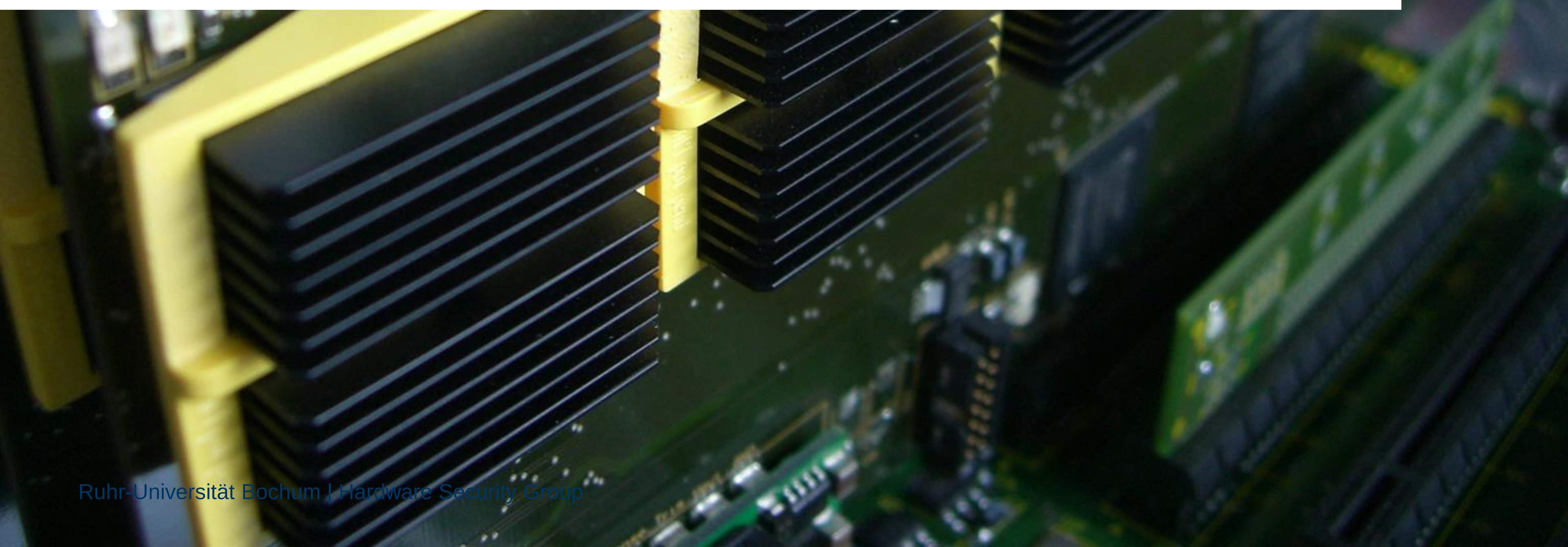
RUHR-UNIVERSITÄT BOCHUM

Side-Channel Protection by Randomizing Look-Up Tables on Reconfigurable Hardware
- Pitfalls of Memory Primitives -

COSADE 2015, Berlin, Germany

Pascal Sasdrich, Oliver Mischke, Amir Moradi, and Tim Güneysu

26.03.2015



Outline

Motivation

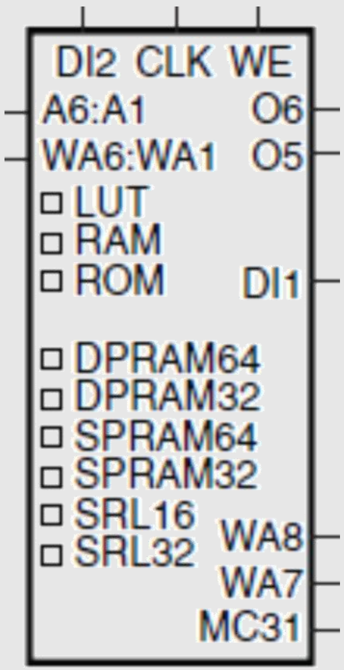
- At CHES 2011:

  Block Memory content Scrambling (BMS) was proposed as an effective way of 1st order side-channel protection.

- Our goals:
  - Analyze different ways for 1st order protection using randomized look-up tables.
  - Find optimal choices for FPGAs and 8-bit S-boxes.

Xilinx Memory Primitives

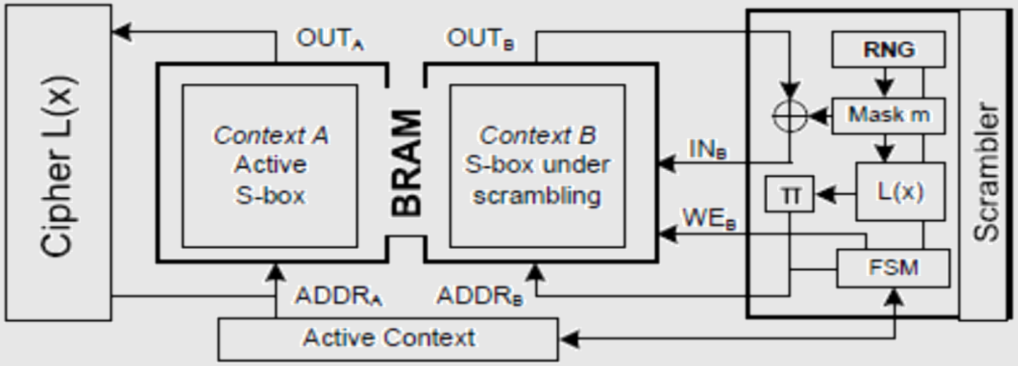
- Specific **slice logic** components can be implemented as distributed memory:
  - RAM32M (32x8bit SPRAM)
  - RAM64M (64x4bit SPRAM)
  - RAM256X1S (256x1bit SPRAM)
- Dedicated block memory primitives (RAMB8BWER) can be used as true dual-port block memory



Block Memory Content Scrambling

Main idea:

- 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.

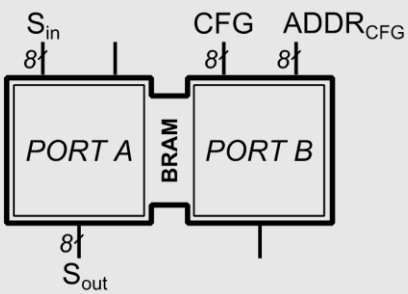
Disadvantages:

- Area overhead
- Latency
- Shared masks

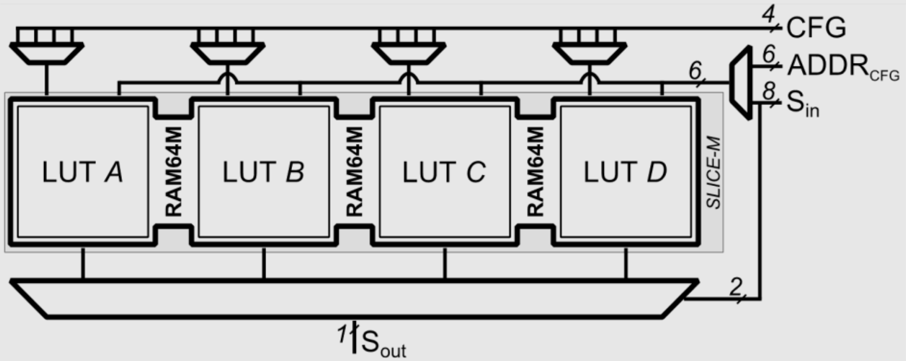
Contribution

1. Analyzed **Xilinx FPGA memory** primitives to prevent 1st order side-channel leakage.
2. Built **randomized look-up tables** of different memory primitives.
3. Evaluated designs using a state-of-the-art **leakage assessment methodology**.
4. We **revealed pitfalls** of using memory primitives for side-channel protection.

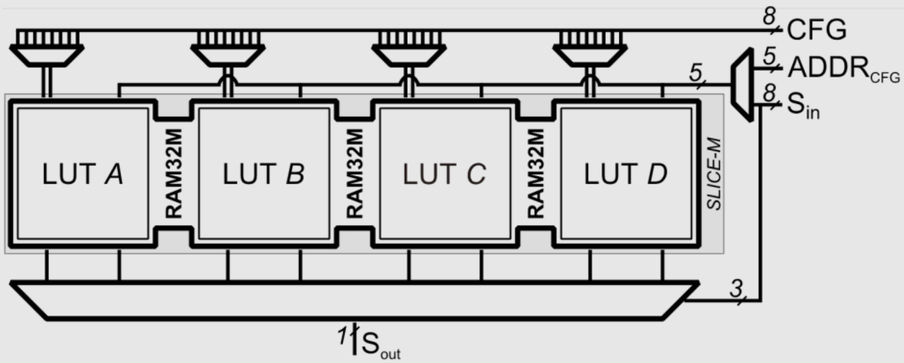
Randomized Look-Up Tables



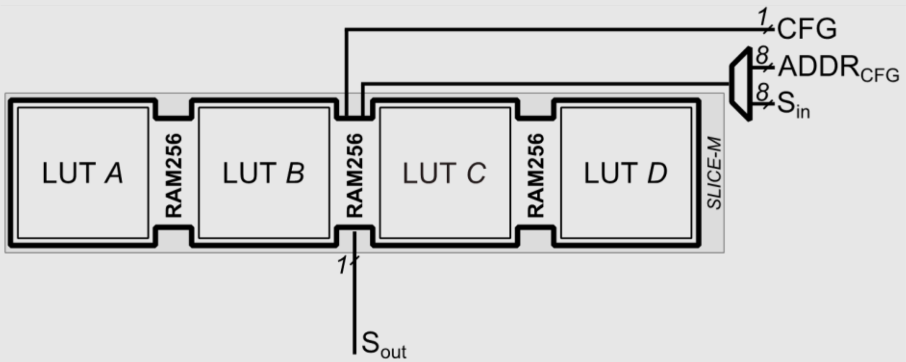
RAMB8BWER



RAM64M



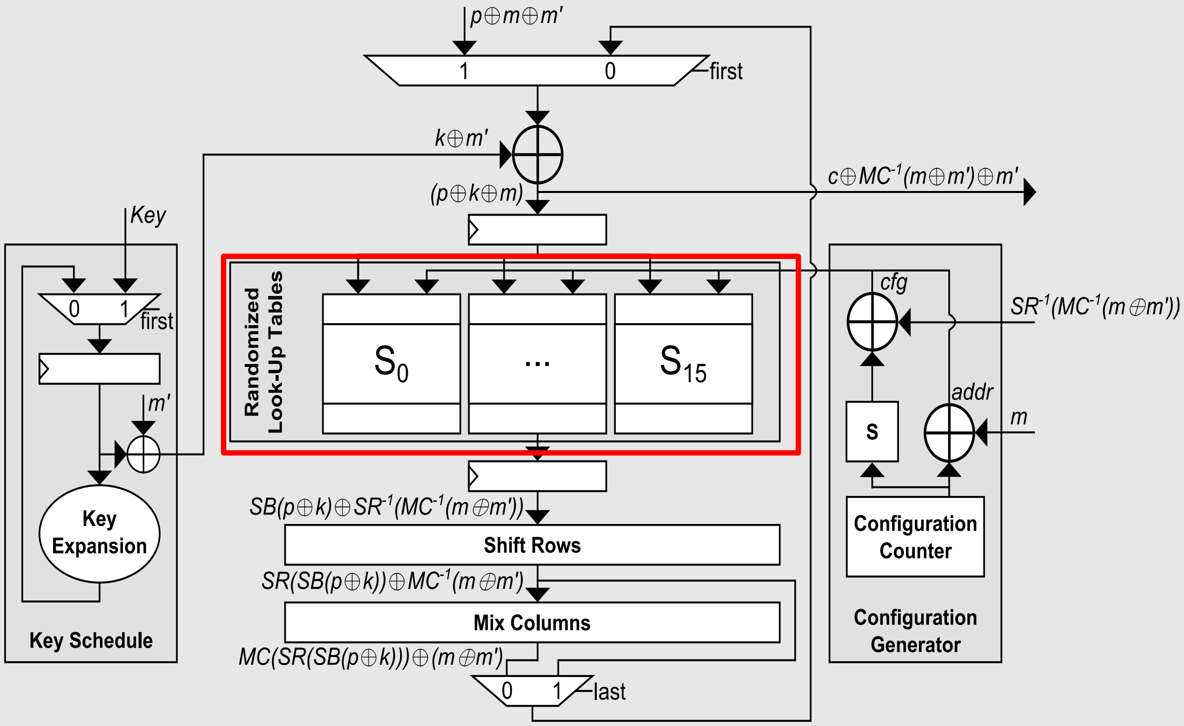
RAM32M



RAM256X1S

Side-Channel Protection by Randomizing Look-Up Tables on Reconfigurable Hardware
Pascal Sasdrich, Oliver Mischke, Amir Moradi, and Tim Güneysu

Case Study



- AES as case study
- Build randomized look-up tables using different memory primitives
- Replaced SubBytes with different designs.

S-Box Designs

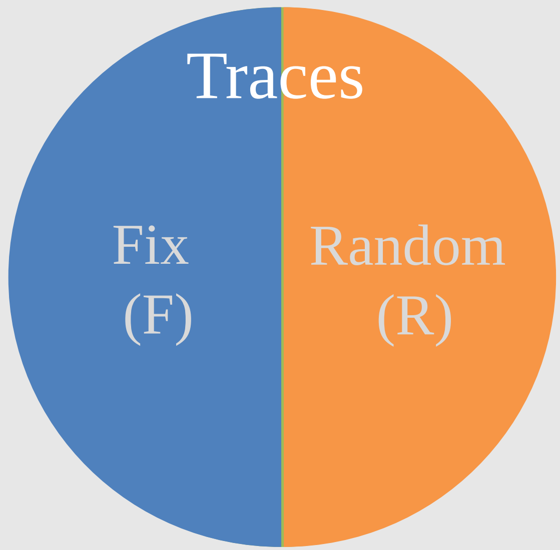
Memory Primitive	**SubBytes**			**Configuration**		**Max. Throughput**
	Logic *(LUT)*	*Dist. Mem.* *(LUT)*	*Block Mem.* *(BRAM16)*	*Logic* *(LUT)*	*Memory* *(FF)*	*(Mbit/s)*
BMS [CHES11]	-	-	16	1706	1169	35.4
RAMB8BWER	-	-	8	298	8	68.6
RAM256X1S	128	512	-	298	8	77.0
RAM64M	768	512	-	727	6	247.3
RAM32M	1920	512	-	1222	5	363.3

Evaluation

Welch's T-Test

fix vs. random:

- 1 fix plaintext



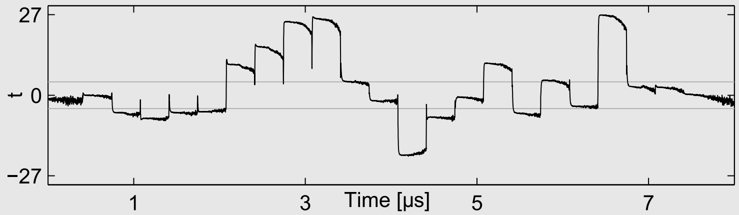
N_F = Size of F
X_F = Mean of F
S_F = Std. deviation of F

N_R = Size of R
X_R = Mean of R
S_R = Std. deviation of R

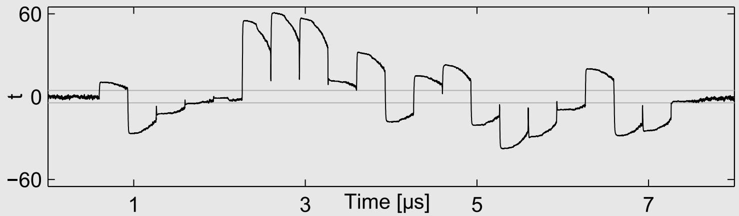
$$T = \frac{X_F - X_R}{\sqrt{\frac{S_F^2}{N_F} + \frac{S_R^2}{N_R}}}$$

Side-Channel Protection by Randomizing Look-Up Tables on Reconfigurable Hardware
Pascal Sasdrich, Oliver Mischke, Amir Moradi, and Tim Güneysu

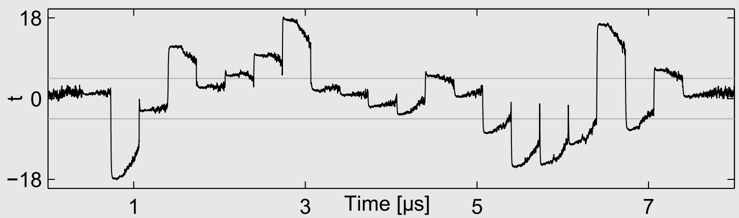
Results



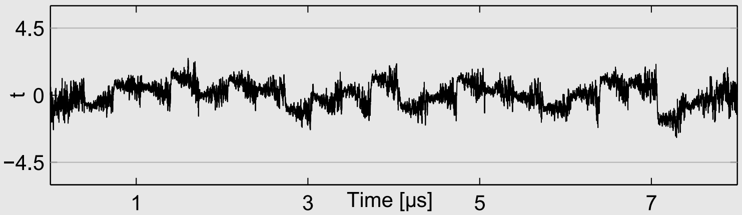
RAM32M (1 million traces)



RAM64M (1 million traces)



RAM256X1S (1 million traces)



RAMB8BWER (10 million traces)

- leakage is detectable for distributed memory primitives
- assume that leakage is due to internal slice architecture
- BRAM primitives exhibit no detectable leakage (even for larger trace numbers)

Conclusion

Our results infer the pitfall of using distributed memory primitives:

- Distributed memory causes a secure scheme to exhibit 1st order leakage.
- Replaced with Block Memory 1st order leakage is not detectable.

Besides, our designs achieve higher throughput and require less randomness than original BMS scheme.

RUB

Side-Channel Protection by Randomizing Look-Up Tables on Reconfigurable Hardware
- Pitfalls of Memory Primitives -

COSADE 2015, Berlin, Germany

Pascal Sasdrich, Oliver Mischke,
Amir Moradi, and Tim Güneysu

26.03.2015

Thank you for your attention!
Any Questions?