RUHR-UNIVERSITÄT BOCHUM

hgi SHA

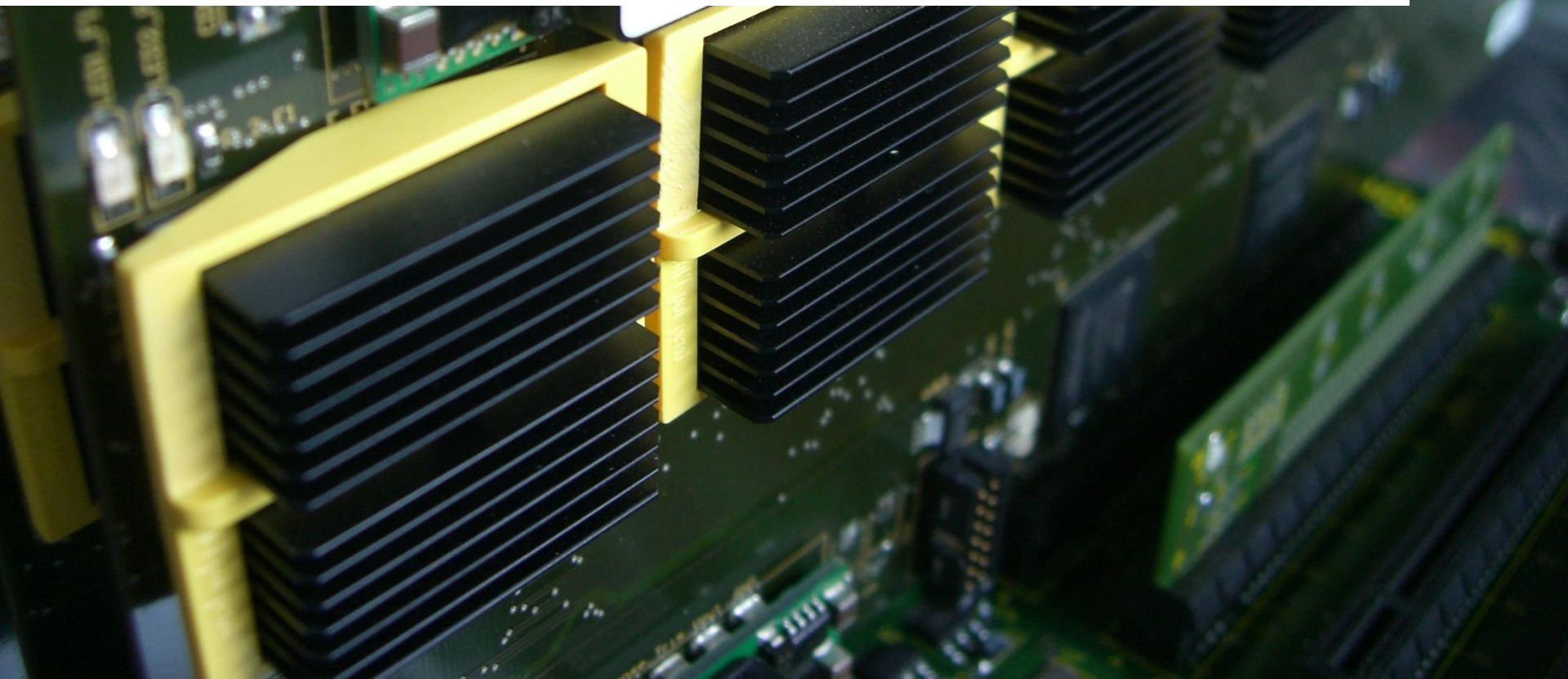
RUB

SIDE-CHANNEL PROTECTION WITH DYNAMIC LOGIC RECONFIGURATION AND RANDOMIZED LOOK-UP TABLES ON FPGAS

PASCAL SASDRICH, AMIR MORADI, OLIVER MISCHKE, TIM GÜNEYSU

CRYPTO-DAY 2015, INFINEON, MUNICH

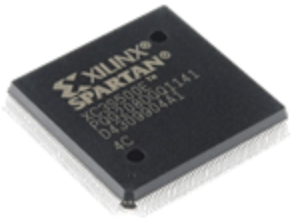
JULY 10, 2015



OUTLINE OF THIS TALK

- **INTRODUCTION**
  - MOTIVATION
- **CASE STUDY 1: PRESENT [HOST15]**
  - IDEA AND CONCEPT
  - DESIGN AND COUNTERMEASURES
- **CASE STUDY 2: AES [COSADE15]**
  - IDEA AND CONCEPT
  - DESIGN AND COUNTERMEASURES
- **RESULTS**
  - LEAKAGE ASSESSMENT METHODOLOGY
  - PRESENT: SPECIFIC t-TEST
  - AES: NON-SPECIFIC t-TEST
- **CONCLUSION**

WHAT IS THE IDEA BEHIND THIS WORK?



- **FPGA**: *(re-)programmable logic device* popular for cryptographic implementations
- **Partial (runtime) reconfiguration**: exchange (partial) designs on demand
- **Observer:** hard to predict current operation and functionality

Idea: Use partial runtime reconfiguration for protection against an external observer or SCA-attacker.

Problem: Exchanging designs and circuits is very slow and can even can take up to *milliseconds*.

Solution: dynamic logic reconfiguration

- since Virtex-5 family Xilinx FPGAs offer Look-Up Tables (LUT6) with *shift register or distributed memory* option
- they are located in certain slices called **SLICEM**
- exchange logic configuration of LUTs but keep routing structure
- only few *clock cycles* rather than *milliseconds*

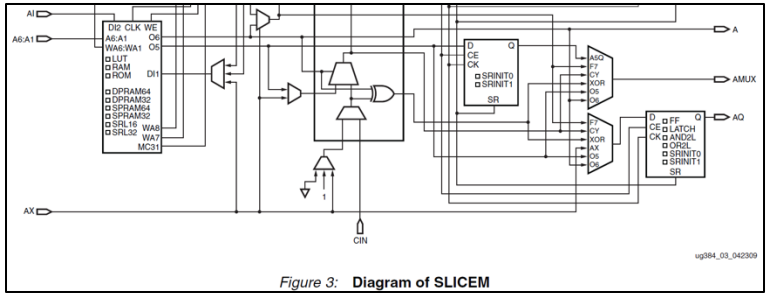
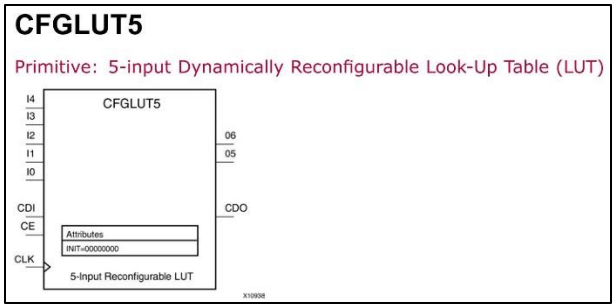


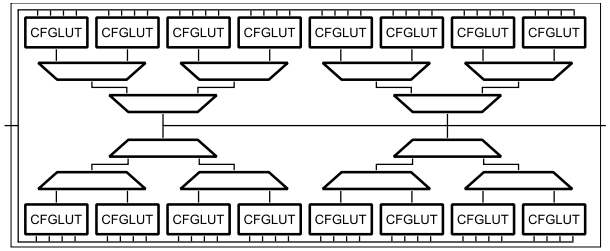
Figure 3: **Diagram of SLICEM**

Question: How can we use these LUTs to build side-channel countermeasures?

CASE STUDY 1: PROTECTING PRESENT [HOST15]

- **Configurable Look-Up Tables** were introduced with *Xilinx Virtex-5* and *Spartan-6* device families
- located in **SLICEM** and based on *shift registers*
- older devices can simply use SRL16E (shift register) instances
- **CFGLUT5** can be used as:
  - single 5×1 LUT (32 cycles for reconfiguration)
  - two 4×1 LUTs with shared inputs (16 cycles for reconfiguration)
- combining multiple CFGLUTs with multiplexers stages we can build $(n \times m)$ ***reconfigurable function tables (RFT)***

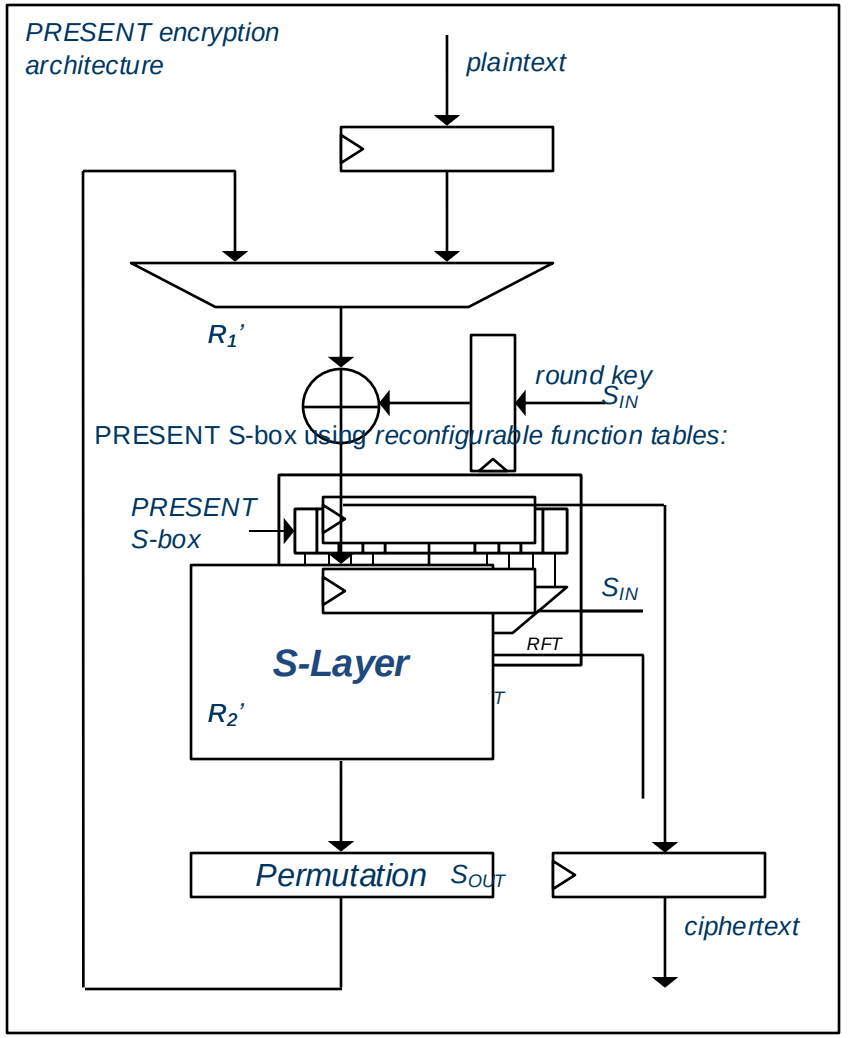




Limitation: For large structures this is inefficient, but for small (4×4) functions like the PRESENT S-box this is an optimal choice.

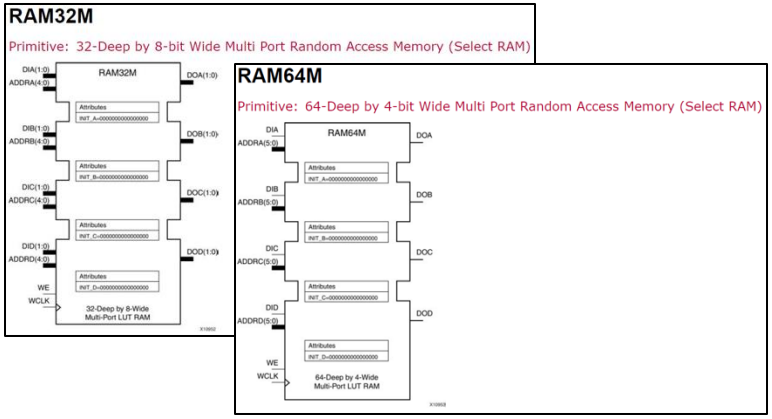
DESIGN AND COUNTERMEASURES FOR PRESENT?

- **round-based architecture** with 16 S-boxes
- all countermeasures target **S-layer**
- implement S-boxes using *reconfigurable function tables*
- **decompose** the PRESENT S-box into two *reconfigurable function tables*
  - first *reconfigurable function table* R_1 is chosen randomly
  - second *reconfigurable function table* R_2 is computed using the original S-box such that: $R_2(R_1(x)) = S(x)$
  - place register stage in between R_1 and R_2 to only store (random) $R_1(x)$
- add **Boolean masking** to both *reconfigurable function tables* and recompute them as:
  $R_1'(x) = R_1(x \oplus m_1) \oplus m_2$
  $R_2'(x) = R_2(x \oplus m_2) \oplus P^{-1}(m_1)$
- insert a second register stage for random **register precharge** to avoid leakage based on the Hamming distance model:
  $HD(x \oplus m, y \oplus m) = HW(x \oplus y)$



CASE STUDY 2: PROTECTING AES [COSADE15]

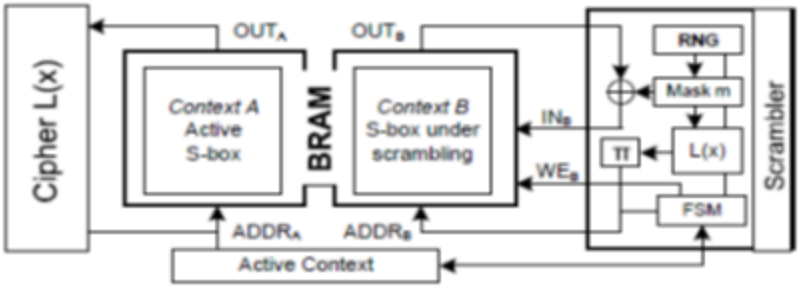
- Use **distributed memory primitives** to build *randomized look-up tables*
- protect S-boxes against first order side-channel attacks (*Boolean masking*)
- efficient for larger structures, since RAM primitives do not loose an input pin (but require address handling instead of shifting data)



Idea: Build Block Memory Content Scrambling [CHES11] approach with *distributed memory primitives*.

Recall the Concept of BMS:

- store 2 S-/T-Tables in one BRAM
- first table is active context and used for encryption
- second table is passive context and updated (scrambled) with fresh randomness
- after update, contexts are switched



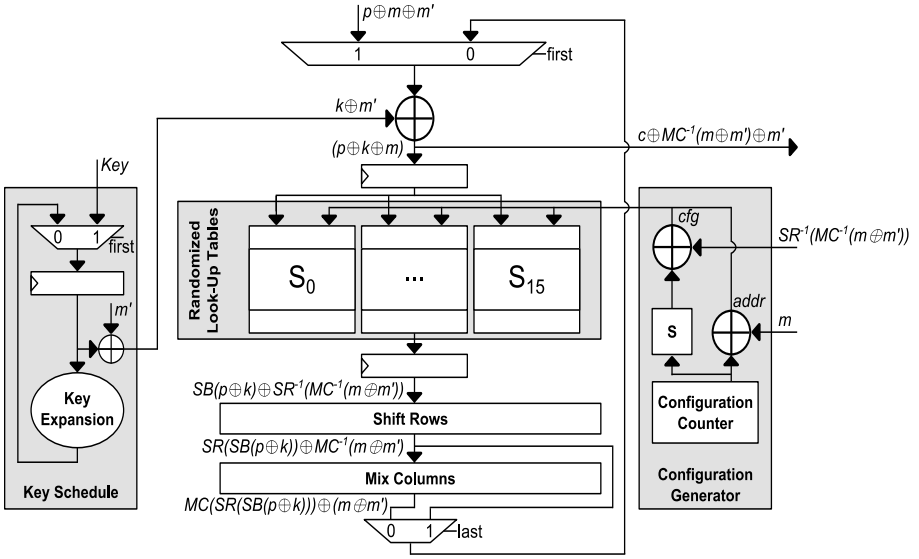
T. Güneysu and A. Moradi. Generic Side-Channel Countermeasures for Reconfigurable Devices.

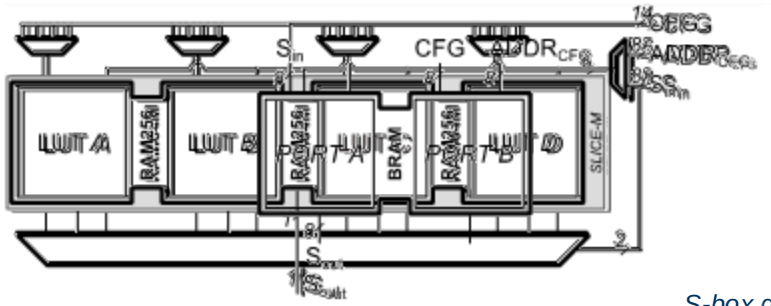
Disadvantages: Area overhead, lower latency, and mask reusing.

DESIGN AND COUNTERMEASURES FOR AES?

- **round-based architecture**
- implement **randomized S-boxes** *(Boolean masking)* using *distributed memory*
- second register stage for random **register precharge** to avoid leakage based on the Hamming distance model:
  $HD(x \oplus m, y \oplus m) = HW(x \oplus y)$
- build S-boxes *BMS-like* with different memory primitives to find optimal choice:
  - RAM32M: *fastest reconfiguration, but highest area overhead*
  - RAM64M: *moderate reconfiguration time with moderate area overhead*
  - RAM256X1S: *slowest reconfiguration but smallest area overhead*
  - RAMB8BWER: *BRAM memory primitive for comparison*
- reconfiguration of S-box in a *prior-to-encryption* fashion (avoids second table)

AES encryption architecture

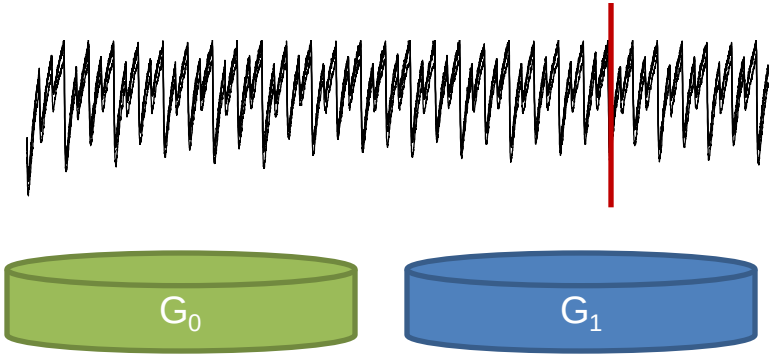




S-box design and memory primitive

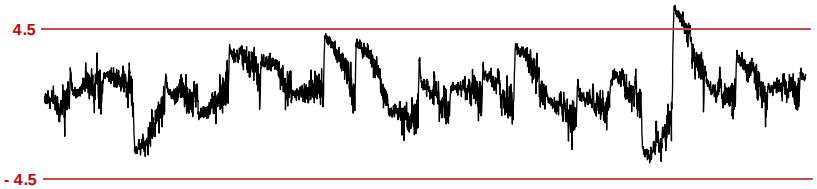
EVALUATION USING WELCH'S *t*-TEST

- measure power traces with digital oscilloscope
- determine distinguisher, e.g.:
  - fix vs. random plaintext (*non-specific t*-test)
  - bit of intermediate round result
  - multi-bit intermediate result
- group traces depending on distinguisher
- compute *sample mean* for each point in time
- compute *sample variance* for each point in time
- determine *t*-statistic for each point in time:



$$t = \frac{\mu(T \in G_1) - \mu(T \in G_0)}{\sqrt{\frac{\delta^2(T \in G_1)}{|G_1|} + \frac{\delta^2(T \in G_0)}{|G_0|}}}$$

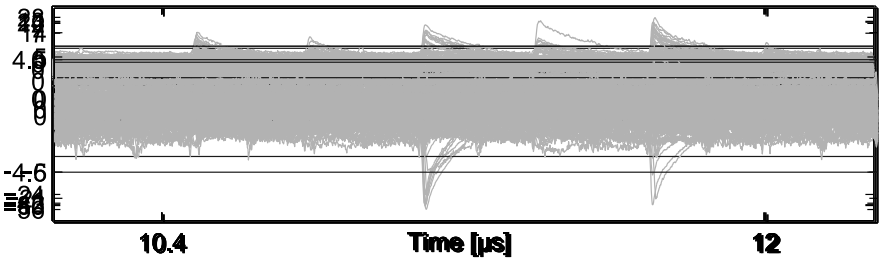
where μ denotes the *sample mean* and δ denotes the *sample variance*.



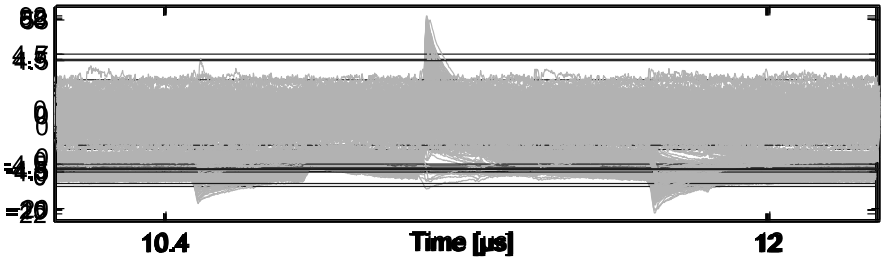
Fail/Pass Criteria: If there is any point in time for which the t-statistic exceeds a threshold of $\pm$ 4.5 the device under test fails.

WHAT ARE THE RESULTS FOR PRESENT?

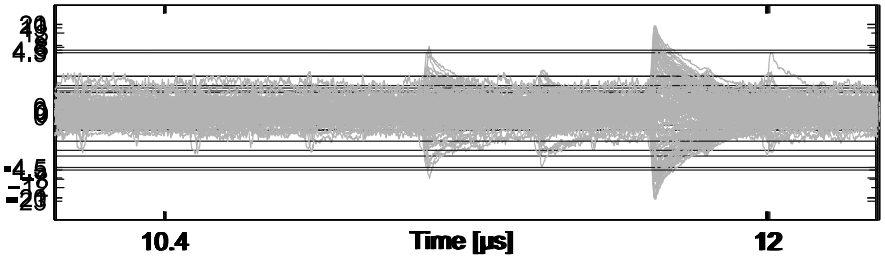
- **distinguisher**: intermediate values of round 16 (bits / nibbles)
- 3 different groups of test:
  - S-box output bits (64 models)
  - XOR of round in and out (64 models)
  - output value of S-box S_0 (16 models)
- 8 different test cases:
  - all countermeasures disabled
  - S-box decomposition
  - Boolean masking
  - register precharge
  - S-box decomposition and register precharge
  - Boolean masking and register precharge
  - S-box decomposition and masking
  - S-box decomposition, masking and register precharge
- 1 million power traces except for last test case: measured 10 million



Group 1: S-box output bits (64 models)



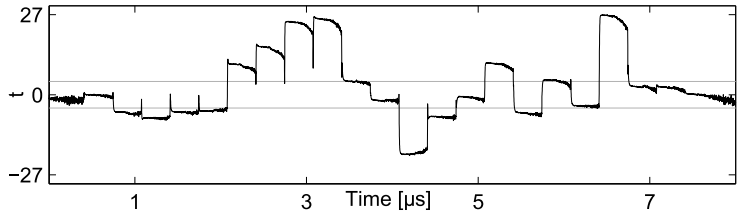
Group 2: XOR of round in and round out (64 models)



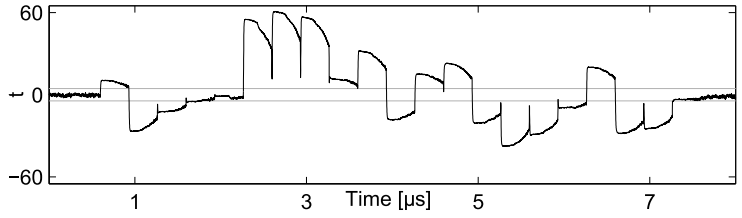
Group 3: Output value of S-box S_0 (16 models)

WHAT ARE THE RESULTS FOR AES?

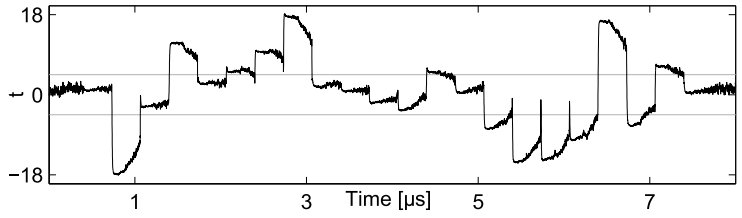
- **distinguisher**: random plaintext vs. fix plaintext
- 4 different test cases:
  - RAM32M primitive
  - RAM64M primitive
  - RAM256X1S primitive
  - RAMB8BWER primitive
- 1 million power traces except for last test case: measured 10 million
- leakage is detectable for all distributed memory primitives
- we assume that leakage is due to internal slice architecture
- BRAM primitive exhibits no detectable leakage



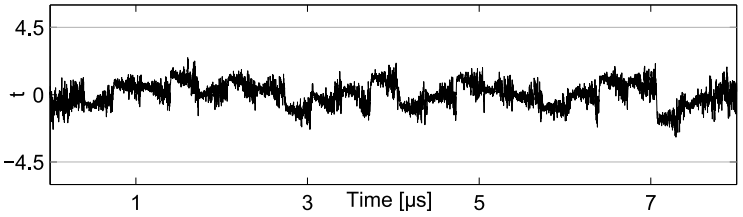
RAM32M (1 million traces)



RAM64M (1 million traces)



RAM256X1S (1 million traces)



RAMB8BWER (10 million traces)

WHAT IS THE CONCLUSION?

- first application of **dynamic logic reconfiguration and randomized look-up tables** based on distributed memory to realize a first-order-resistant masking scheme
- we provide **practical examination** of all designs and countermeasures
- used state-of-the-art **leakage assessment methodology** (specific and non-specific *t*-test)
- designs are first-order resistant even after measuring **10 million** power traces

CAN BE AN EFFECTIVE TECHNIQUE TO ACHIEVE FIRST-ORDER SCA RESISTANCE ON FPGA-BASED PLATFORMS!

BUT OUR RESULTS ALSO INFER THE PITFALL OF USING DISTRIBUTED MEMORY PRIMITIVES!

RUHR-UNIVERSITÄT BOCHUM

hgi SHA

RUB

ACHIEVING SIDE-CHANNEL PROTECTION WITH DYNAMIC LOGIC RECONFIGURATION ON MODERN FPGAS

pascal.sasdrich@rub.de

CRYPTO-DAY 2015, INFINEON, MUNICH

JULY 10, 2015

Thank you for your attention!
Any Questions?