high dimension.

The general idea is to transform the MI computation into a maximization problem and to use backpropagation to solve it

$$I(Z,L) = D_{KL}(\mathbb{P}_{(Z,L)} \parallel \mathbb{P}_Z \otimes \mathbb{P}_L)$$

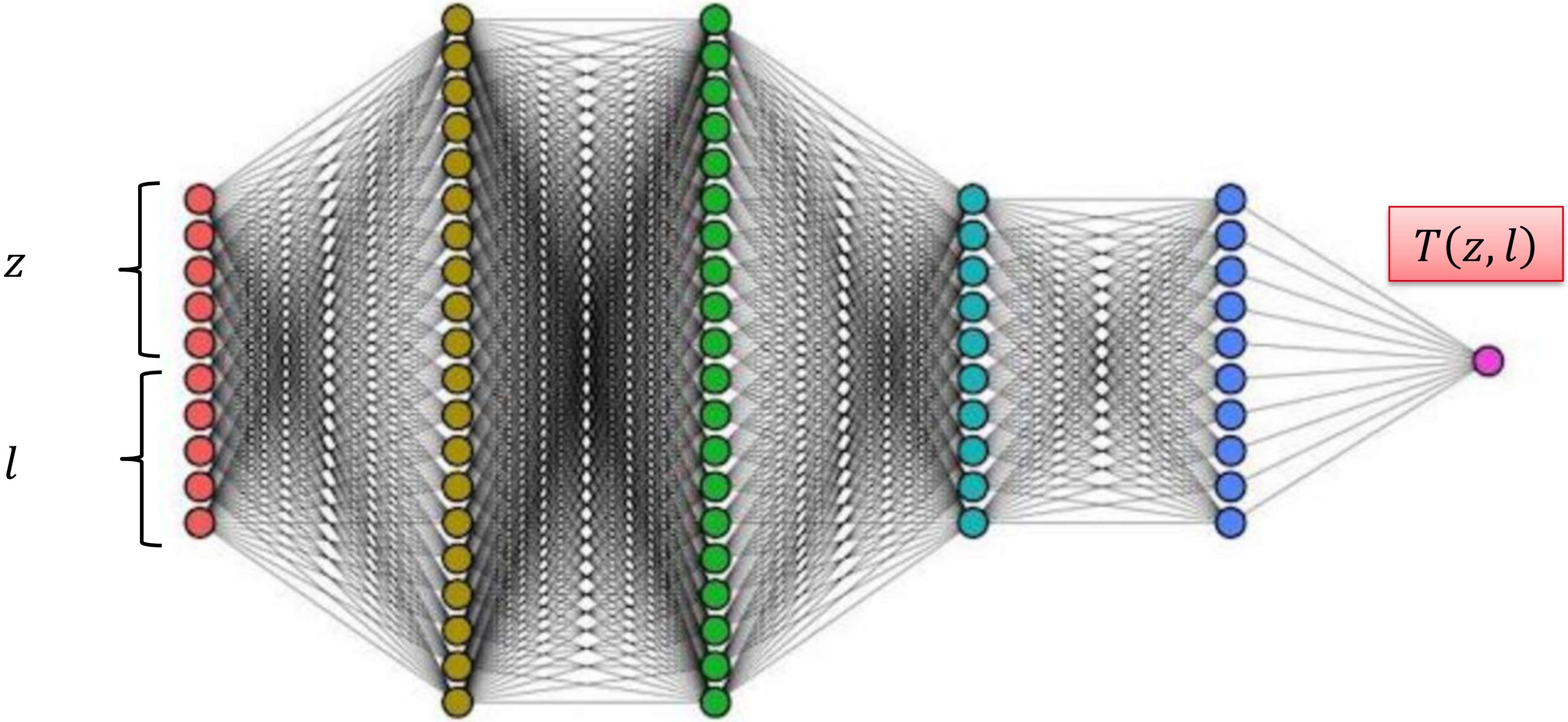
Loss function

$$D_{KL}(\mathbb{P}_X || \mathbb{P}_Y) = \sup_{T:\Omega\to\mathbb{R}} \boxed{\mathbb{E}_X[T(X)] - log(\mathbb{E}_Y[e^{T(Y)}])}$$

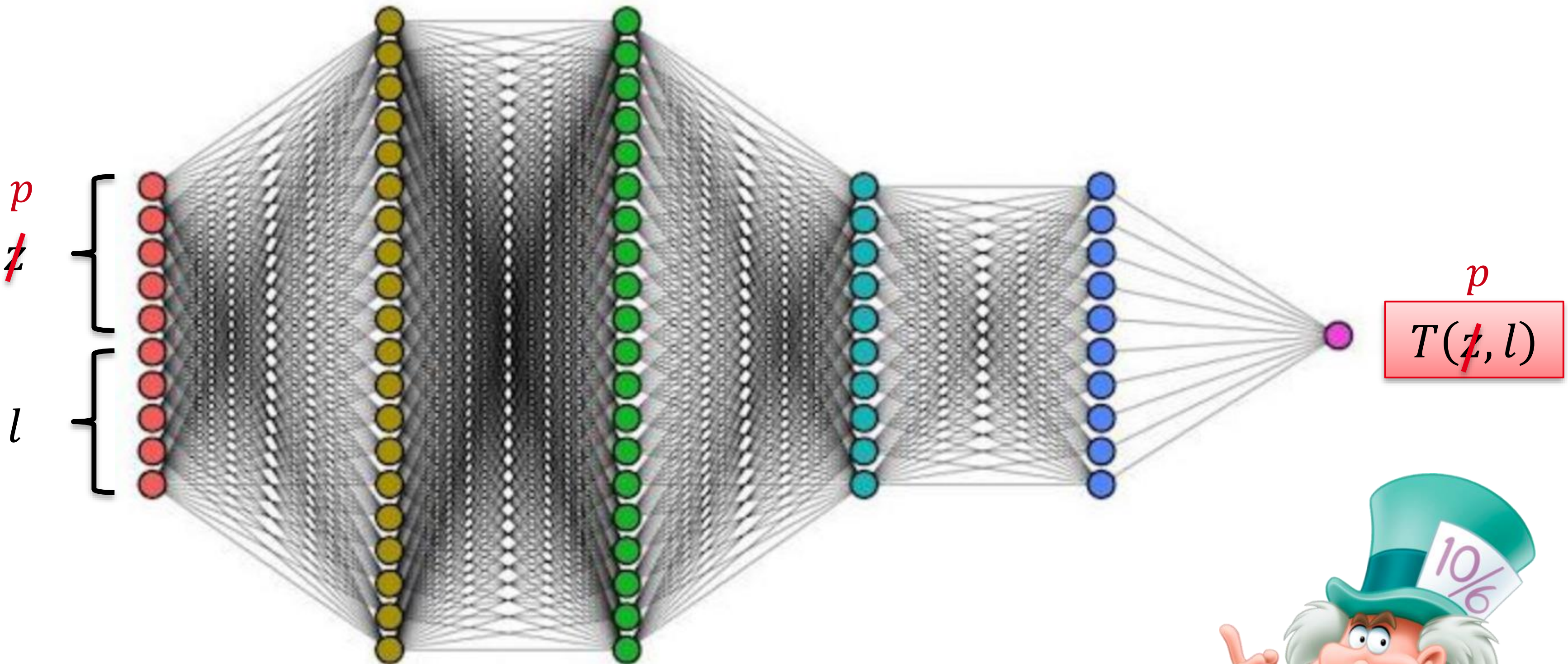
- Search for the maximum over all the functions T representable by a neural network
- Training can be done via gradient ascent
- The loss function should converge towards $I(Z,L)$

MI ESTIMATION NETWORK

$$I(Z,L) = \sup_{T:\,\Omega \to \mathbb{R}} \mathbb{E}_{\mathbb{P}_{Z\ L}}[T(Z,L)] - log(\mathbb{E}_{\mathbb{P}_Z \otimes \mathbb{P}_L)}[e^{T(Z,L)}])$$



DOES ONE REALLY NEED Z ?



Do we really need the knowledge of the key ?

No ! MI stays constant bijective transformation of the input variables !

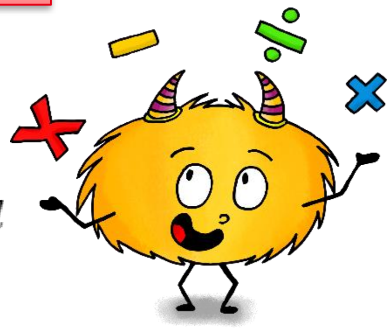
$$I(Z, L) = I(f_k(P), L) = I(P, L)$$

EVALUATE THIS TECHNIQUE ON SYNTHETIC TRACES

- Generate synthetic traces with a Hamming weight leakage model
- Each trace is composed of:
  - n_l leakage samples leaking the Hamming weight of Z with some Gaussian noise $\mathcal{N}(0, \sigma)$
  - n_r random non-informative samples
- Since the leakage is controlled the true MI can be computed analytically:

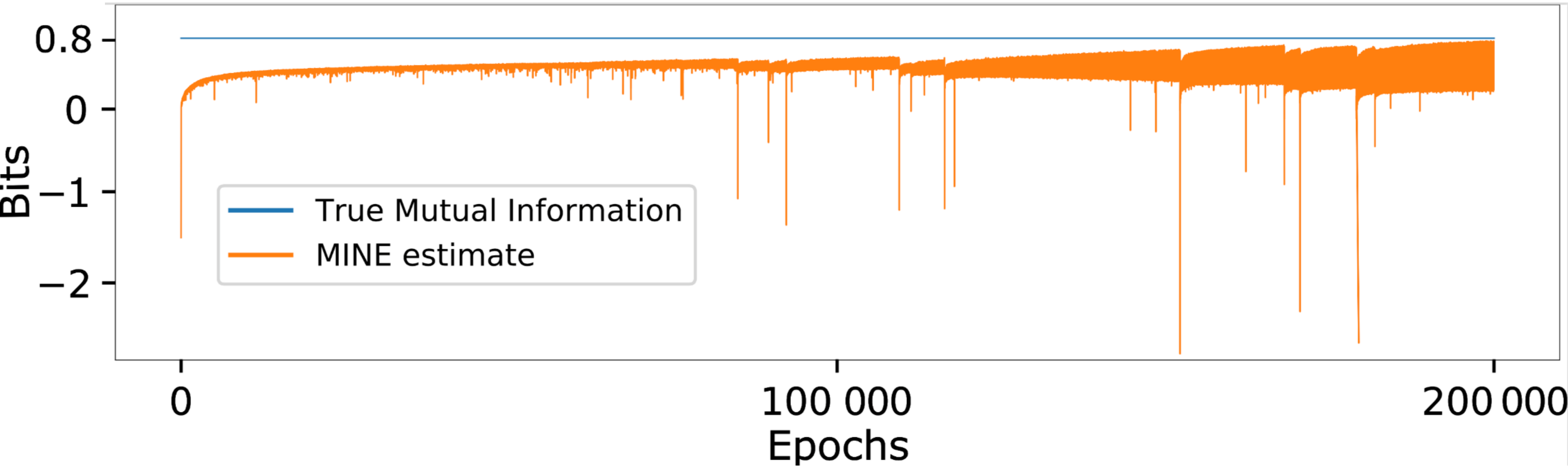
$$\mathcal{I}(Z, L) = H(Z) - H(Z|L)$$
$$= 8 - \sum_{z=0}^{255} \int_{-\infty}^{\infty} \Pr(z, l) \cdot log_2\left(\frac{1}{\Pr(z|l)}\right) dl$$
$$= 8 - \sum_{z=0}^{255} \int_{-\infty}^{\infty} \frac{1}{2^8} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(l - HW(z))^2} \cdot log_2\left(\frac{\sum_{z'=0}^{255} e^{-\frac{1}{2}(l-HW(z'))^2}}{e^{-\frac{1}{2}(l-HW(z))^2}}\right) dl$$
$$\approx 0.8 \text{ bits}$$

$n_l = 1, n_r = 0, \sigma = 1$



EVALUATE THIS TECHNIQUE ON SYNTHETIC TRACES

$n_l = 1, n_r = 0, \sigma = 1$



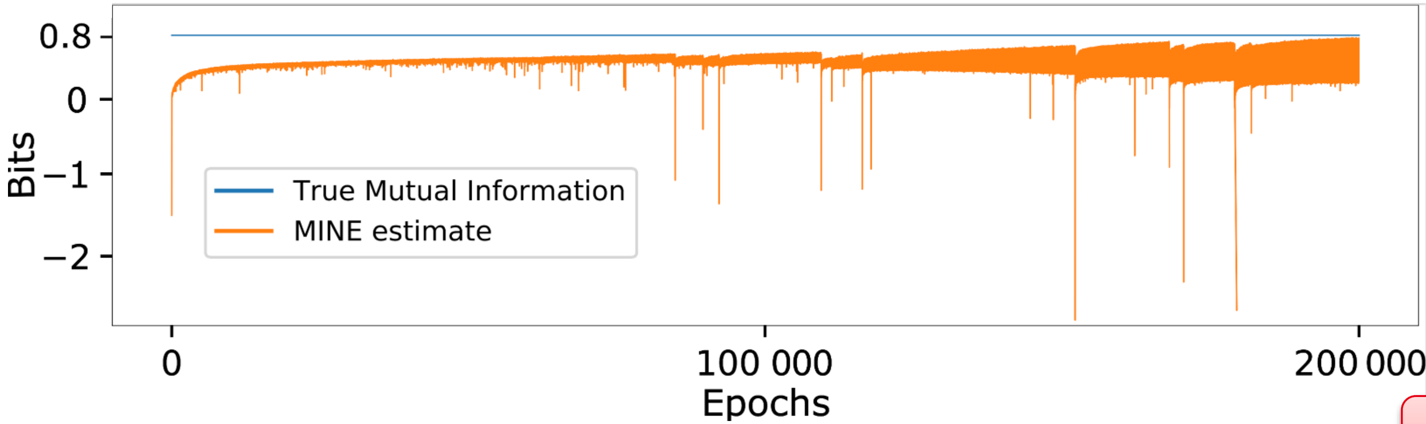
- ➢ A straight application of MINE is not of any use for side channel…

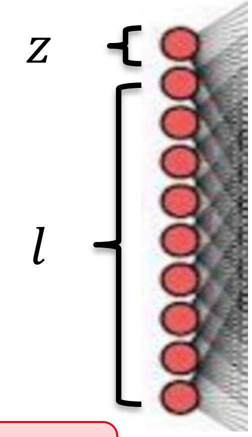
Why is it that hard for the network to train in this context ?

INPUT DECOMPRESSION

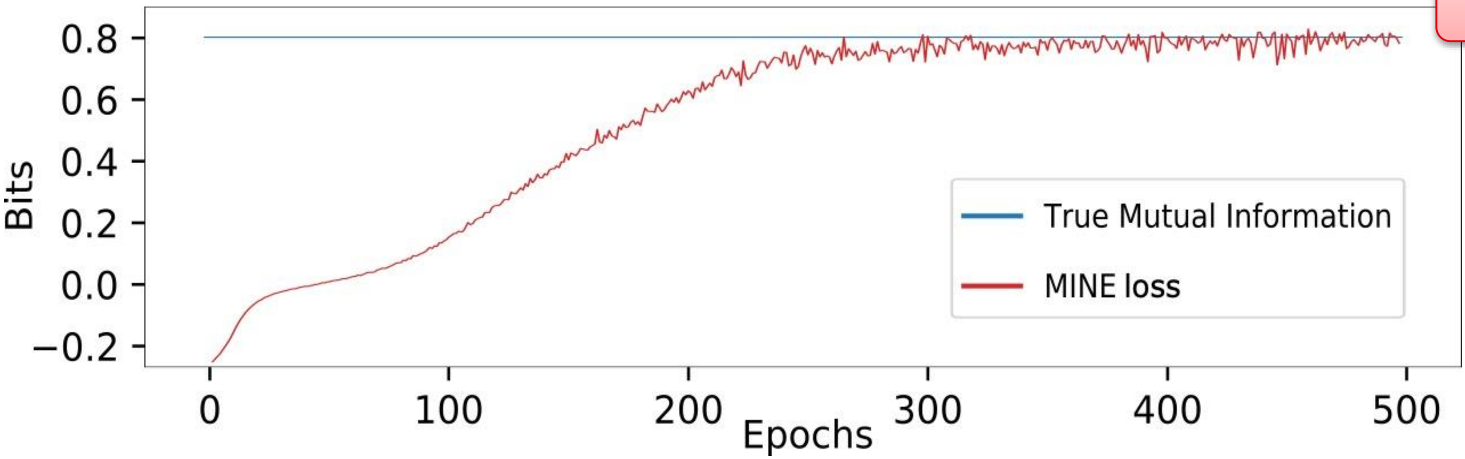
$n_l = 1, n_r = 0, \sigma = 1$

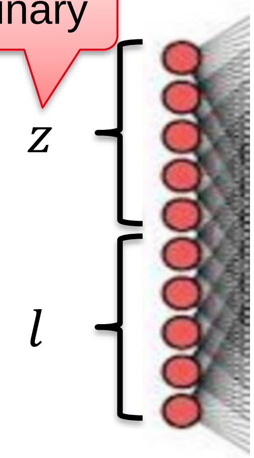
Input layer





In binary

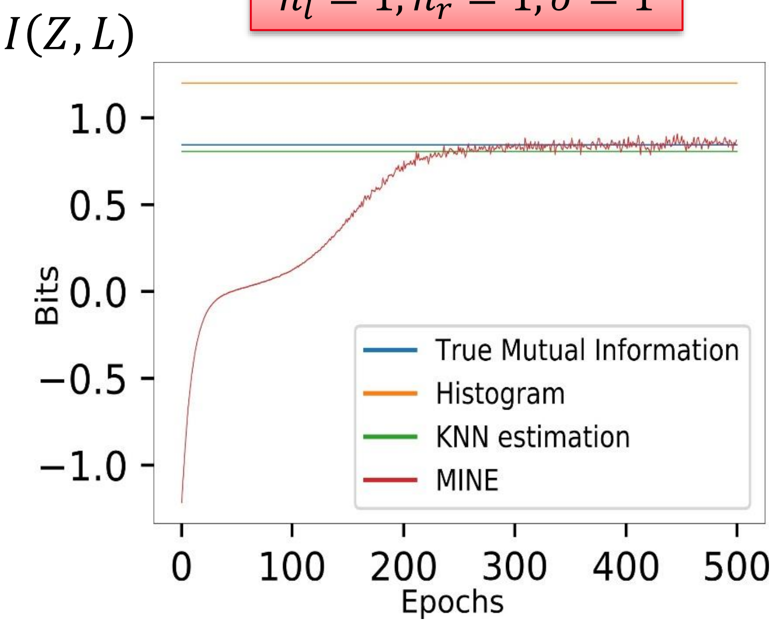




Close z values does not carry the same information at all ! Ex: 127 and 128

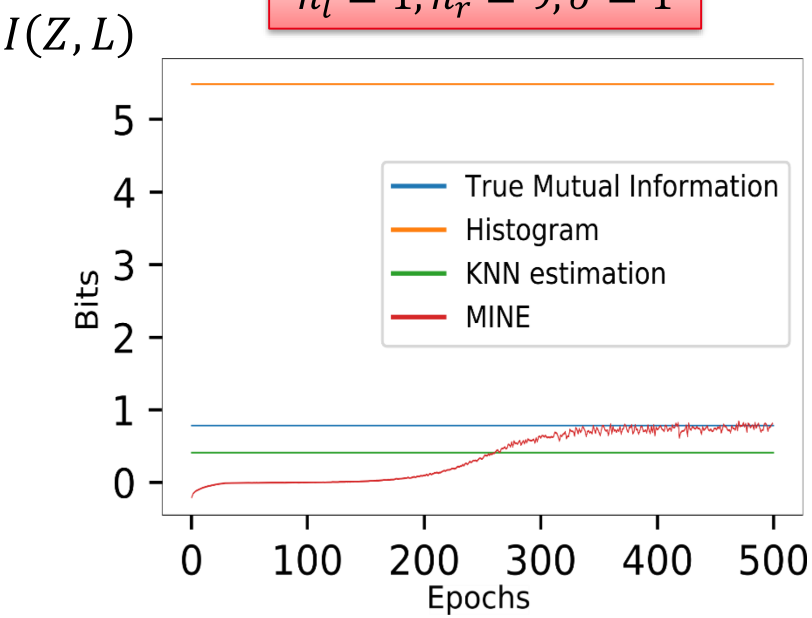
COMPARISON OF MINE WITH CLASSICAL ESTIMATORS

$n_l = 1, n_r = 1, \sigma = 1$



Trace dimension = 2

$n_l = 1, n_r = 9, \sigma = 1$



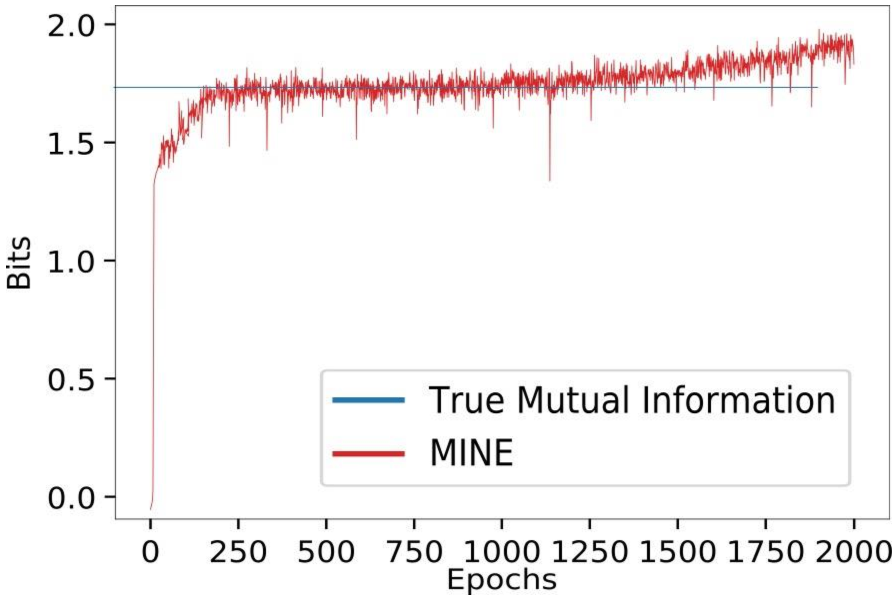
Trace dimension = 10

- MINE always converges towards the true MI even in higher dimension which is not the case of classical estimators…

OVERFITTING ?

$I(Z,L)$

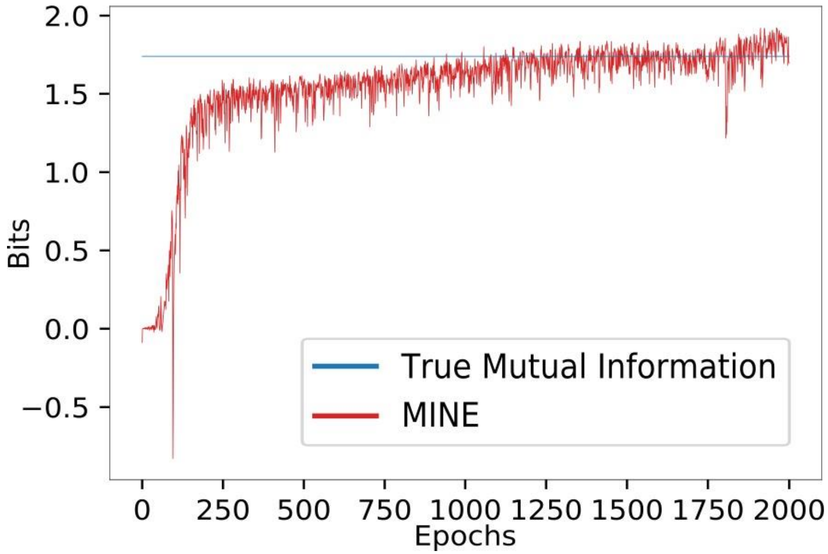
$n_l = 5, n_r = 10, \sigma = 1$



Trace dimension = 15

$I(Z,L)$

$n_l = 5, n_r = 995, \sigma = 1$



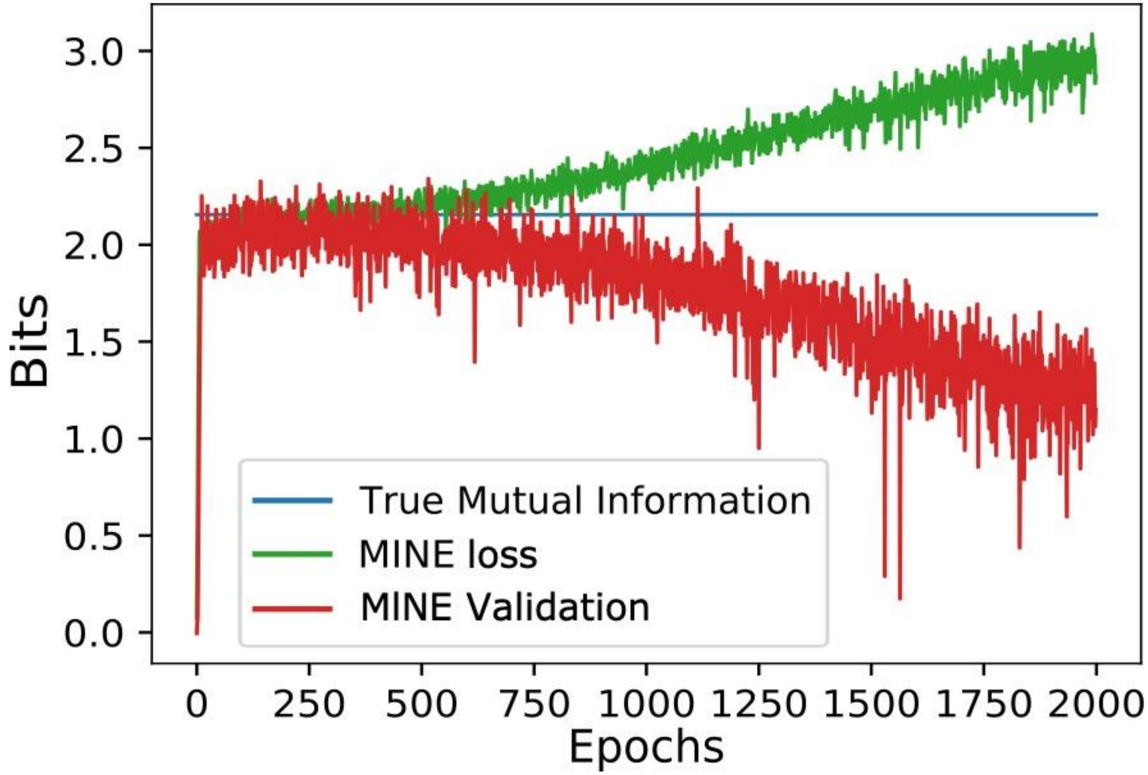
Trace dimension = 1000

It is weird ! It was a supremum in the formula…

- When the training last for a very long time MINE may overfit
- An in depth analysis of this problem can be found in the paper

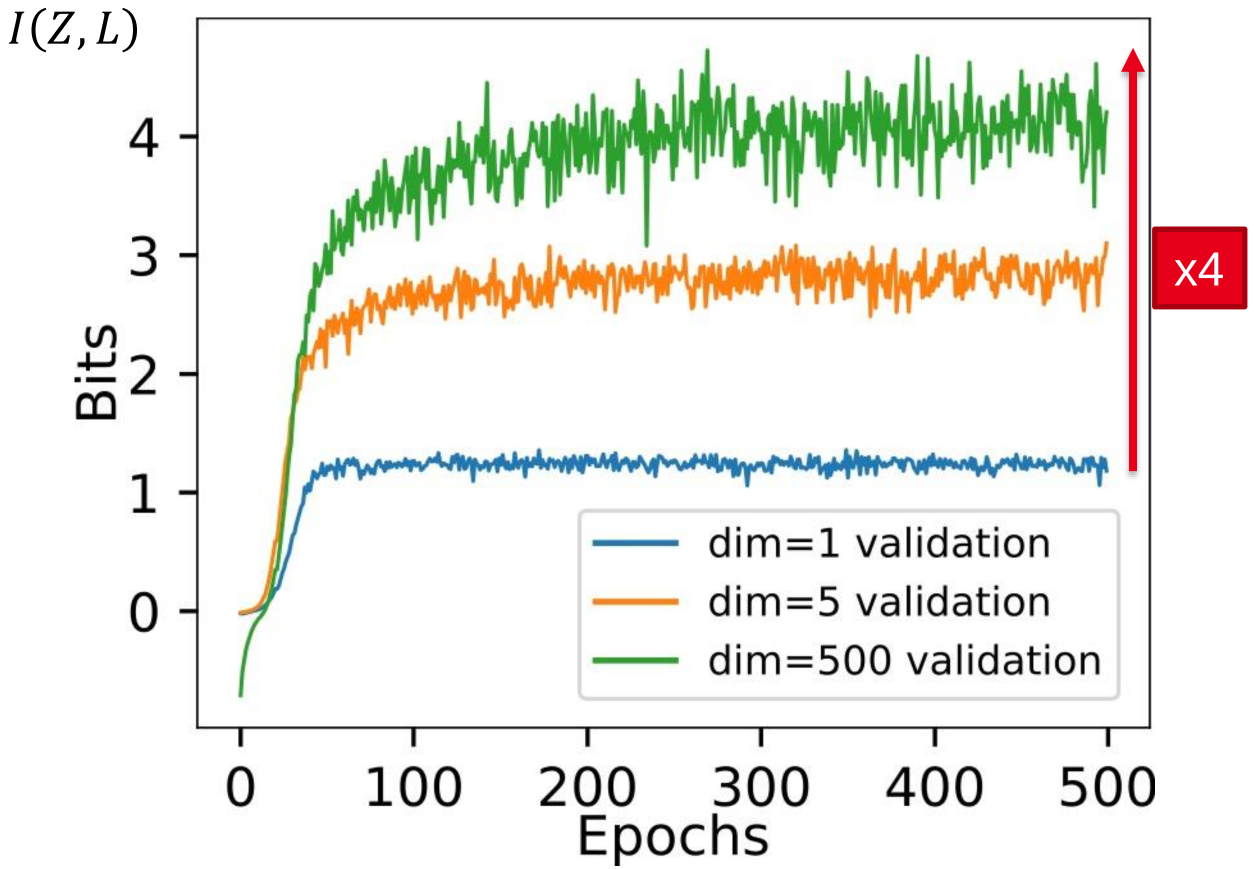
VALIDATION LOSS FUNCTION

$I(Z, L)$



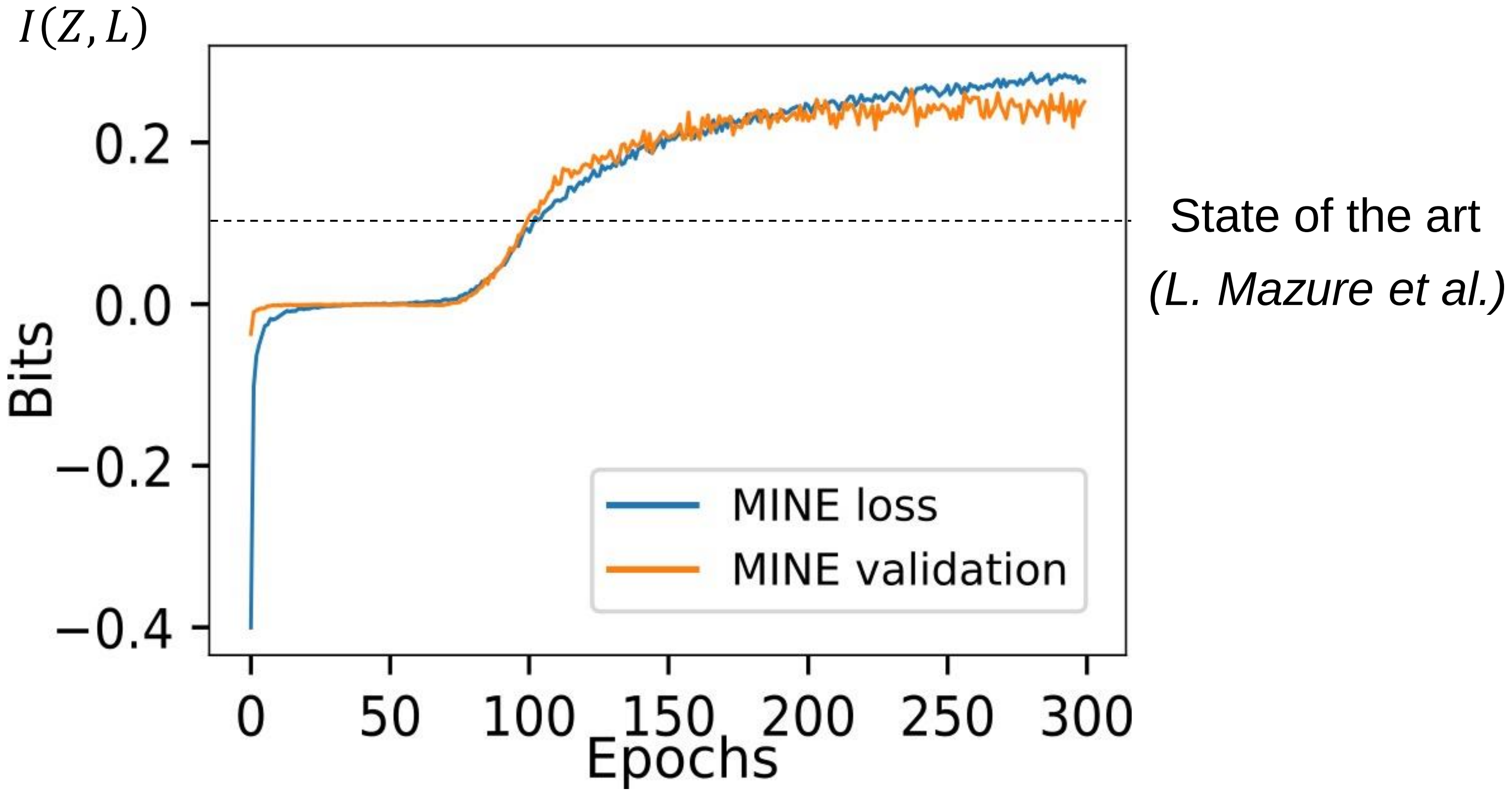
- Split the dataset into a training and a validation one
- Only update the network's parameters from the training dataset but compute the loss function for both datasets

MINE ON REAL LIFE EXPERIMENT (UNPROTECTED AES)



- We have kept the n-th samples with the highest SNR
- This shows that the more samples kept in the analysis, the more information retrieved

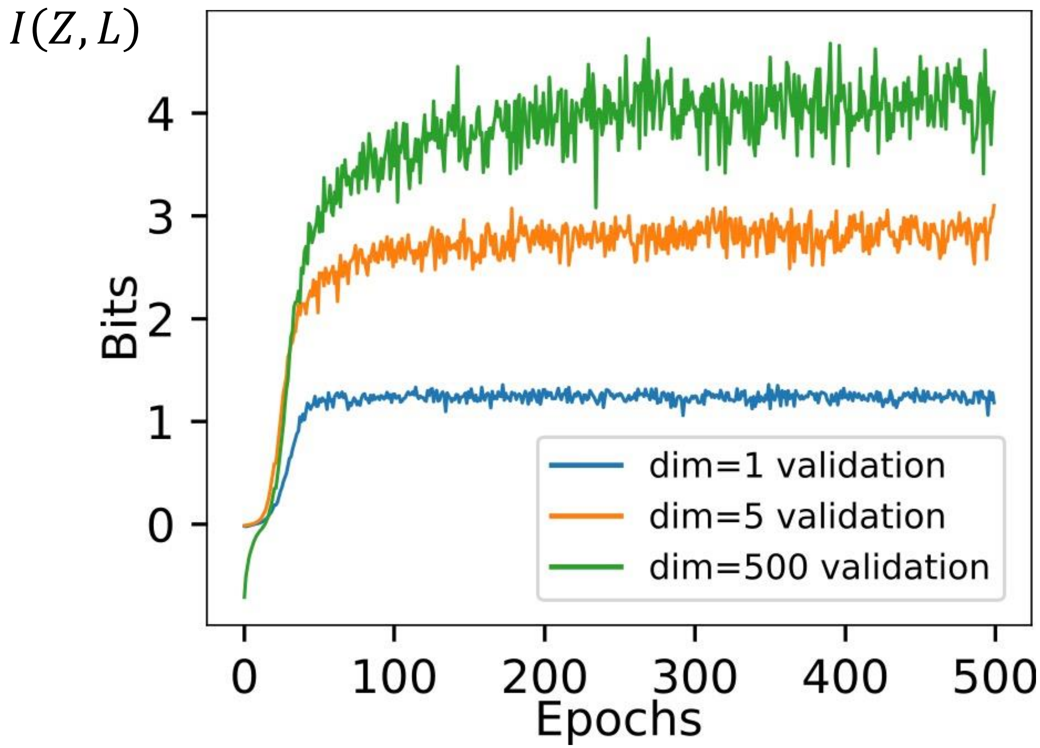
MINE AGAINST MASKED IMPLEMENTATION ? (ASCAD)



- MINE can combine samples and detect higher order leakages
- Mazure *et al.* claimed an information of 0.07 bits on this dataset, we obtained 0.2

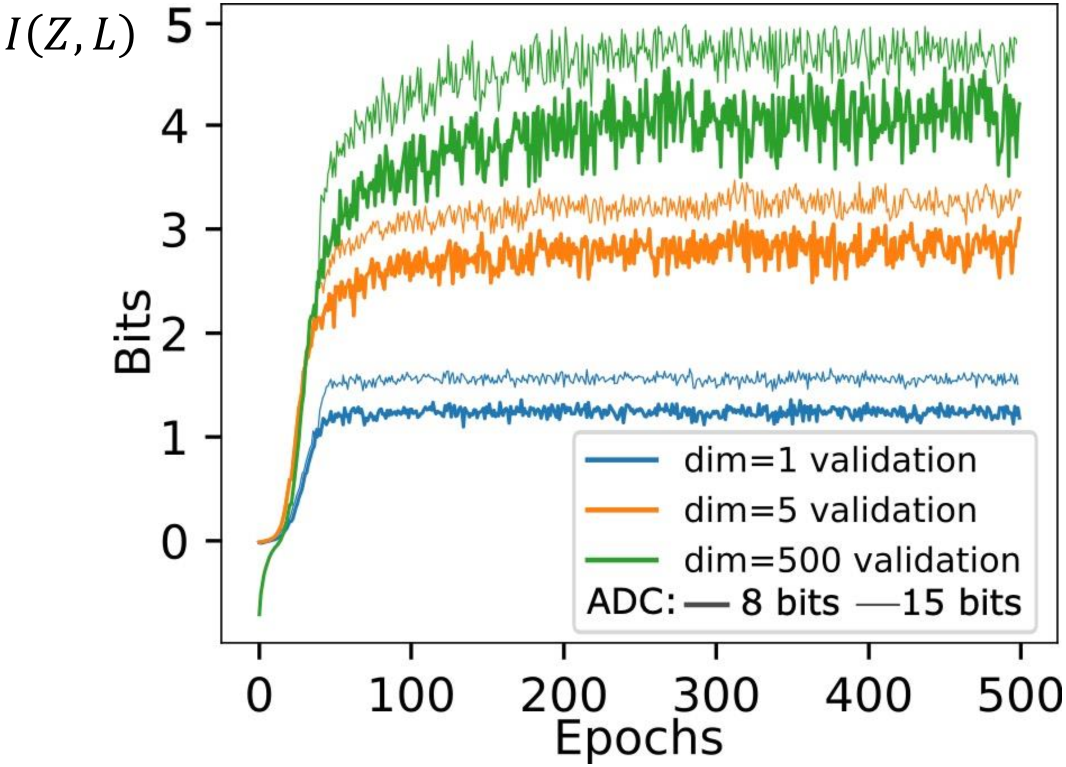
MINE COMPARISON INTEREST

« Is it really worth it to buy the newest scope with the enhanced ADC precision ? »



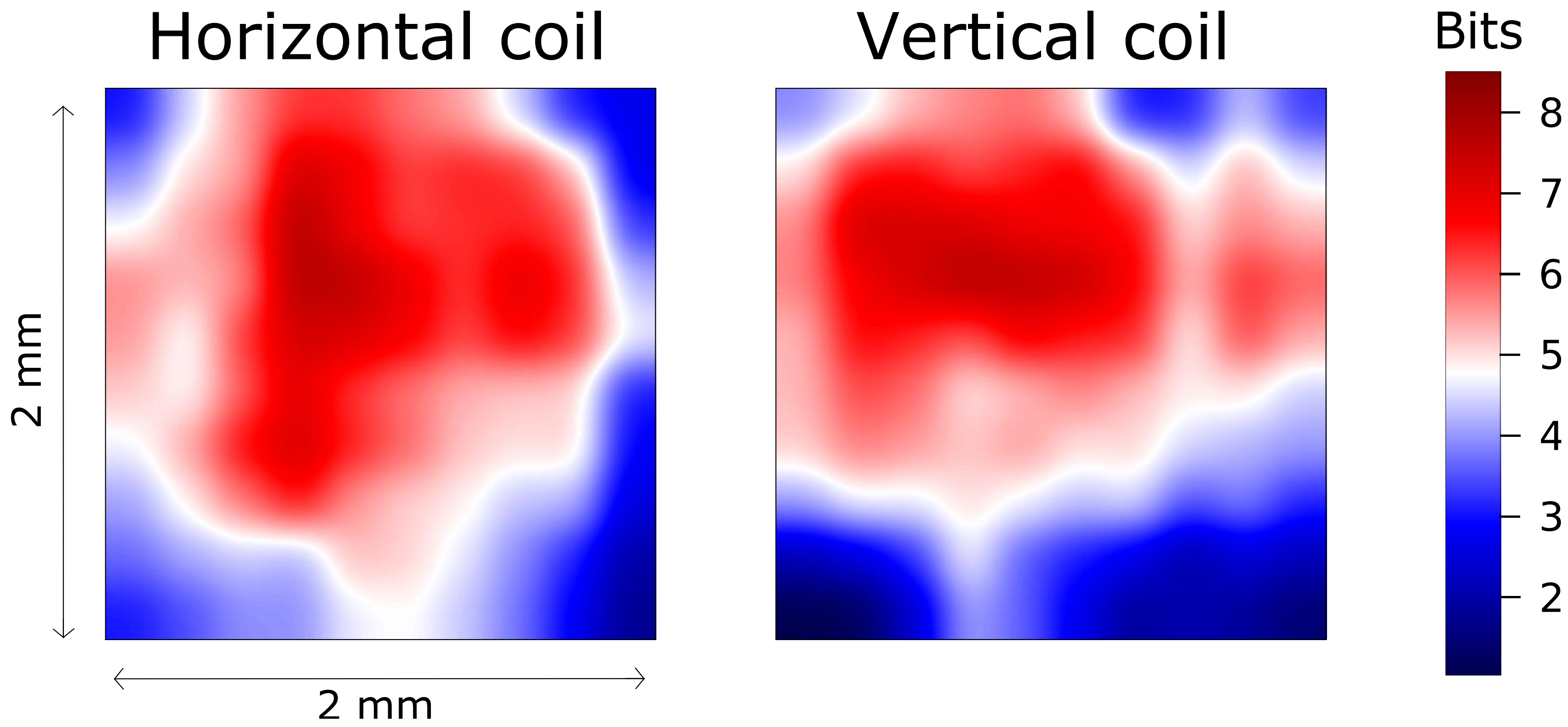
MINE COMPARISON INTEREST

« Is it really worth it to buy the newest scope with the enhanced ADC precision ? »



- MINE can answer in an objective and quantitative way !
- The 15 bits ADC precision provides a slight improvement (around 10%)

INSTRUCTIONS LEAKAGES



➢ MINE can also detect leakages directly from assembly instructions

MINE AS A LEAKAGE ASSESSMENT TOOL

- MINE constitutes a new **leakage assessment** tool that considers full traces as leakage variables
- **All** the potential **leakage sources** are detected
- MINE can **recombined** samples and extract higher order leakages
- MINE is a great **comparison** tool either to compare implementation, countermeasures or hardware setups in order to maximize the MI

Published at ACNS 2020 - Best paper award

A natural problem is now to investigate how to extract this information:

How to use MINE in an attack context ?

Thank you !

ANY QUESTIONS ?