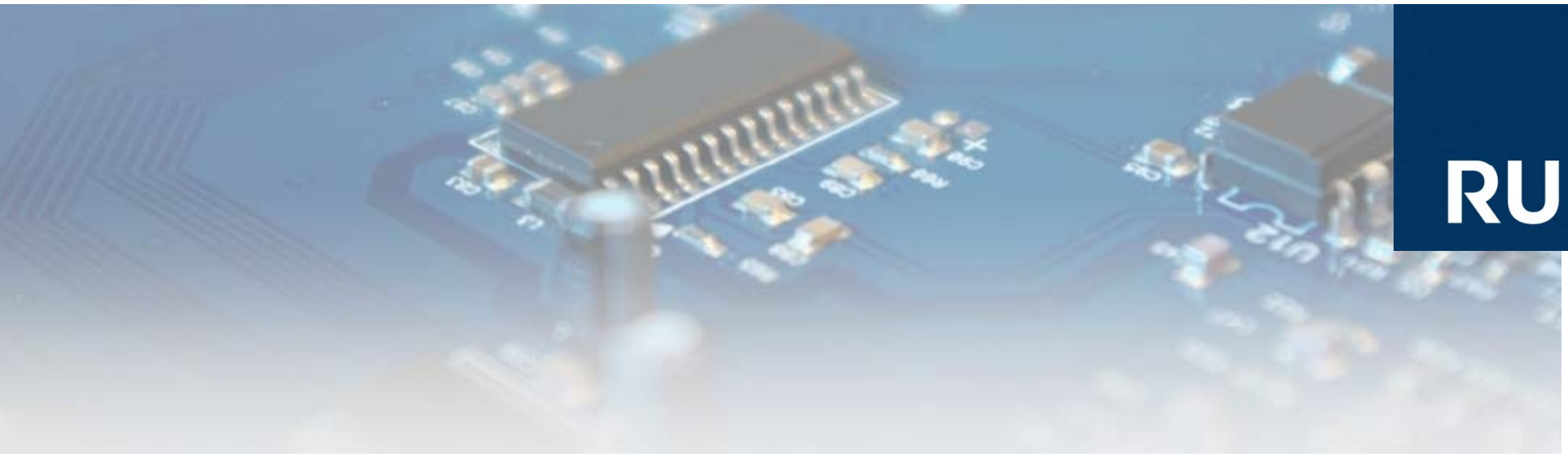


RUB

RUHR-UNIVERSITÄT BOCHUM

Side-channel based Collision Attacks, Theory to Practice

3. December 2010

hgi
Horst-Görtz Institut
für IT Sicherheit

Amir Moradi

Embedded Security Group, Ruhr University Bochum, Germany

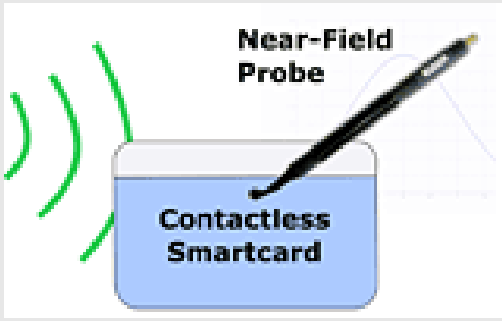
EMSEC
EmbeddedSecurity

Outline

- Classical side-channel attacks
- What is a side-channel based collision attack?
- Implementation platforms and problems
- A newly introduced side-channel based correlation collision attack
- Some hints when implementing

Classical Side-Channel Attacks

- Collecting the side-channel leakage
  - Using an oscilloscope for power analysis attacks
    - and an electromagnetic probe for electromagnetic analysis attacks
  - Using a timer for timing attacks



Classical Side-Channel Attacks

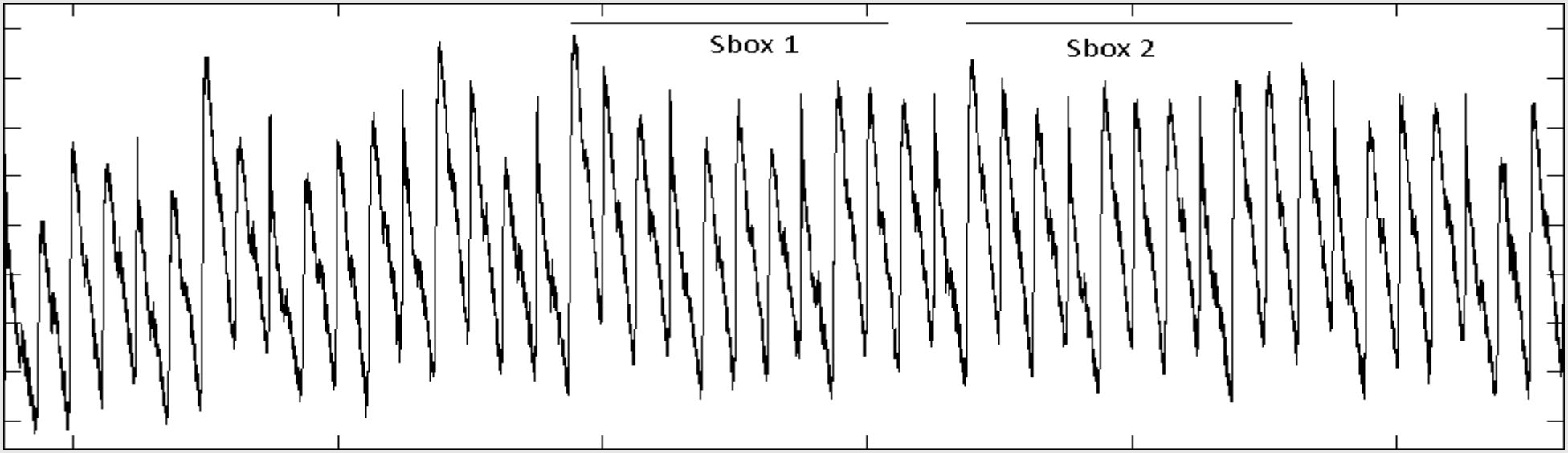
- Define the hypothetical power model
  - In differential power analysis
  - In correlation power analysis
- Define the distinguisher
  - In mutual information analysis
- Examine the relation between the (hypothetical) model and the real measurements using statistical tools
  - difference of means
  - correlation coefficient
  - entropy

What is a Side-Channel Based Collision Attack?

- avoids any model to predict the power consumption
  - Independent of the leakage type
- Examines the similarity of the measurements for different processed values
  - when a collision is found, a relation between parts of the secret is revealed

Side-Channel Based Collision Attack [example 1]

- Implementation platform: a micro-controller
- Target algorithm: the AES encryption
- Strategy of the attack: looking at the similar power consumption traces for different Sbox outputs



- $Sbox(P_1+K_1) = Sbox(P_2+K_2) \Rightarrow P_1+K_1=P_2+K_2 \Rightarrow K_1+K_2 = C$

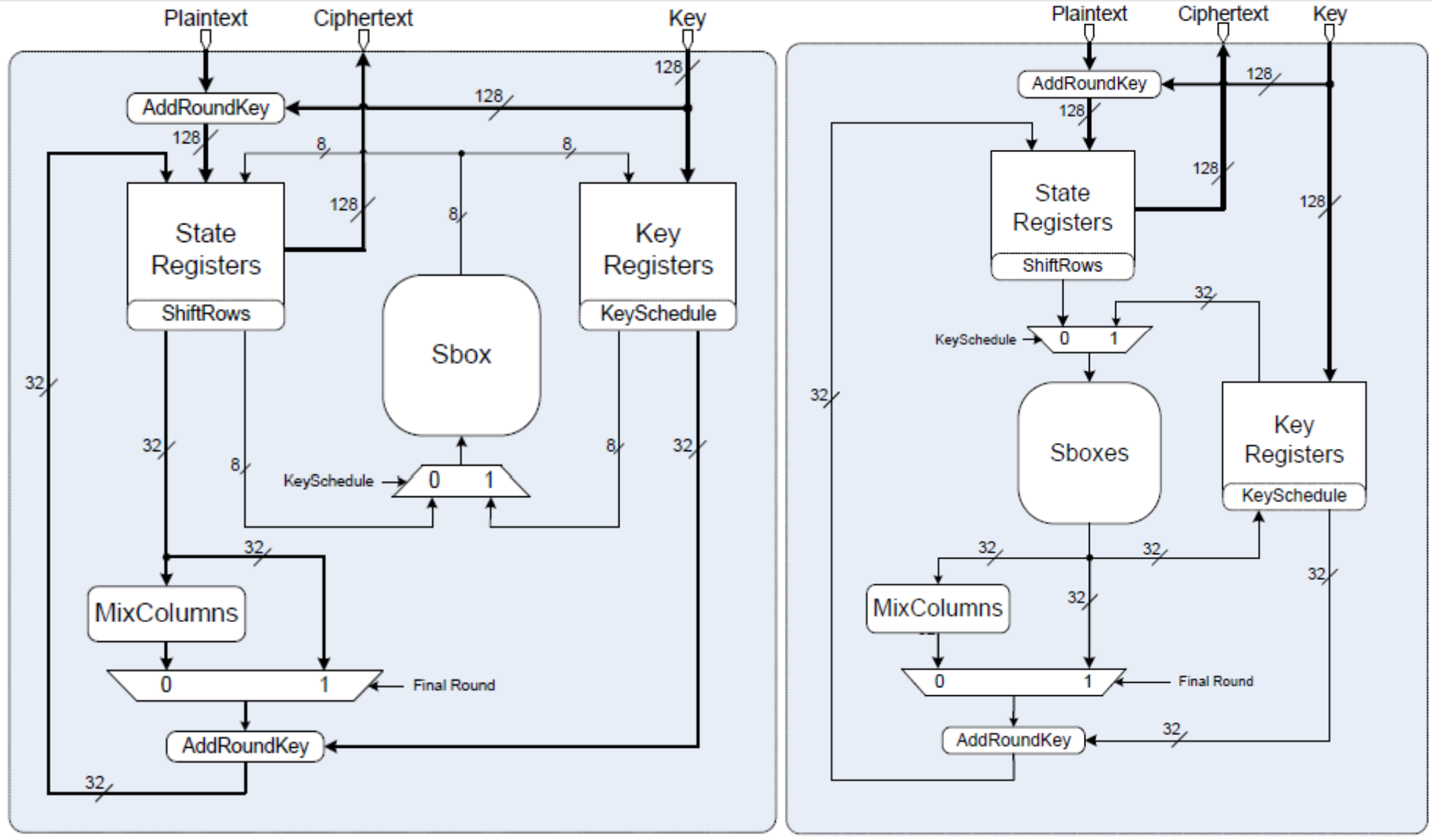
Side-Channel Based Collision Attack [example 1]

- Presence of countermeasures
  - Masking: wait till a collision may occur on both masks and Sbox outputs, depends strongly on the masking order
  - Shuffling: extending the search area to consider all clock cycles, may lead to false positive results
  - Masking and Shuffling: efficiency of the attack is drastically reduced!

Side-Channel Based Collision Attack [example 2]

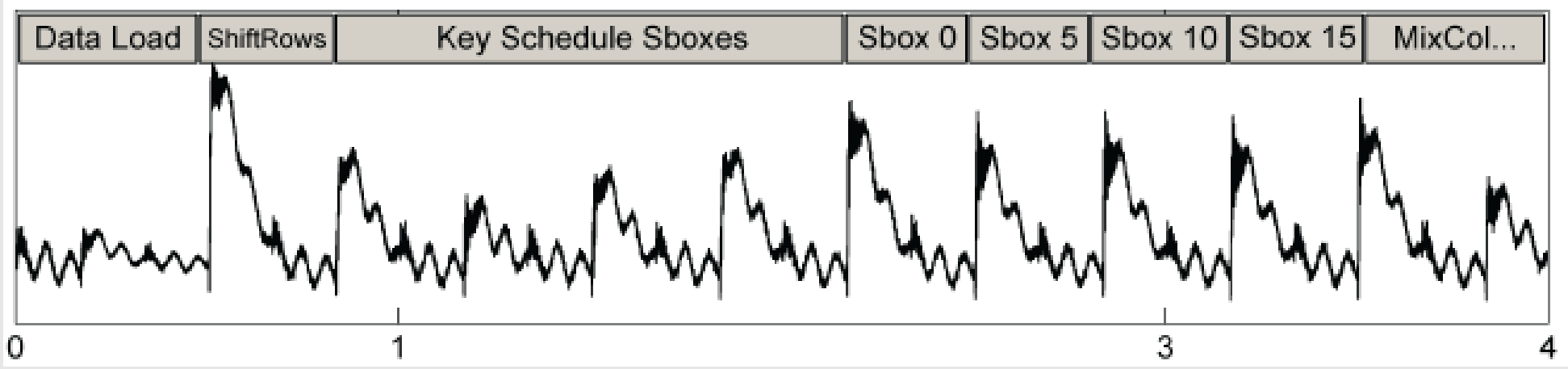
- Implementation platform: an FPGA/ASIC
- Target algorithm: the AES encryption
- Strategy of the attack: cannot be decided without knowing the architecture

An Overview of the Architecture

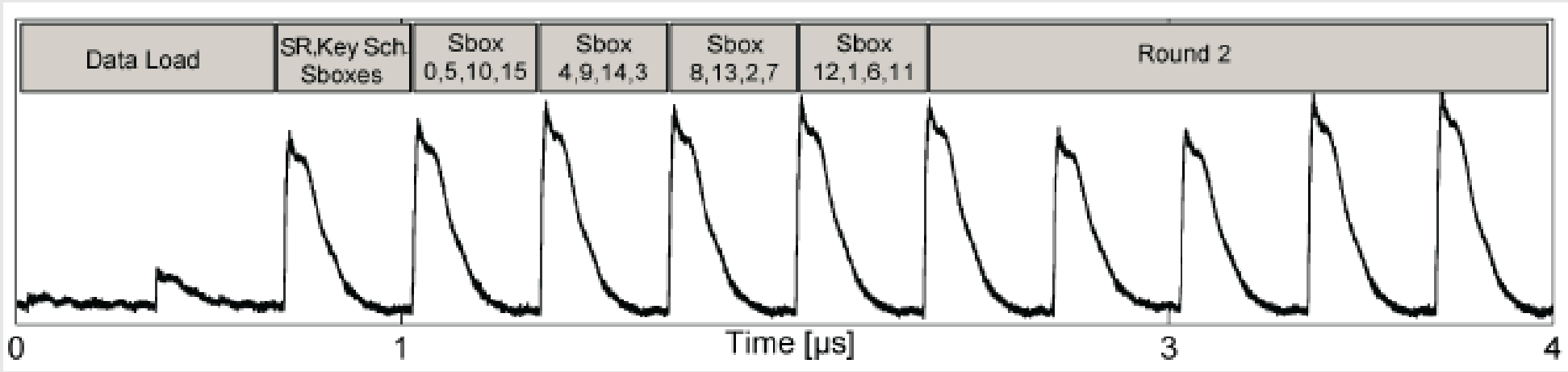


How do the power traces look like?

- 8-bit architecture



- 32-bit architecture



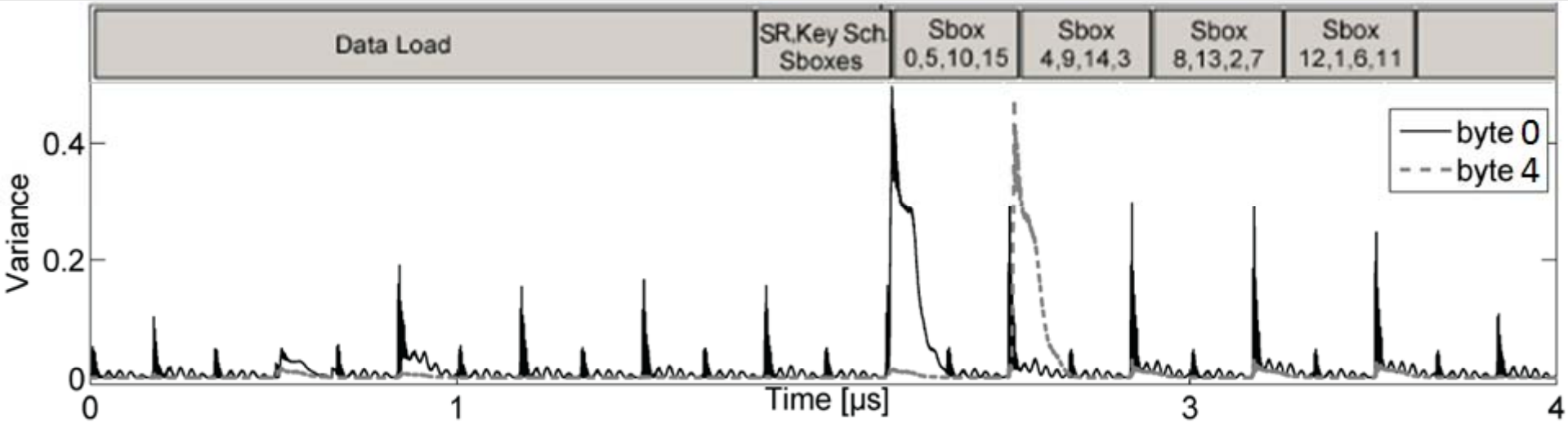
Side-Channel Based Collision Attack [example 2]

- Implementation platform: an FPGA/ASIC
- Target algorithm: the AES encryption
- Strategy of the attack:
  - 8-bit architecture: roughly the same as μC case
  - 32-bit architecture: not easy because of the low probability of collision
- The attack does not work efficiently
  - Switching noise is added in comparison to the μC
  - Power consumption depends also on the last processed values
- Worse situation in the presence of countermeasures

What can we do?

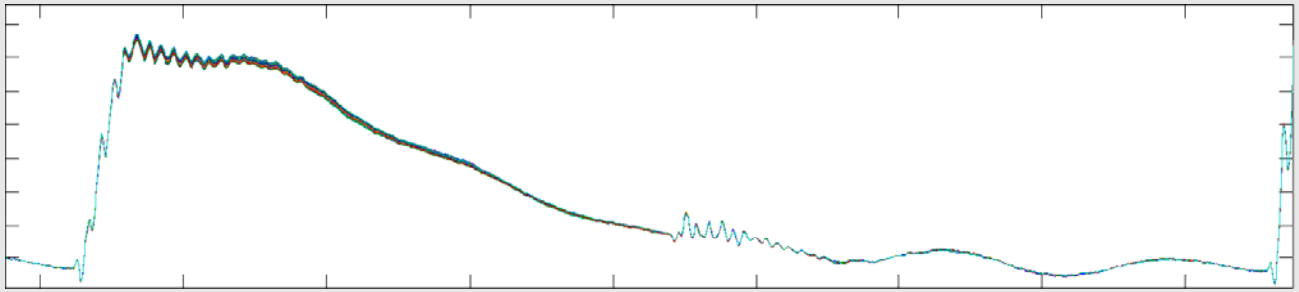
[Usually a DPA/CPA using HD/HW model works + MIA]

- Before developing an attack
  - First, averaging based on plaintext bytes (32-bit arch.)
    - 256 mean traces for each plaintext byte
    - Variance over mean traces (each plaintext byte separately)

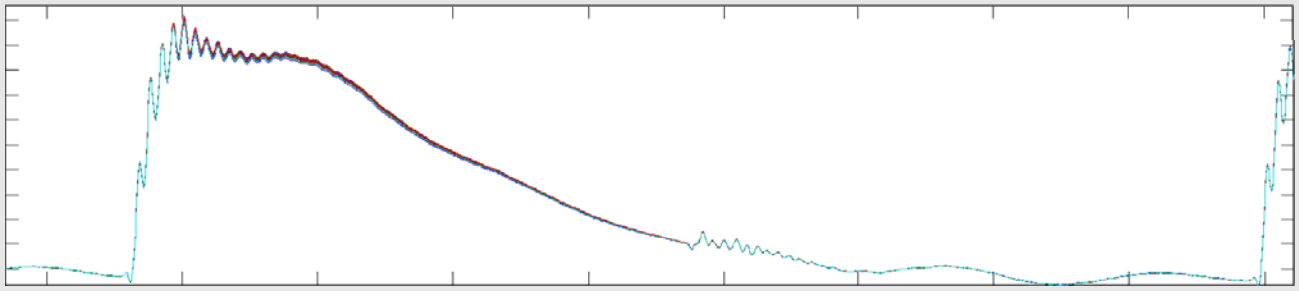


Designing an Attack

- Supposing knowing a key byte, we get mean traces for the corresponding Sbox input byte



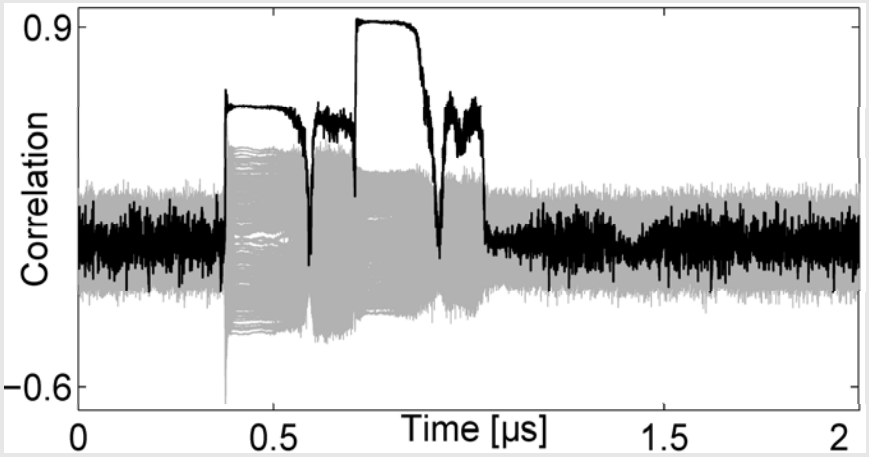
- For another plaintext byte (unknown key), we get mean traces

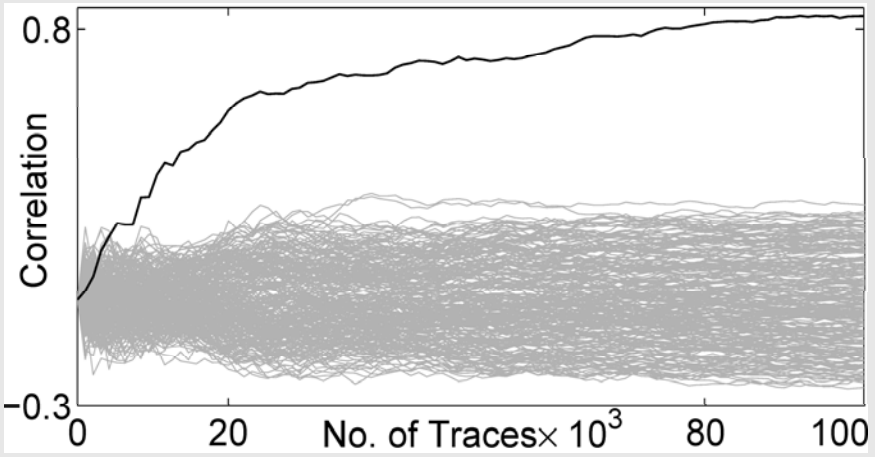


- How are these mean traces related to each other?

Designing an Attack

- The mean traces for the unknown key bytes can be generated for each key byte hypothesis
- The correct key byte can be found comparing the mean traces at each time instance
  - Correlation helps here!
    - Correlation of two sets of mean traces based on key hypothesis (is almost 1 for right key (due to equal power consumption))





Extending the Attack

- If the first key byte (for the first mean traces) is not known, what we recover is the linear difference between two key bytes: k_1+k_2 , because of addroundkey of AES
  - The same attack shown on μC but using all possible collisions!

Why does it work?

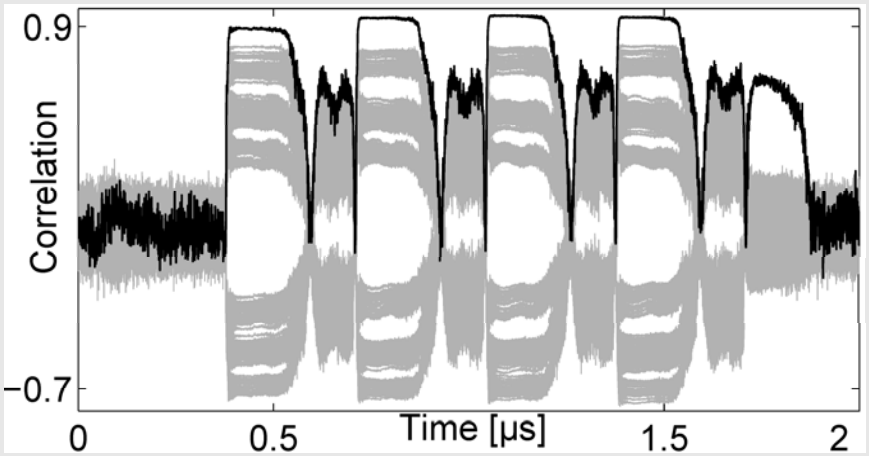
- There are four instances of S-box in the 32-bit arch.
  - The power consumption characteristics of the same instance of the S-box is used in mean traces
  - Power consumption of an instance of the S-box is compared to itself in different clock cycles
- What does happen for larger architecture?
  - The same netlist for the S-boxes, even the same placement and routing, but still process variations exists
    - Small differences on power consumption characteristics of different instances of the S-box
  - The same instances of the S-box should be compared

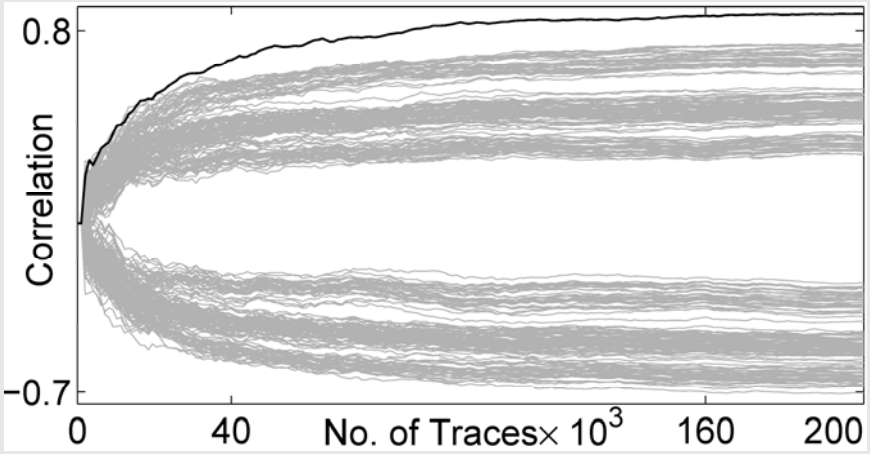
The gain of the attack

- Relation between key bytes
  - 8-bit arch. → 15 relations, 2^8 candidates for the 128-bit key
  - 32-bit arch. → 12 relations, 2^{32} candidates for the 128-bit key
- How to get the correct key?
  - A pair of plain-/ciphertext
  - Continue the attack on the second round of the AES for each key candidate

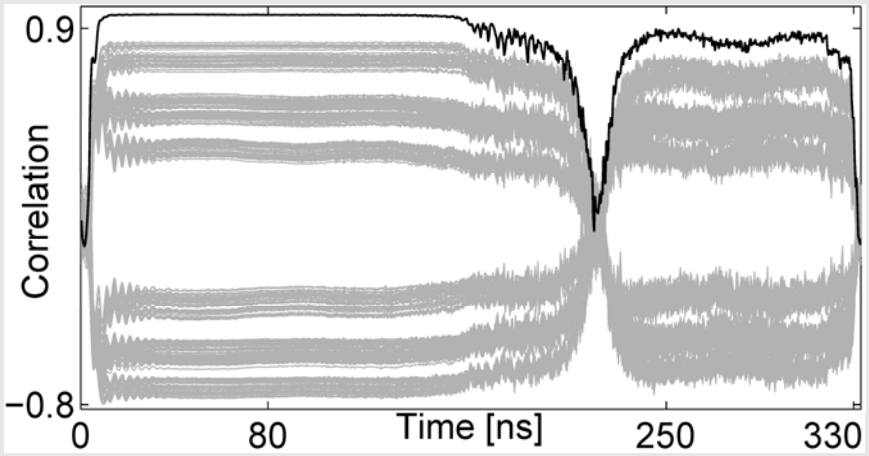
How about Shuffling?

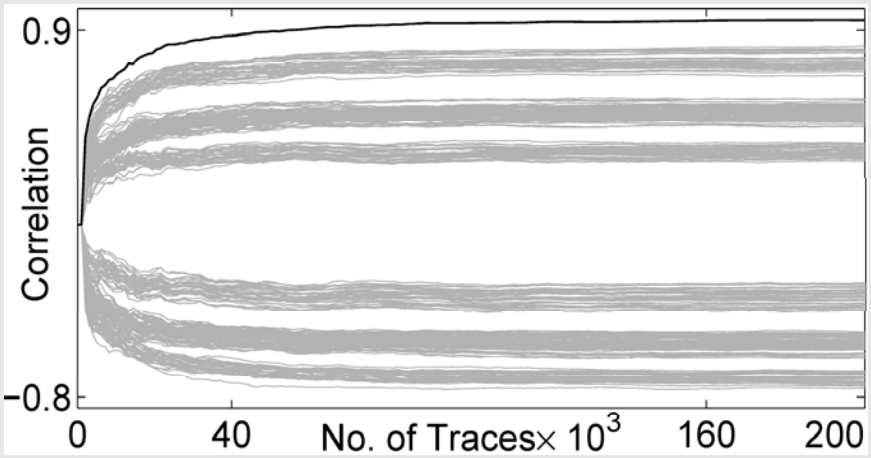
- Shuffling is done on the order of Sbox runs





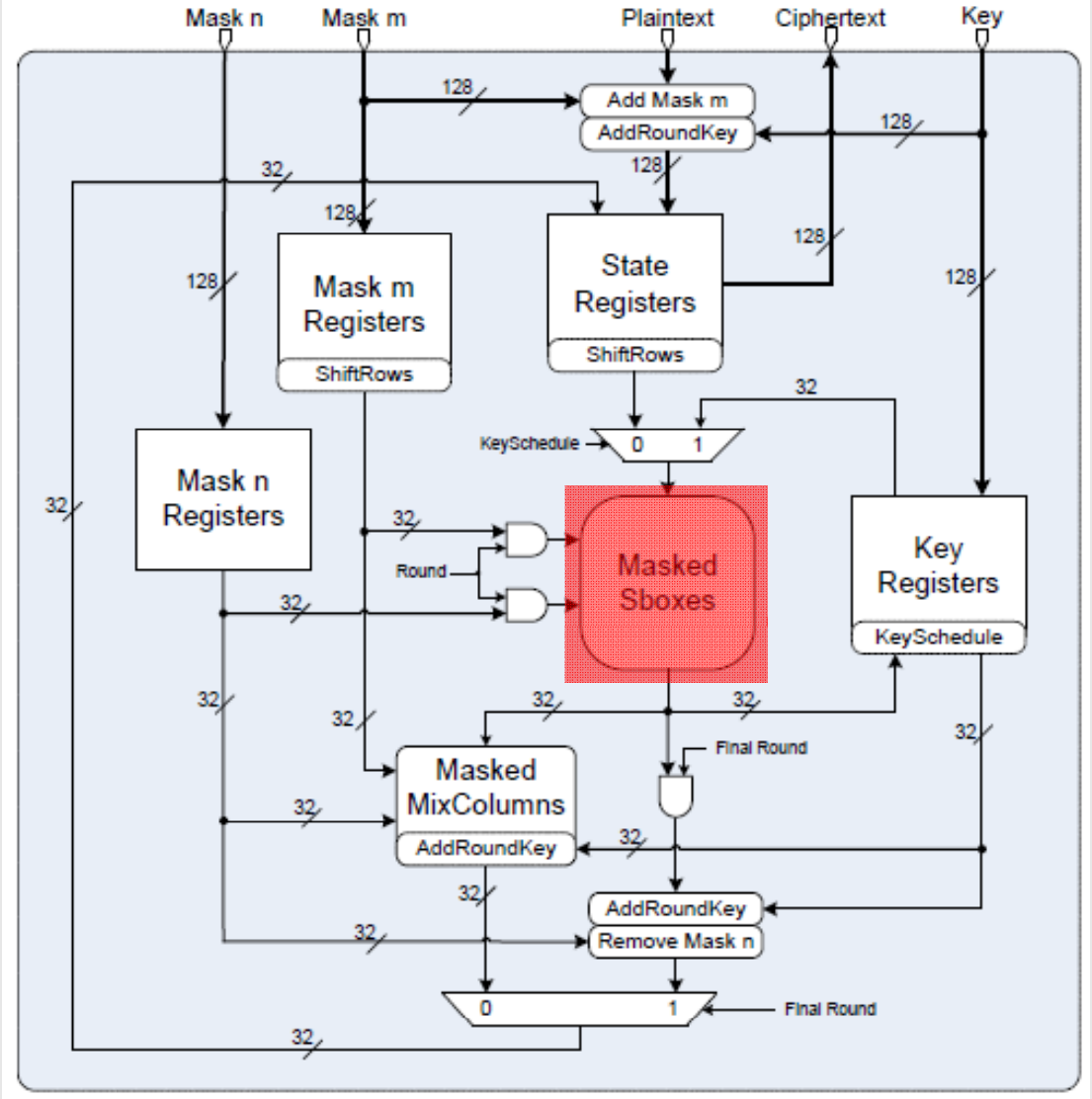
- Using combing [what's combing?]



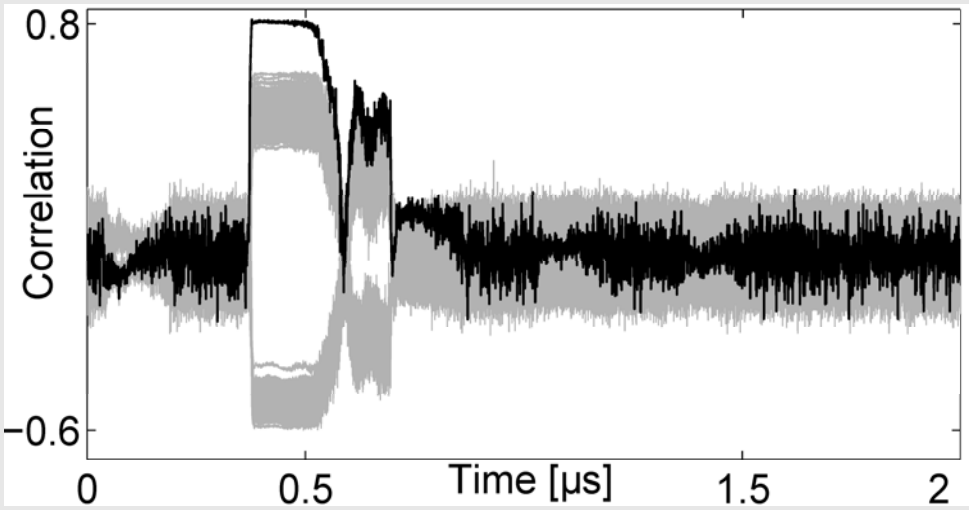


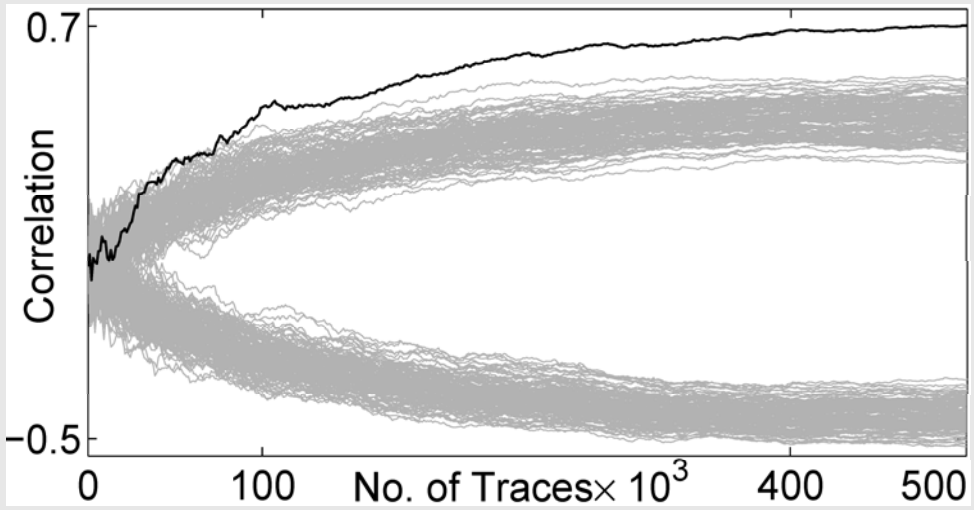
How about Masking?

- Looking into the literatures
- smallest masked AES S-box by Canright and Batina
- 1st order leakage is obvious because of glitches

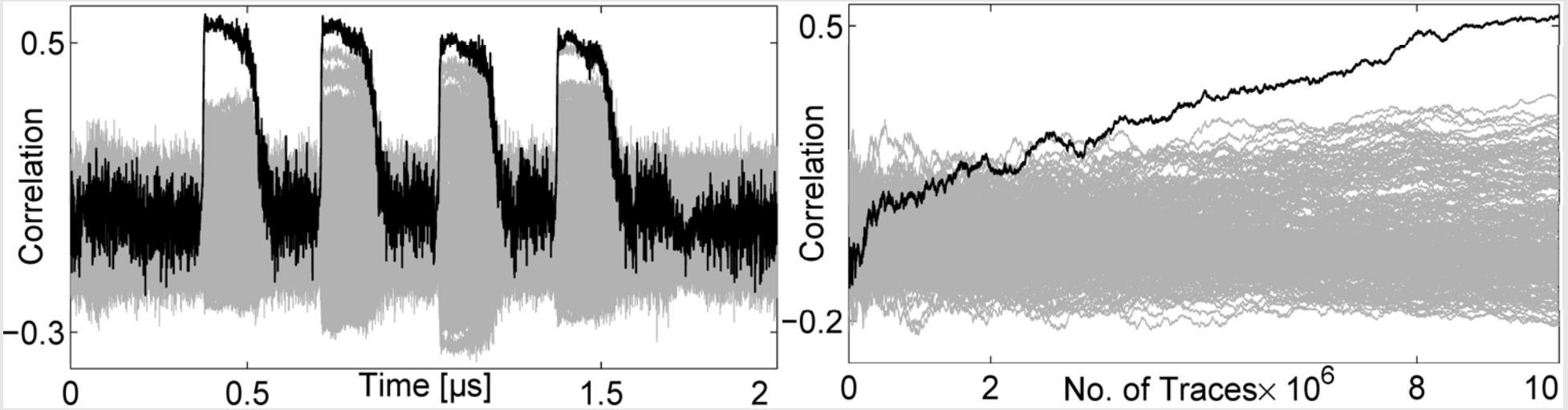


Results when masking is implemented

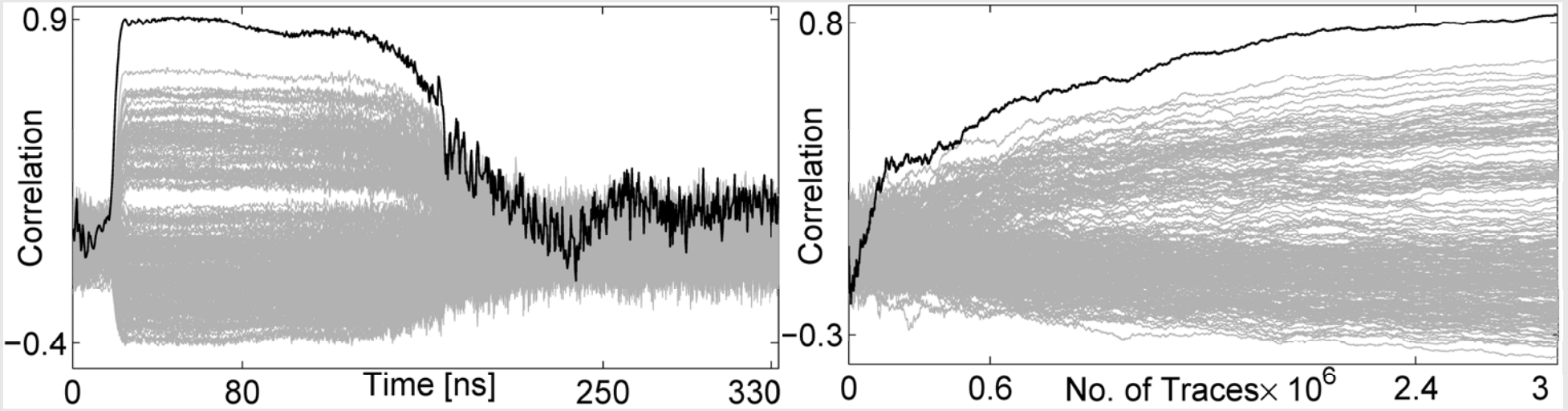




Masking combined with Shuffling?



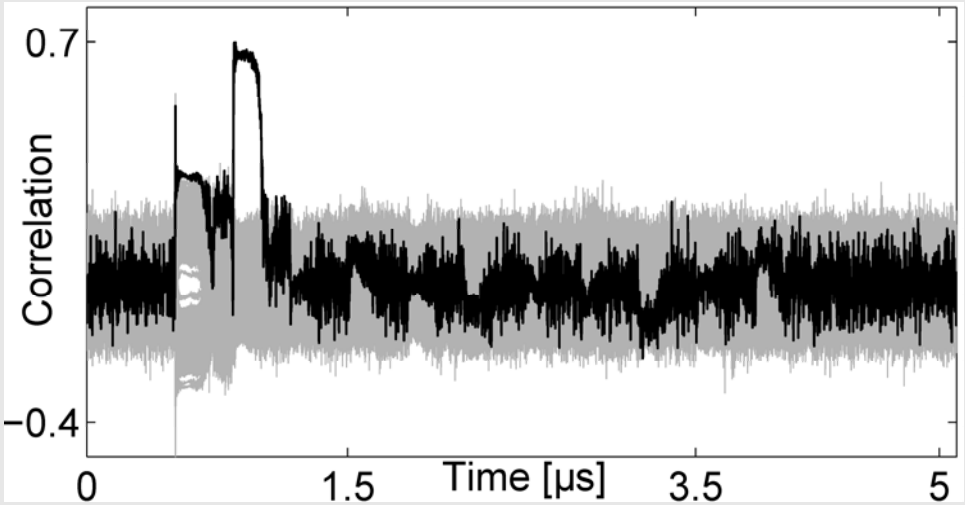
- Using combing

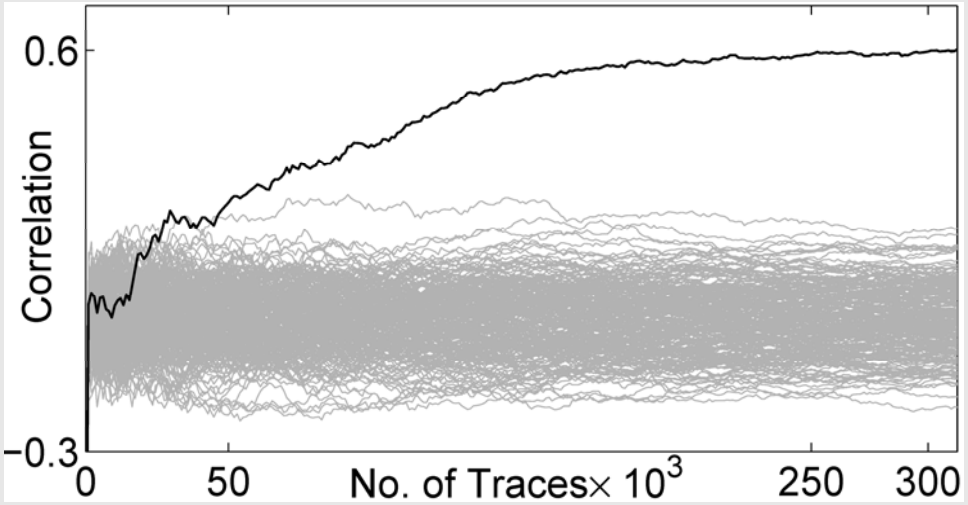


First Hints

- The attack works when an instance of the Sbox is shared for a computation of a round
- Try to avoid Sbox [hardware] sharing
  - going through round-based implementation
    - 128-bit architectures
    - even unrolled architectures

Results of on 128-bit arch. [unmasked]

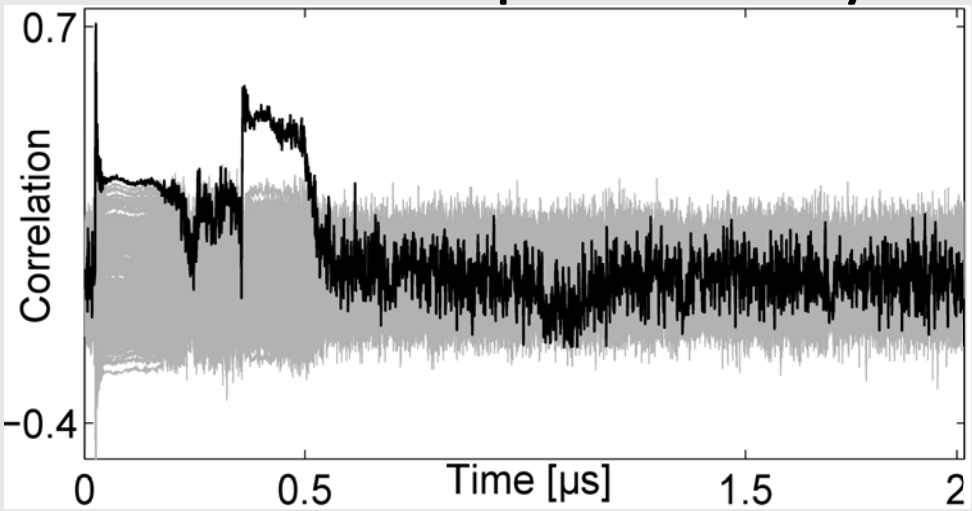


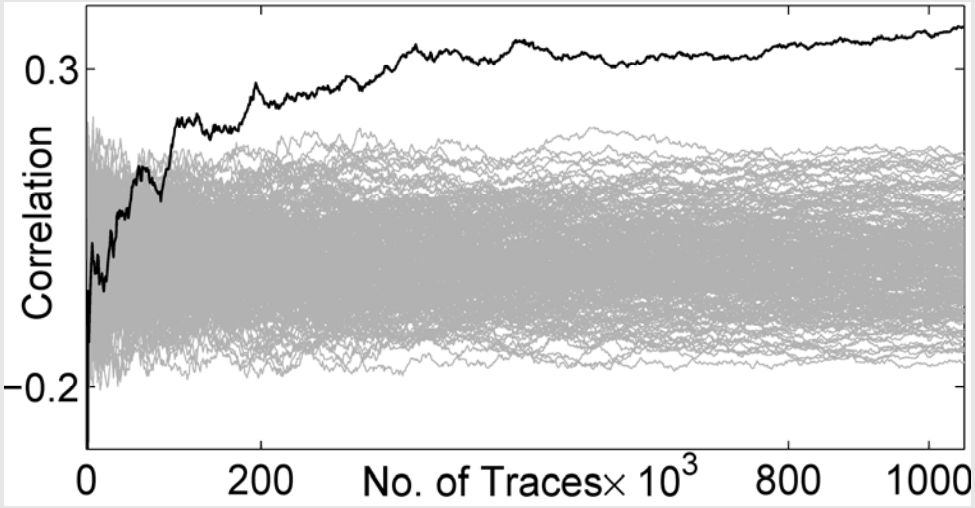


- not achieved for all key bytes
  - because of difference between netlist of different instances of Sbox

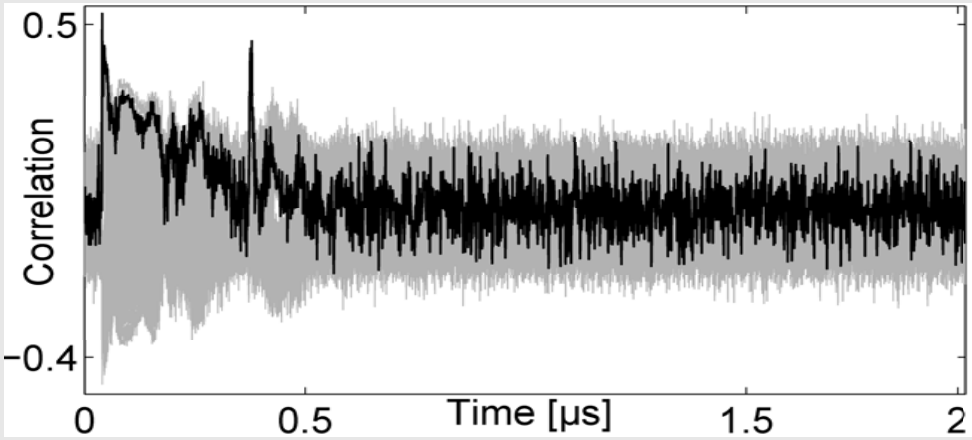
How about unrolled implementations?

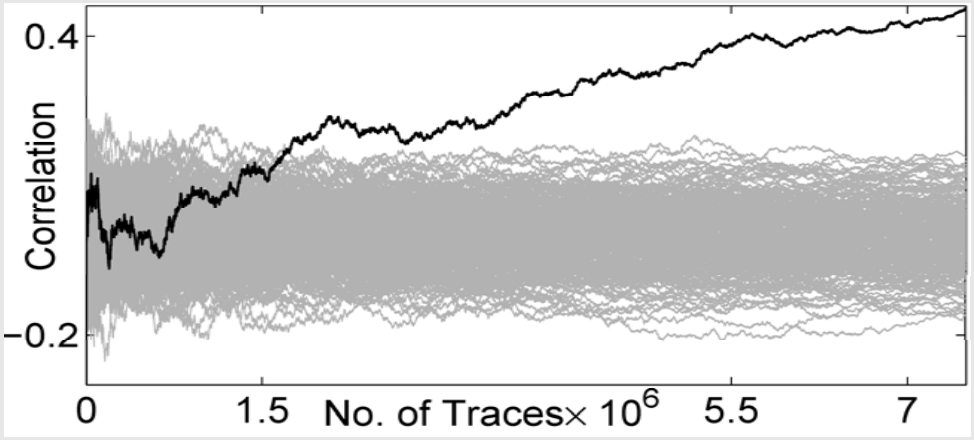
- two rounds per clock cycle





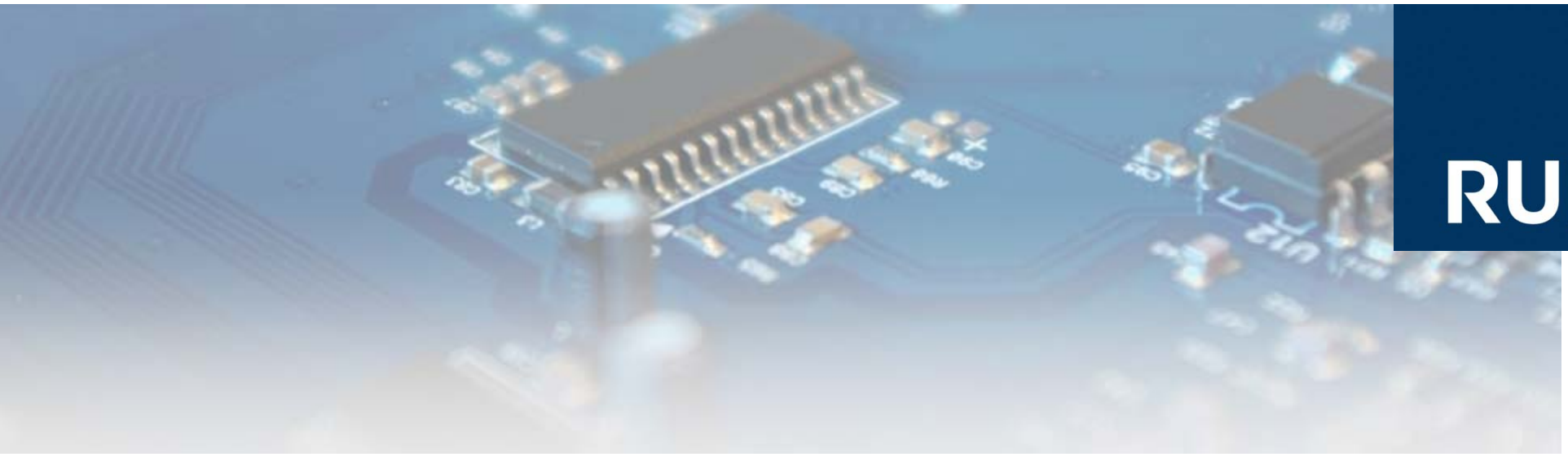
- three rounds per clock cycle





Second Hints

- The attack still works on some key bytes even on unrolled implementations
- To avoid such an attack it is recommended to used different netlists for different instances of the Sbox
  - the result will avoid similarity of the power consumption of different instances of the Sbox
- The world still is not enough
  - at the end of the day, a statistical tool, e.g., MIA, will recover the secret!



Thanks!

Any questions?

Thanks to my colleagues:

Oliver Mischke
Thomas Eisenbarth

Embedded Security Group, Ruhr University Bochum, Germany