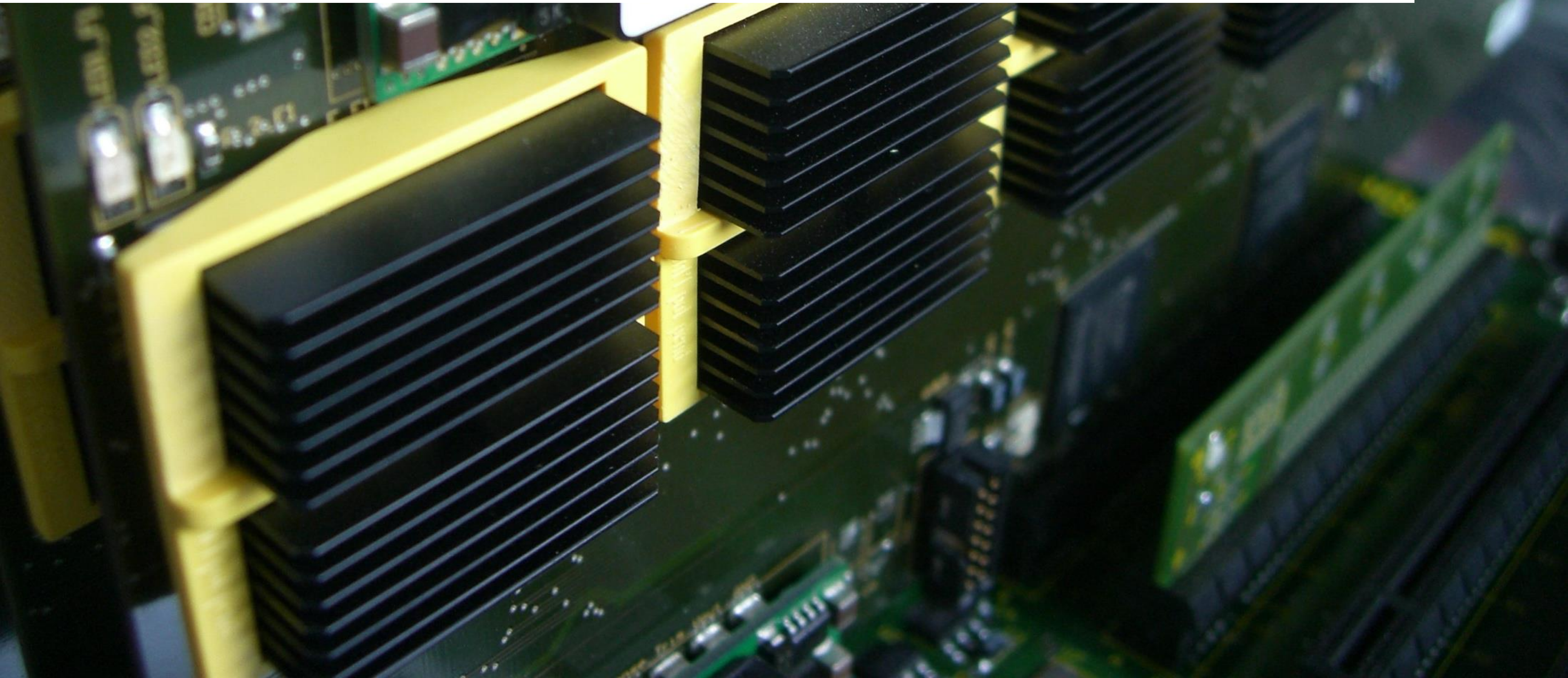
RUHR-UNIVERSITÄT BOCHUM

A GRAIN IN THE SILICON: SCA-PROTECTED AES IN LESS THAN 30 SLICES

PASCAL SASDRICH, TIM GÜNEYSU

27TH IEEE CONFERENCE ON APPLICATION-SPECIFIC SYSTEMS, ARCHITECTURES AND PROCESSORS, LONDON, ENGLAND

JULY 6, 2016



WHAT IS THE IDEA OF THIS WORK?

Situation:	Standardized symmetric encryption is required by multiple components within a complex System-on-a-Chip (SoC) or on FPGAs.
Problem:	Components are provided by different vendors with not always mutually trusted relationships.
Current Solution:	Every security-critical SoC component has a separate security subsystem with a separate AES core.
Future Solution:	Provide a shared AES core that provides all necessary functions and security features for all Soc components.

Questions:

- Can we build an AES-128 encryption for Xilinx FPGAs that is smaller than existing solutions, i.e. the PicoBlaze soft core micro-processor (26 Slices & 1 BRAM)?
- Can we build an encryption core that is self-contained (key expansion, PRNG,...)?
- Can lightweight implementations still provide protection against side-channel attacks?

Our Contribution:

- We provide the currently smallest AES-128 implementation optimized for Xilinx FPGAs, including on-the-fly key expansion in only **21 slices** without any BRAM.
- Random number generation and side-channel protection extends this to only **28 slices.**
- Confirmation of basic protection through **practical side-channel measurements** and evaluation.

XILINX FIELD-PROGRAMMABLE GATE ARRAYS (FPGA)

Let's take a short glance at Xilinx FPGAs:

- organized in special cells: Configurable Logic Blocks (CLB), memory (BRAM), IO,…
- CLBs contain 2 slices with 4 LUTs / 8 FF each (general purpose logic)
- there are three different kind of slices:
  - **Slice-X**: basic logic functionality
  - **Slice-L**: enhanced logic functionality
  - **Slice-M**: additional memory features

In particular the Slice-M are interesting:

- all 4 LUTs could be combined and used as 4 x 64 bit ***Distributed Memory***
- different memory options such as
  - 32 x 8-bit (RAM32M),
  - 64 x 4-bit (RAM64M),
  - and 256 x 1-bit (RAM256X1S)

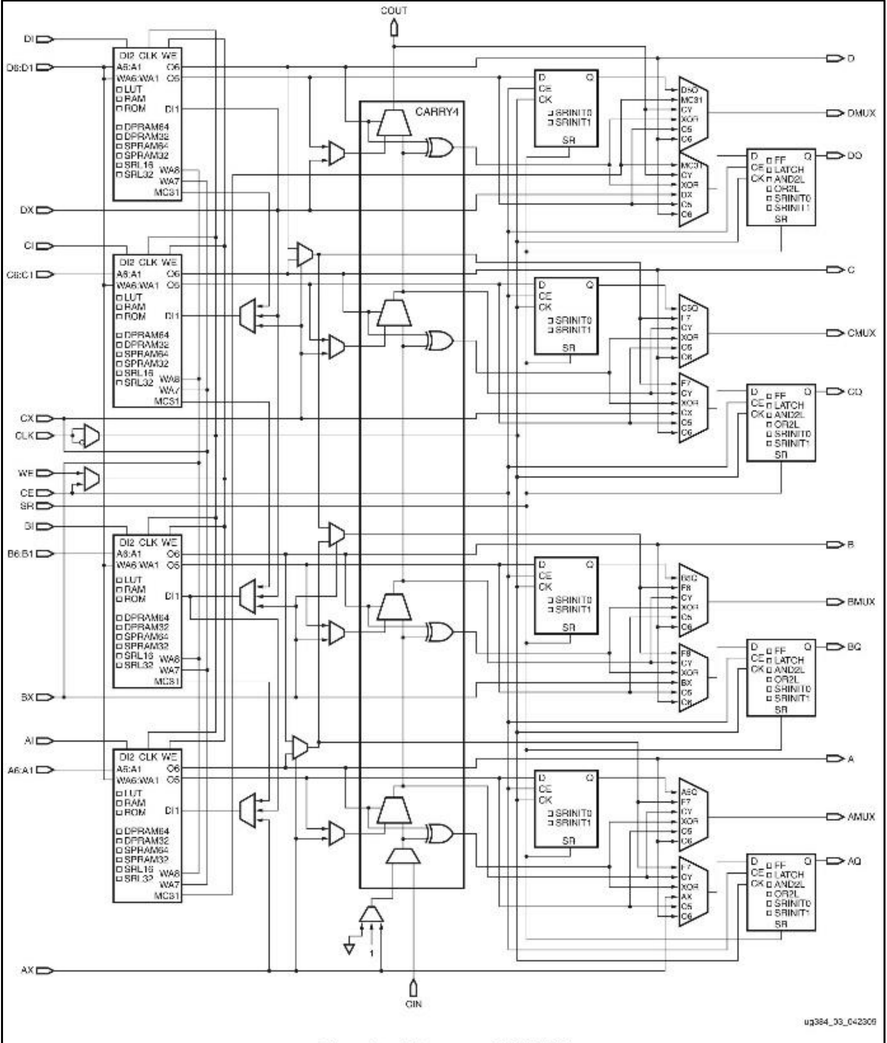


Figure 3: **Diagram of SLICEM**

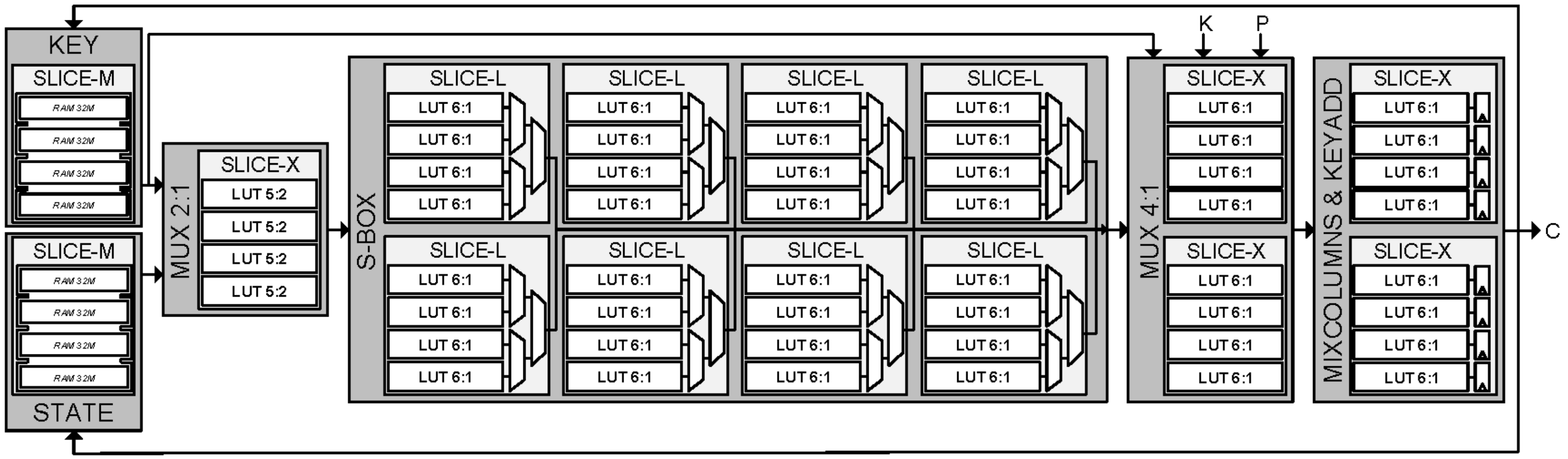
WHAT IS THE MAIN CONTRIBUTION OF THIS WORK?

Problem: Find an optimal mapping of the AES algorithm to implement it as small as possible on current Xilinx FPGA technology still providing physical protection.

Ideas:

- use ***byte-serial implementation*** and store intermediate state into ***Distributed Memory***:
  - AES has 128-bit internal state and most operations are byte-wise
  - Xilinx FPGAs can store up to 32 x 8-bit values (256 bit) into a single slice
- the **AES S-box** is the most costly part in terms of logic resources:
  - share S-box for key schedule and round computation
- **MixColumns** most challenging operation:
  - only operation that works on 4 bytes (column) of the state
  - main operation is XOR (besides constants multiplication in GF(2^8))
  - following AddRoundKey operation can be merged with MixColumns computation
- basic protection by **hiding side-channel leakage** in random noise:
  - random shuffling of execution order
  - supported inherently due to *Distributed Memory*
  - self-contained random number generation (using AES core as PRNG)

THE BASIC ARCHITECTURE (ENCRYPTION ONLY)



- all state and key information is stored in *Distributed Memory*
- data paths for round function and key schedule are merged

Atomic functions of AES-128 are realized as:

- ***SubBytes***: implemented as look-up table (8 slices)
- ***ShiftRows***: implemented by memory address translation (*inherent support for shuffling*)
- ***MixColumns***: ALU with register for intermediate results, providing 4 operations

$$00 : r_i = x_i \ (\text{set}) \qquad 10 : r_i = r_{i-1} \oplus 02 \cdot x_i \ (\text{add two times})$$
$$01 : r_i = r_{i-1} \oplus x_i \ (\text{add}) \qquad 11 : r_i = r_{i-1} \oplus 03 \cdot x_i \ (\text{add three times})$$

- ***AddRoundKey***: merged with *MixColumns* using add operation

ENHANCED AND PROTECTED ARCHITECTURES

Enhancements:

- protection against physical **side-channel attacks**, such as *Differential Power Analysis* (DPA)
  - **hide side-channel information** within power consumption
  - avoid predicting of intermediate results through execution randomization (*shuffling*)
  - supported naturally (memory address translation)
- providing **randomness** for side-channel countermeasure (random permutation generation)

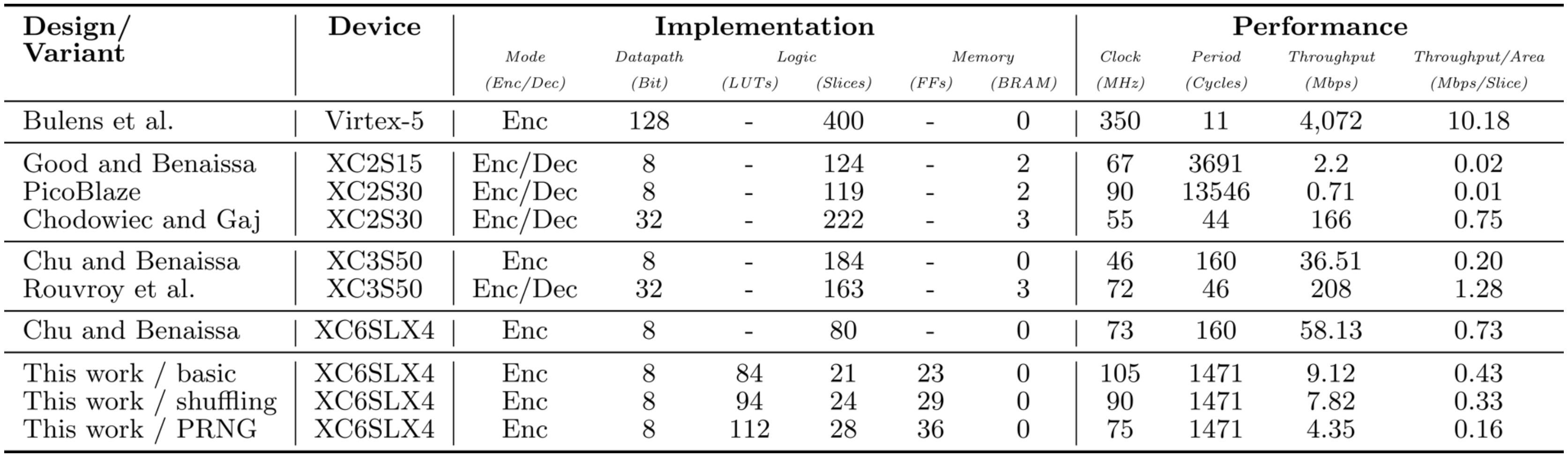
Problems:

- on-the-fly permutation generation for byte execution randomization and shuffling
- sufficient fresh randomness per encryption (has to be provided by PRNG)

Solutions:

- permutation generation through random swapping of elements (of previous permutation)
  - can be done using again *Distributed Memory*
  - performed in parallel with first key addition operation to hide additional latency
- randomness extraction from initial seed using AES-128 core as CPRNG
  - one extraction provides randomness for two encryptions (128 bits in total)
  - overall performance is decreased by 1/3

IMPLEMENTATION RESULTS AND COMPARISON



Design/ Variant	Device	Implementation						Performance			
		Mode	*Datapath*	*Logic*		*Memory*		*Clock*	*Period*	*Throughput*	*Throughput/Area*
		(Enc/Dec)	*(Bit)*	*(LUTs)*	*(Slices)*	*(FFs)*	*(BRAM)*	*(MHz)*	*(Cycles)*	*(Mbps)*	*(Mbps/Slice)*
Bulens et al.	Virtex-5	Enc	128	-	400	-	0	350	11	4,072	10.18
Good and Benaissa	XC2S15	Enc/Dec	8	-	124	-	2	67	3691	2.2	0.02
PicoBlaze	XC2S30	Enc/Dec	8	-	119	-	2	90	13546	0.71	0.01
Chodowiec and Gaj	XC2S30	Enc/Dec	32	-	222	-	3	55	44	166	0.75
Chu and Benaissa	XC3S50	Enc	8	-	184	-	0	46	160	36.51	0.20
Rouvroy et al.	XC3S50	Enc/Dec	32	-	163	-	3	72	46	208	1.28
Chu and Benaissa	XC6SLX4	Enc	8	-	80	-	0	73	160	58.13	0.73
This work / basic	XC6SLX4	Enc	8	84	21	23	0	105	1471	9.12	0.43
This work / shuffling	XC6SLX4	Enc	8	94	24	29	0	90	1471	7.82	0.33
This work / PRNG	XC6SLX4	Enc	8	112	28	36	0	75	1471	4.35	0.16

- most related work was designed for older Spartan-3 devices
  - devices do not support *Distributed Memory* feature
  - a fair comparison to those implementations is hardly possible

- recent work of **Chu and Benaissa** provides results for Spartan-6 devices
  - application of *Shift Registers* instead of *Distributed Memory*
  - mixed data path (32-bit for MixColumns, 8-bit otherwise)
  - still better performance than our design due to lower number of clock cycles per round

SIDE-CHANNEL ANALYSIS UNDER LAB CONDITIONS

Lab conditions:

- isolated environment
- optimal conditions for observer

Evaluation:

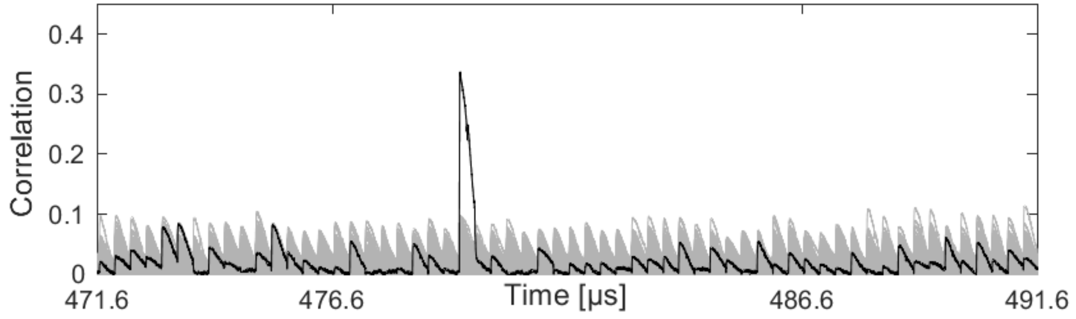
- CPA with HW model
- random inputs
- S-box input of last round

Test 1:

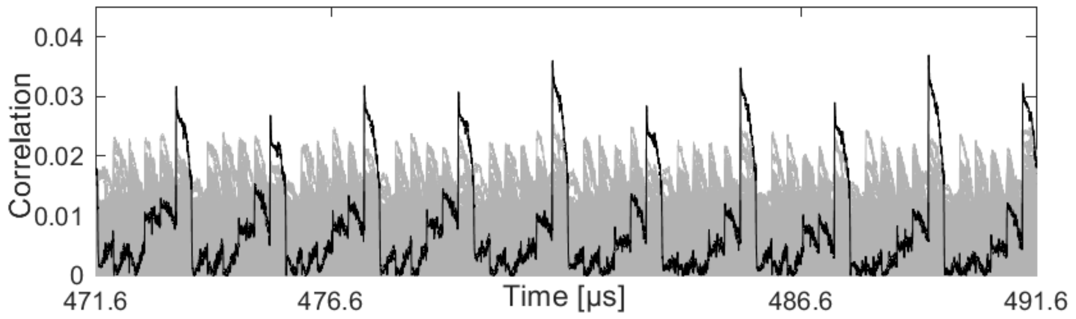
- verification setup
- shuffling disabled
- 100,000 traces
- *correct key*: 200 traces

Test 2:

- shuffling enabled
- 500,000 traces
- *correct key*: 50,000 traces



Test 1: without shuffling (unprotected, S-box input)



Test 2: with shuffling (protected, S-box input)

SIDE-CHANNEL ANALYSIS UNDER PRACTICAL CONDITIONS

Practical conditions:

- arithmetic computations in parallel (arithmetic noise)
- 80% of FPGA occupied

Evaluation:

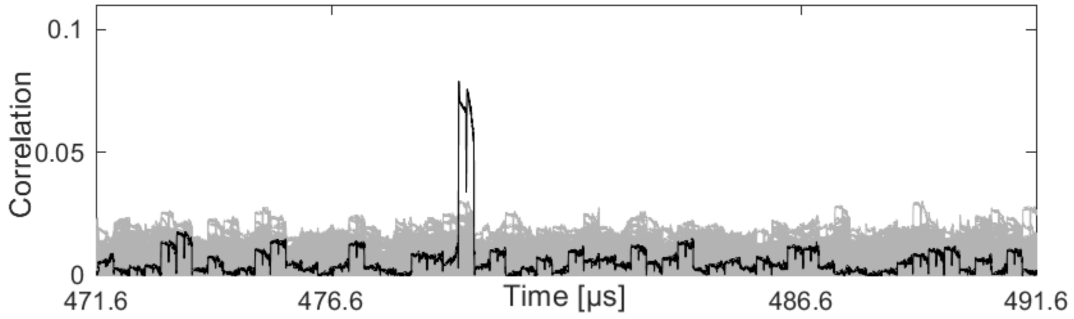
- CPA with HW model
- random inputs
- S-box input of last round

Test 3:

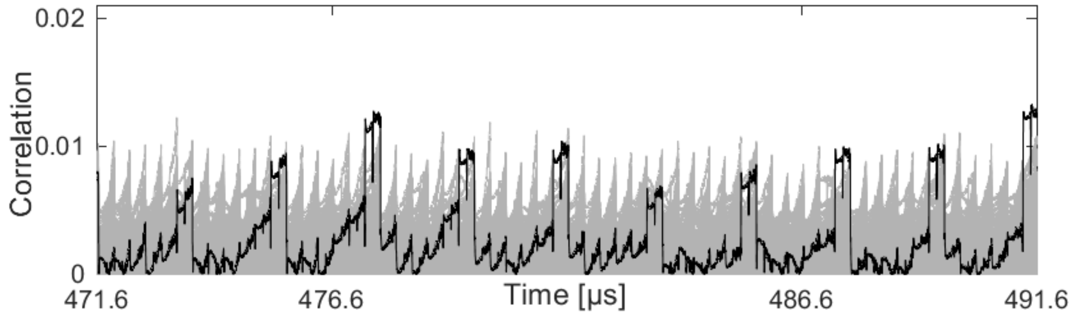
- verification setup
- shuffling disabled
- 100,000 traces
- *correct key*: 5,000 traces

Test 4:

- shuffling enabled
- 1,000,000 traces
- *correct key*: 800,000 traces



Test 3: without shuffling but arithmetic noise (unprotected, S-box input)



Test 4: with shuffling and arithmetic noise (protected, S-box input)

CONCLUSION

In this work, we have shown that:

- we can build an AES-128 core smaller than existing solutions.
- lightweight architectures do not have to exclude side-channel protection.
- our core can be used as PRNG in order to create a self-contained architecture.

We have designed the <u>smallest, side-channel protected, self-contained</u> AES-128 encryption core tailored for modern Xilinx FPGAs.

RUHR-UNIVERSITÄT BOCHUM

hg EMSEC

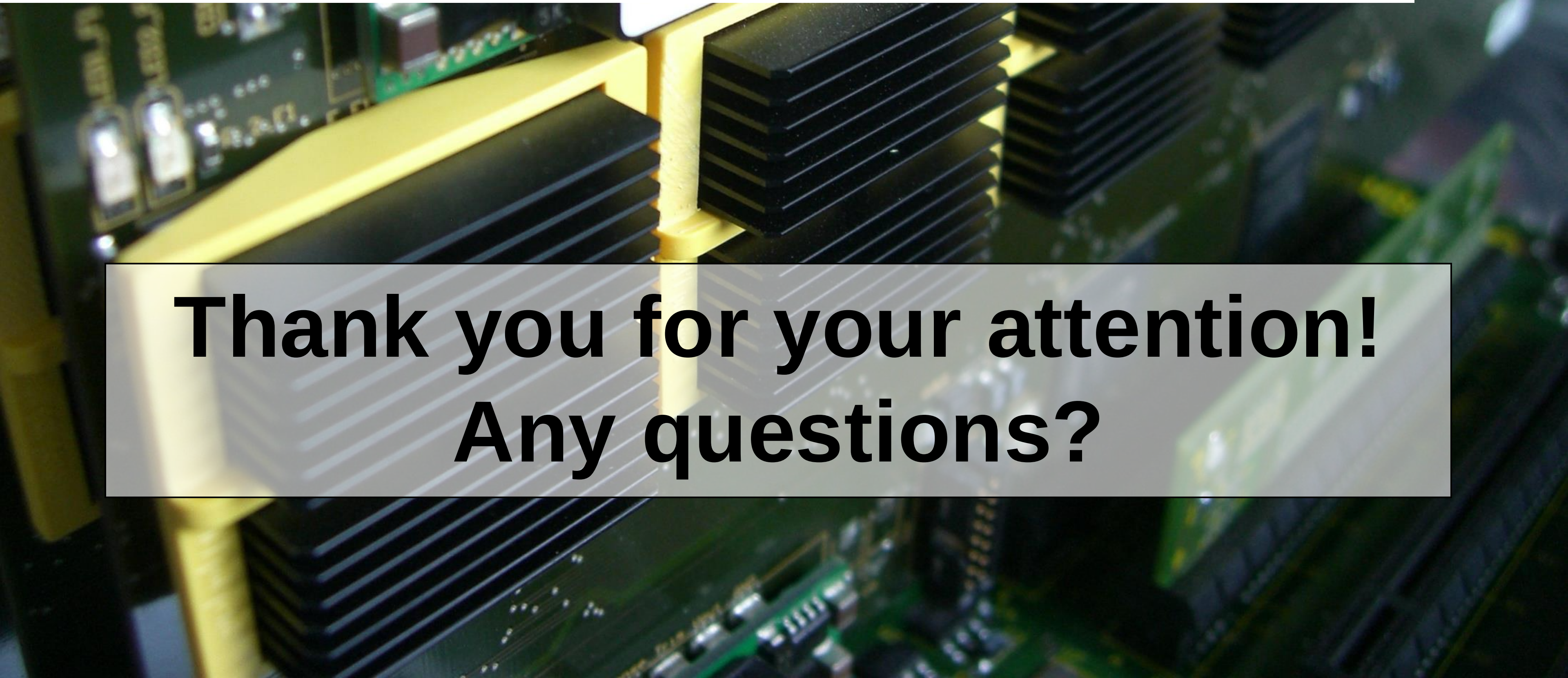
RUB

A GRAIN IN THE SILICON:
SCA-PROTECTED AES IN LESS THAN 30 SLICES

pascal.sasdrich@rub.de

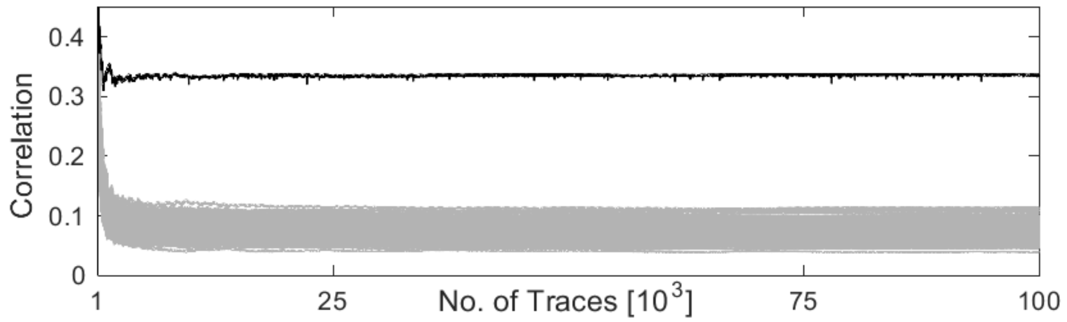
27TH IEEE CONFERENCE ON APPLICATION-SPECIFIC SYSTEMS, ARCHITECTURES AND PROCESSORS, LONDON, ENGLAND

JULY 6, 2016

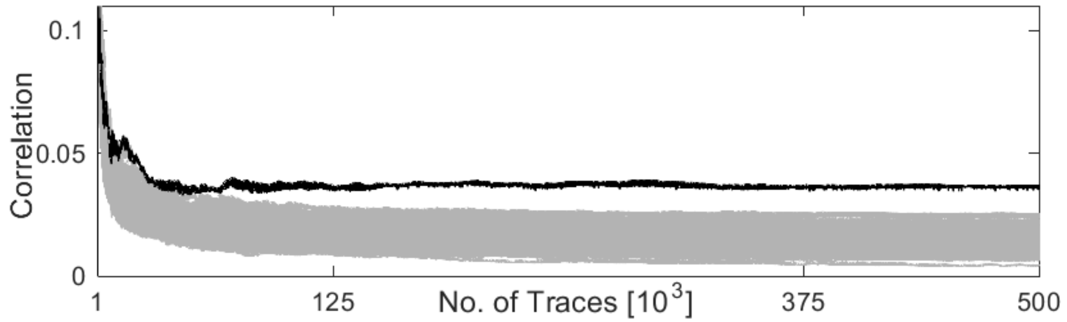


Thank you for your attention!
Any questions?

SIDE-CHANNEL ANALYSIS UNDER LAB CONDITIONS

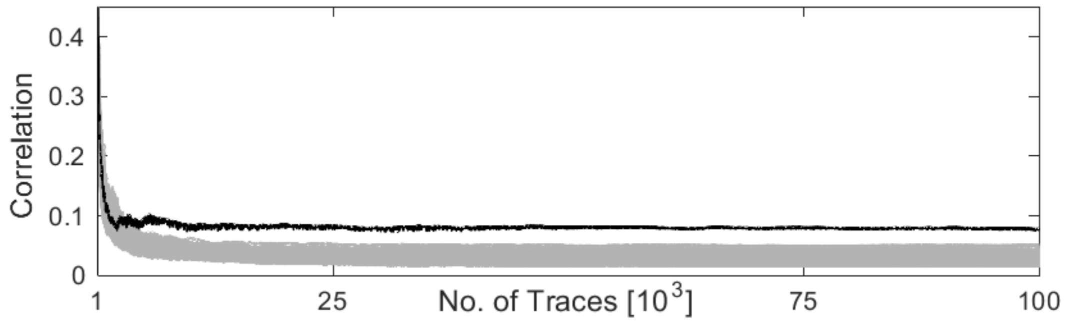


without shuffling (unprotected, S-box input)

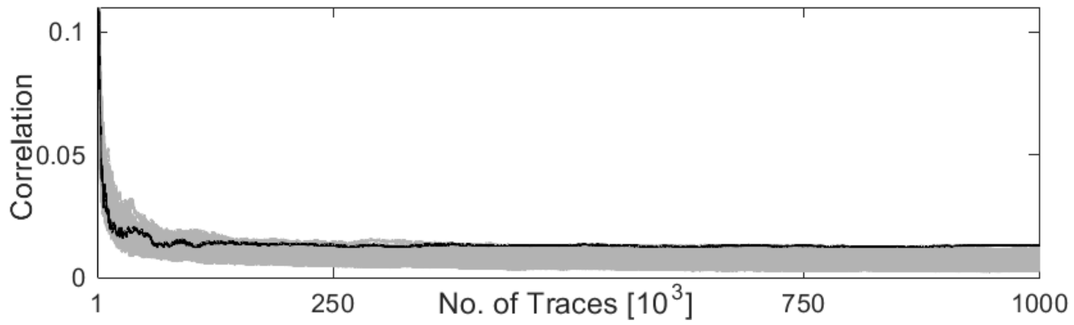


with shuffling (protected, S-box input)

SIDE-CHANNEL ANALYSIS UNDER PRACTICAL CONDITIONS



without shuffling but arithmetic noise (unprotected, S-box input)



with shuffling and arithmetic noise (protected, S-box input)