ROCKY : Rotation Countermeasure for the Protection of Keys and Other Sensitive Data

K. Miteloudi[1] L. Batina[1] J. Daemen[1] N. Mentens[2,3]

[1]iCIS - Digital Security Group, Radboud University, The Netherlands

[2]imec-COSIC - ES&S, ESAT, KU Leuven, Belgium

[3]LIACS, Leiden University, The Netherlands

SAMOS XXI - International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation
July 5 - July 7, 2021

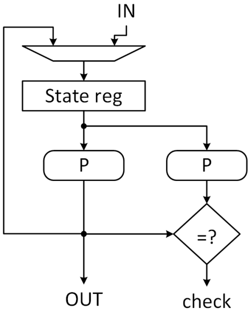
Outline

Fault attacks (FA)

- Physical disturbance
- Erroneous computation
- Leakage of secret information

Modular redundancy techniques:

- Efficient detection of many types of random faults
- Nullified when injecting the same fault to all executions or replications of the computation



Our Contribution

The basic idea

- every execution of the algorithm processes the internal data in a different, semi-randomized representation

New countermeasure ROCKY

- calculate a rotated representation of output based on a rotated representation of input
- applies to cryptographic primitives
- permutation with (almost) shift-invariant round functions

FPGA Hardware architectures for ROCKY

- examined the overhead of the countermeasure

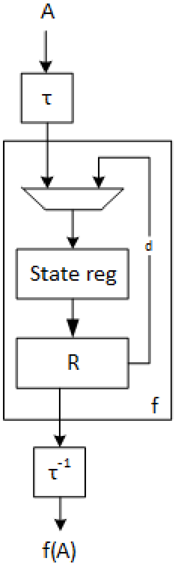
Shift-Invariance

- A cryptographic permutation f can be applied with a shift-invariant round function to a shifted version of a state A.
- Let $f = \mathrm{R}^d$ with d the number of rounds, then we have:

$$f(A) = \tau^{-1}(f(\tau(A))$$

- If the round function includes the addition of a round constant C and assume without loss of generality that the round constant is added at the end of the round. Let $B = \mathrm{C} + \mathrm{R}(\mathrm{A})$, then:

$$B = \tau^{-1}(\tau C + \mathrm{R}(\tau(A)))$$



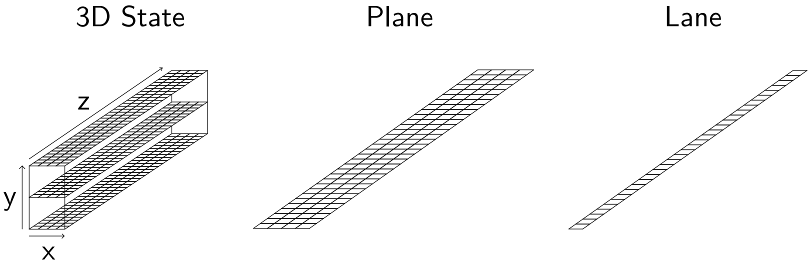
Cryptographic Permutations

- with (almost) shift-invariant round function

Primitive	state size	width	height	depth	Round constant
Salsa	512	**4**	**4**	(32)	no
Chacha	512	**4**	**4**	(32)	no
Keccak-f	1600	**64**	5	5	yes
Ascon	320	**64**	5	(1)	yes
Xoodoo	384	**32**	3	**4**	yes
Subterranean	257	**257**	(1)	(1)	yes
AES unkeyed	128	**4**	4	(8)	no

- Our case study is Xoodoo.

State representation of Xoodoo



- $(x = 4 \ , \ y = 3 \ , \ z = 32) \Rightarrow$ 3 planes and 12 lanes of 32 bits
- Mapping from 3D State to 1D bit array : $i = z + 32(x + 4y)$.

Xoodoo Algorithm

Algorithm 1: Definition of Xoodoo[n_r] with n_r the number of rounds

Parameters: Number of rounds n_r

for *Round index i from 1 - n_r to 0* **do**

  $A = R_i(A)$

end

Here R_i is specified by the following steps:

θ :

$$P \leftarrow A_0 \oplus A_1 \oplus A_2$$
$$E \leftarrow P \lll (1,5) \oplus P \lll (1,14)$$
$$A_y \leftarrow A_y \oplus E \quad for\ y \in \{0,1,2\}$$

ρ_{west} :

$$A_1 \leftarrow A_1 \lll (1,0)$$
$$A_2 \leftarrow A_2 \lll (0,11)$$

ι :

$$A_0 \leftarrow A_0 \oplus C_i$$

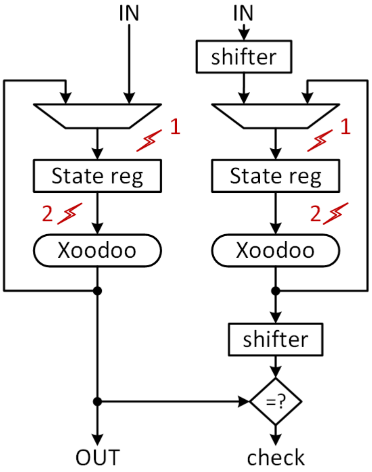
χ :

$$B_0 \leftarrow \overline{A_1} \bullet A_2$$
$$B_1 \leftarrow \overline{A_2} \bullet A_0$$
$$B_2 \leftarrow \overline{A_0} \bullet A_1$$
$$A_y \leftarrow A_y \oplus B_y \quad for\ y \in \{0,1,2\}$$

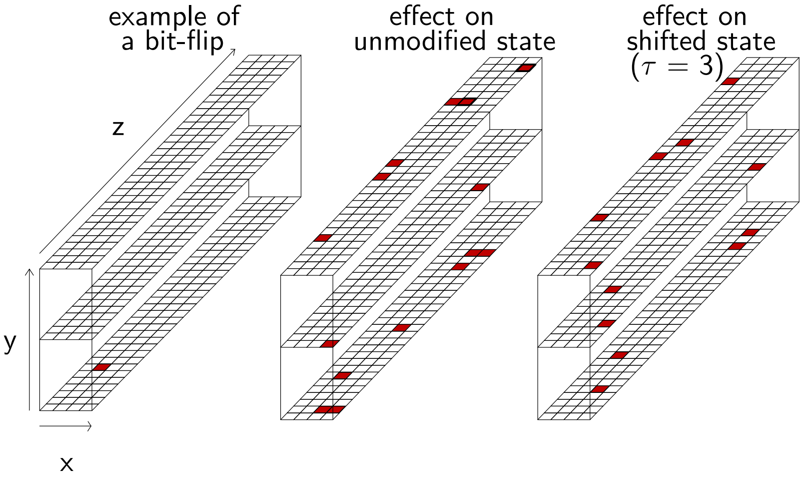
ρ_{east} :

$$A_1 \leftarrow A_1 \lll (0,1)$$
$$A_2 \leftarrow A_2 \lll (2,8)$$

- 100% fault detection
- if shift value τ is unknown
- or attacker cannot guess it

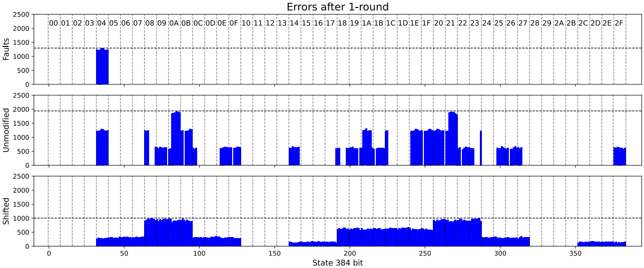


Example of errors on state after 1 round of Xoodoo



Distribution of errors on the State

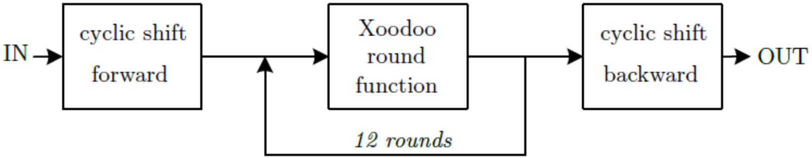
- Simulation experiment in python (sample 10K)
- Faults induced with a random bit-flip (uniform distribution) in the state of Xoodoo at a specific group of eight bits (0x04)
- In order to examine the effect the fault space of the State for:
  - an unmodified version of the algorithm (middle) and
  - the state is randomly shifted before every execution (bottom).



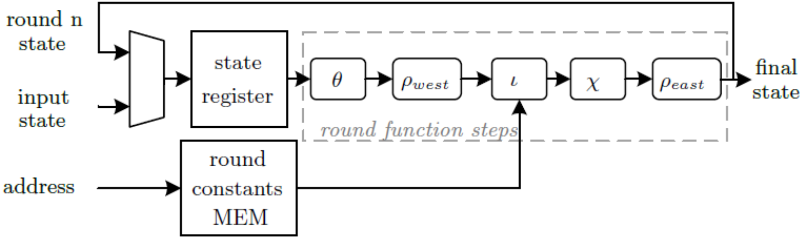
- A larger fault space makes it more difficult for the adversary to achieve the desired fault with the desired precision.

FPGA implementation

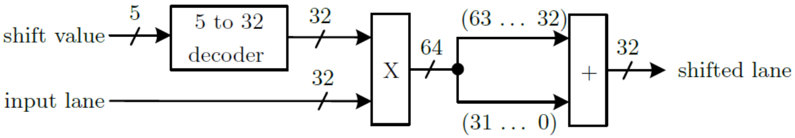
Top level Architecture



Xoodoo module

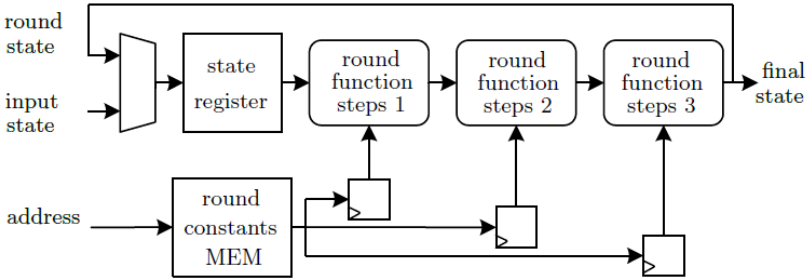


Cyclic Shift module

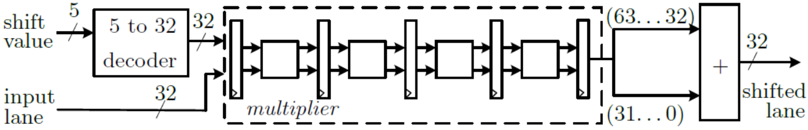


- Constant-time input rotation, independently of the rotation value.

Xoodoo module with 3 rounds/cycle



Cyclic Shift module with 5-stage pipeline



- Target board: Xilinx Artix-7 FPGA (XC7A100T-FTG256)

Timing

Architecture	Clock cycles	Clock period	Latency
Comb. mult. & 1 round/cycle	40 (5.26%)	9.8 ns (196%)	392 ns (212.6%)
Comb. mult. & 3 rounds/cycle	32 (-15.8%)	9.8 ns (196%)	313.6 ns (150%)
5-stage pipel. mult. & 1 round/cycle	50 (31.6%)	3.9 ns (18.18%)	195 ns (55.5%)
Unprotected	38 (- %)	3.3 ns (- %)	125.4 ns (- %)

Resources

Architecture	LUTS	Flip-Flops	DSP48E1	BRAM
Comb. mult. & 1 round/cycle	1,083 (-18.9%)	1,335 (10.2%)	8	1
Comb. mult. & 3 rounds/cycle	2,383 (78.5%)	1,758 (31.1%)	8	1
5-stage pipel. mult. & 1 round/cycle	1,452 (8.76%)	1,544 (27.5%)	8	1
Unprotected	1,335 (- %)	1,211 (- %)	0	0

Conclusion and next steps

Three designs for Xoodoo optimized for FPGA

- Protected with ROCKY
  - basic version
  - optimized number of cycles
  - optimized clock frequency

Overhead of ROCKY implementation

- in terms of latency and area
- results appear to be promising

Further research is in progress:

- design optimizations for FPGA and ASIC oriented architectures
- security analysis for Side Channel Attacks and Fault Attacks

Rocky

Thanks you all!