Laser Fault Attack on Physically Unclonable Functions

Shahin Tajik, Heiko Lohrke, Fatemeh Ganji,
Jean-Pierre Seifert, Christian Boit

HALBLEITERBAUELEMENTE

Telekom Innovation Laboratories

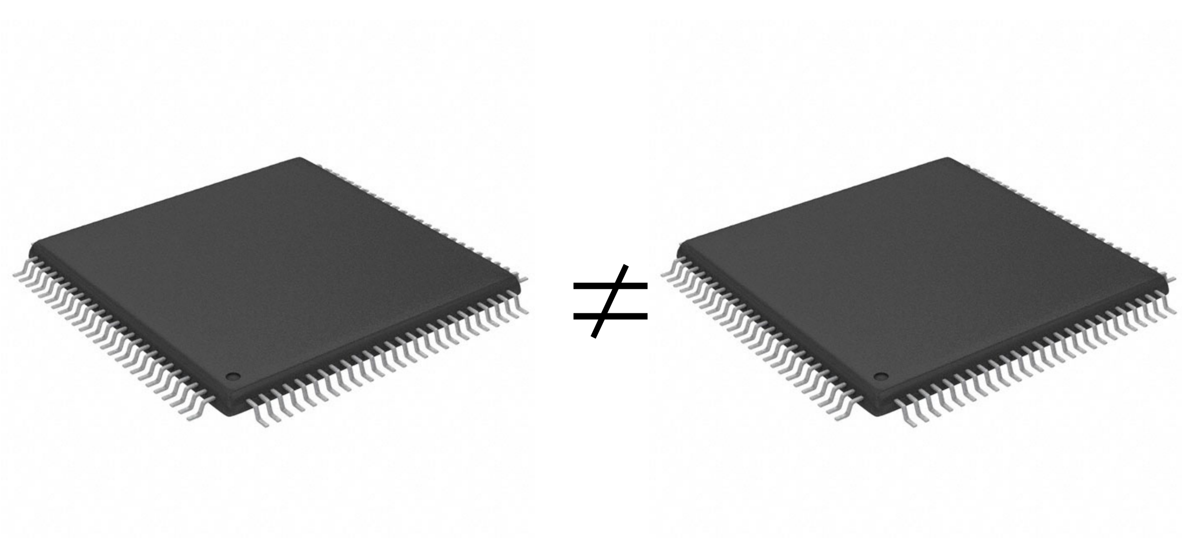
Helmholtz Research School on Security Technologies

Physically Unclonable Functions (PUFs)

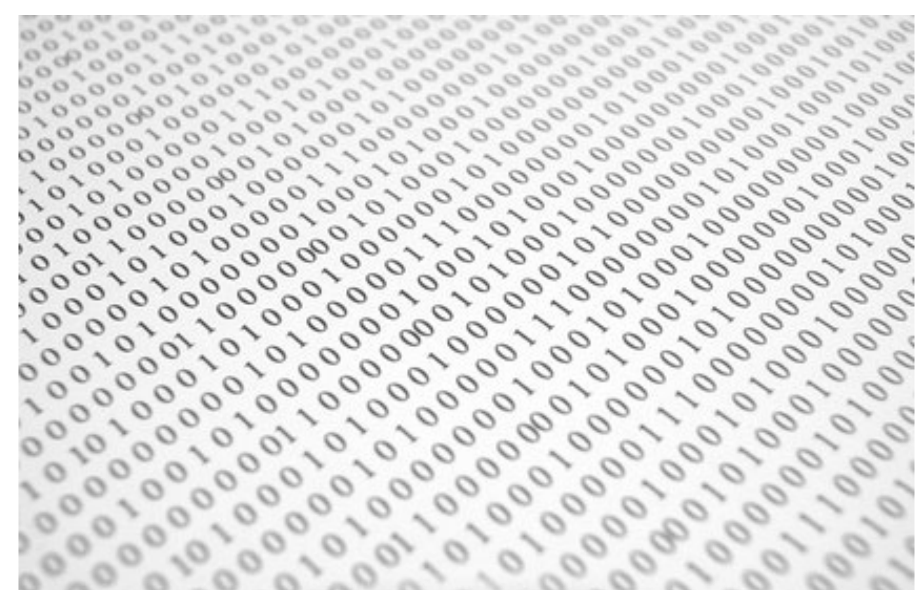
Utilizing manufacturing processing variations on different chips

Challenge → **PUF** → **response**

Device Authentication



Key Generation



Which PUF is better?

- **Authentication:** PUFs with large challenge spaces:
  - e.g., **Arbiter PUF Family** & **Bistable Ring PUF**
- **Key Generation Generation:** PUF with high response entropies,
  - e.g., **Ring-oscillator PUF**

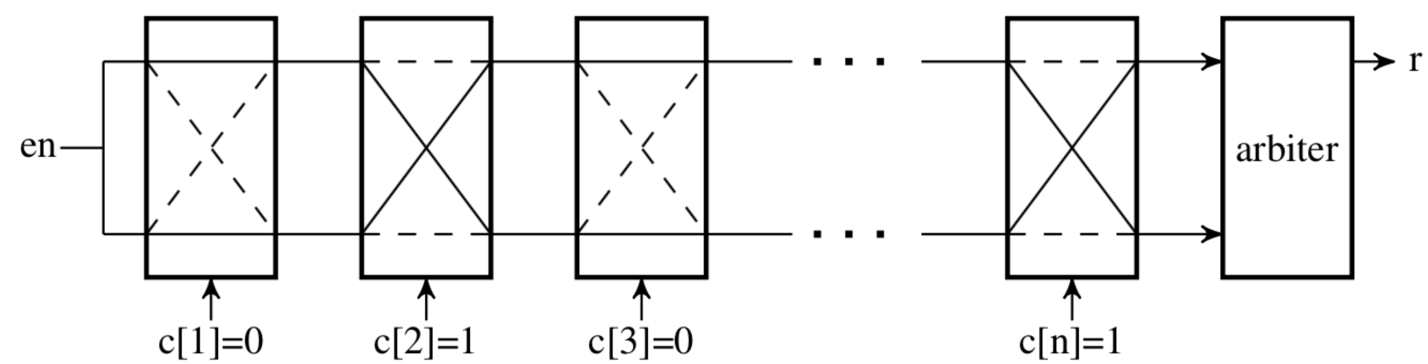
Authentication Scenario

Arbiter PUF Family

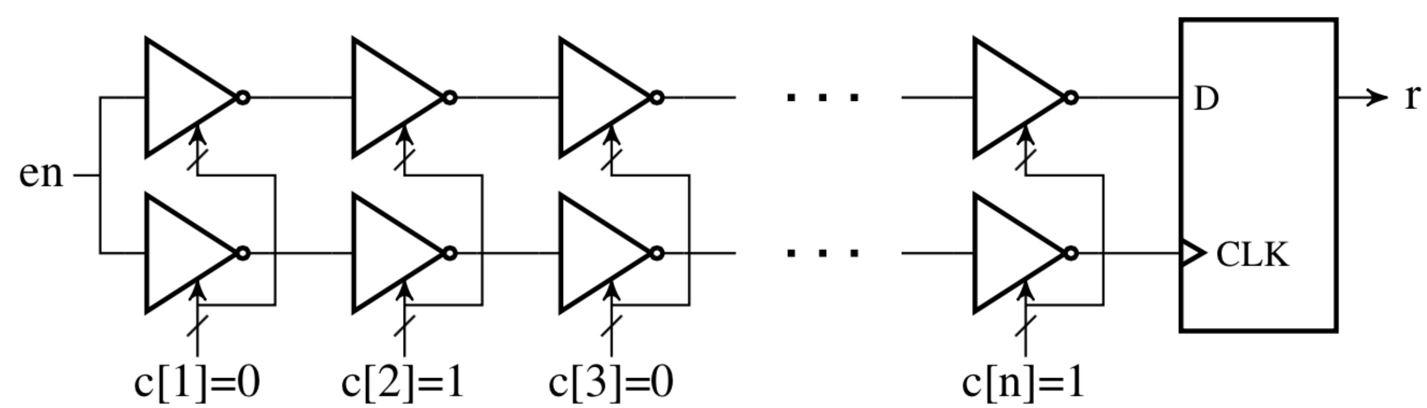
Advantage: Large Challenge space for **authentication**

Disadvantage: Vulnerable to Machine Learning

- **Experimentally and Theoretically broken by ML Attacks!**
- **Arbiter PUF is PAC-Learnable!**



Schematic

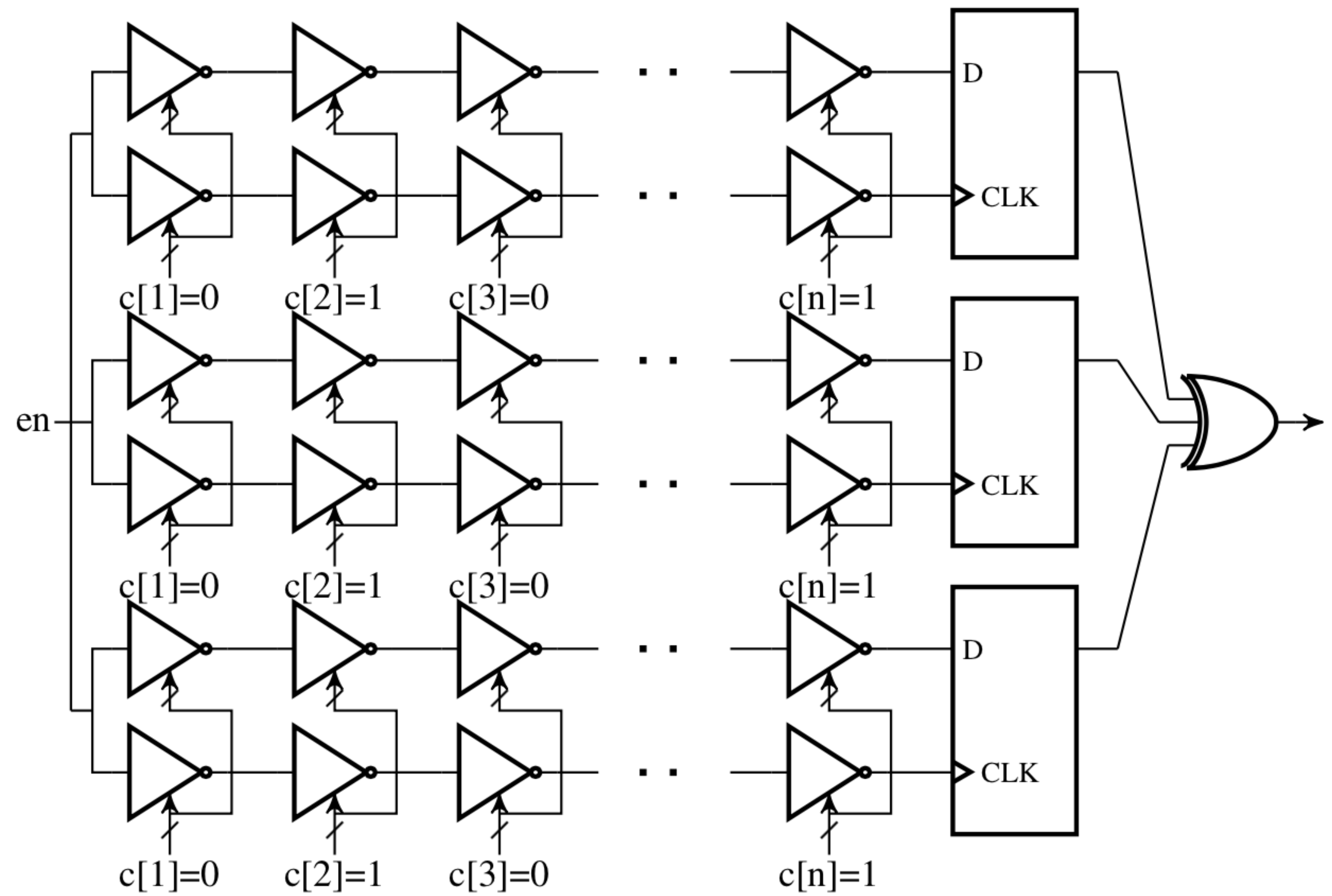


Implementation on FPGA

Countermeasure to ML Attacks: XOR Arbiter PUF

- ✳ **With limited number of arbiter chains: still vulnerable to ML attacks!!!**

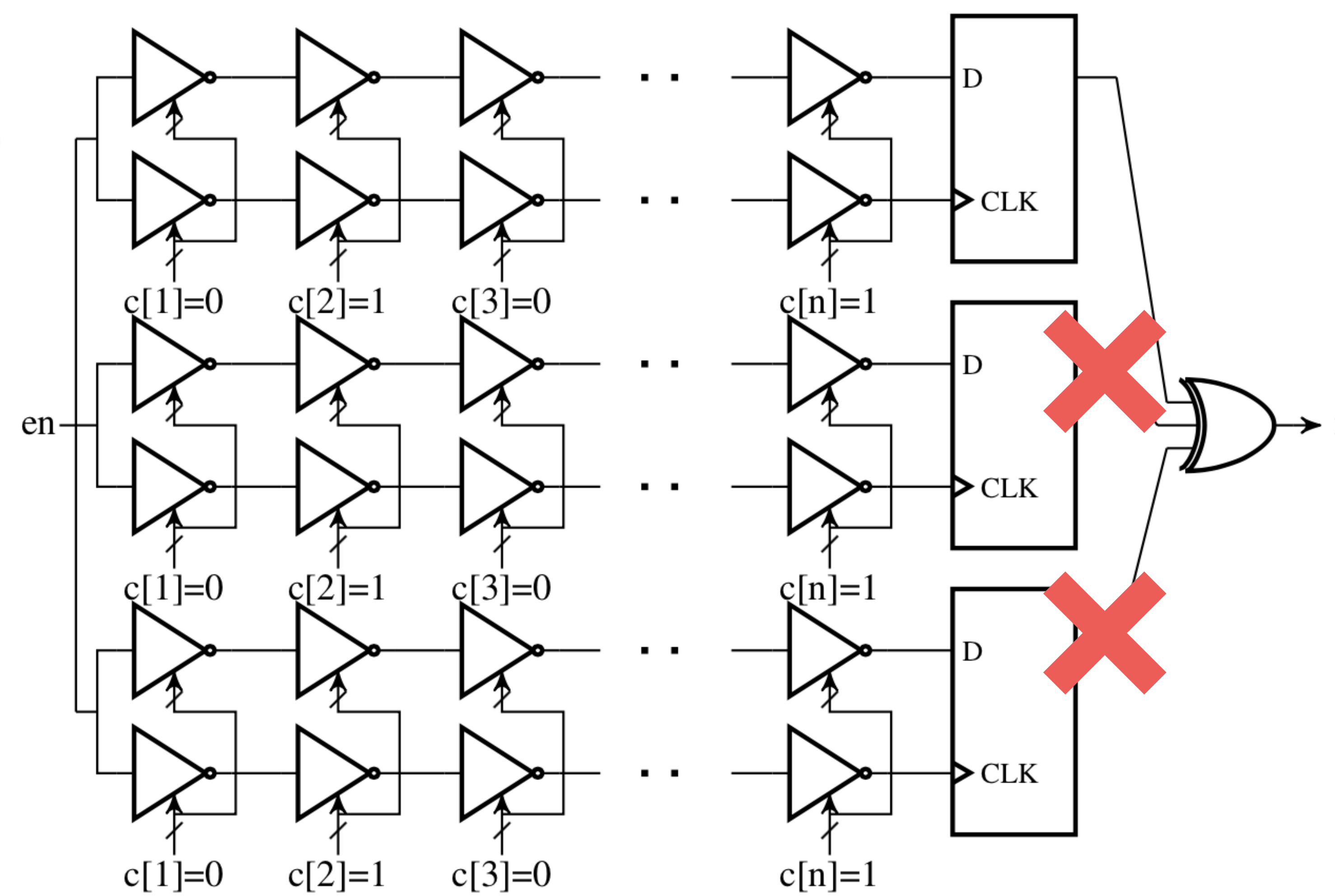
- ★ **However, large number of arbiter chains cannot be learned in polynomial time!**



e.g., 3-XOR arbiter PUF

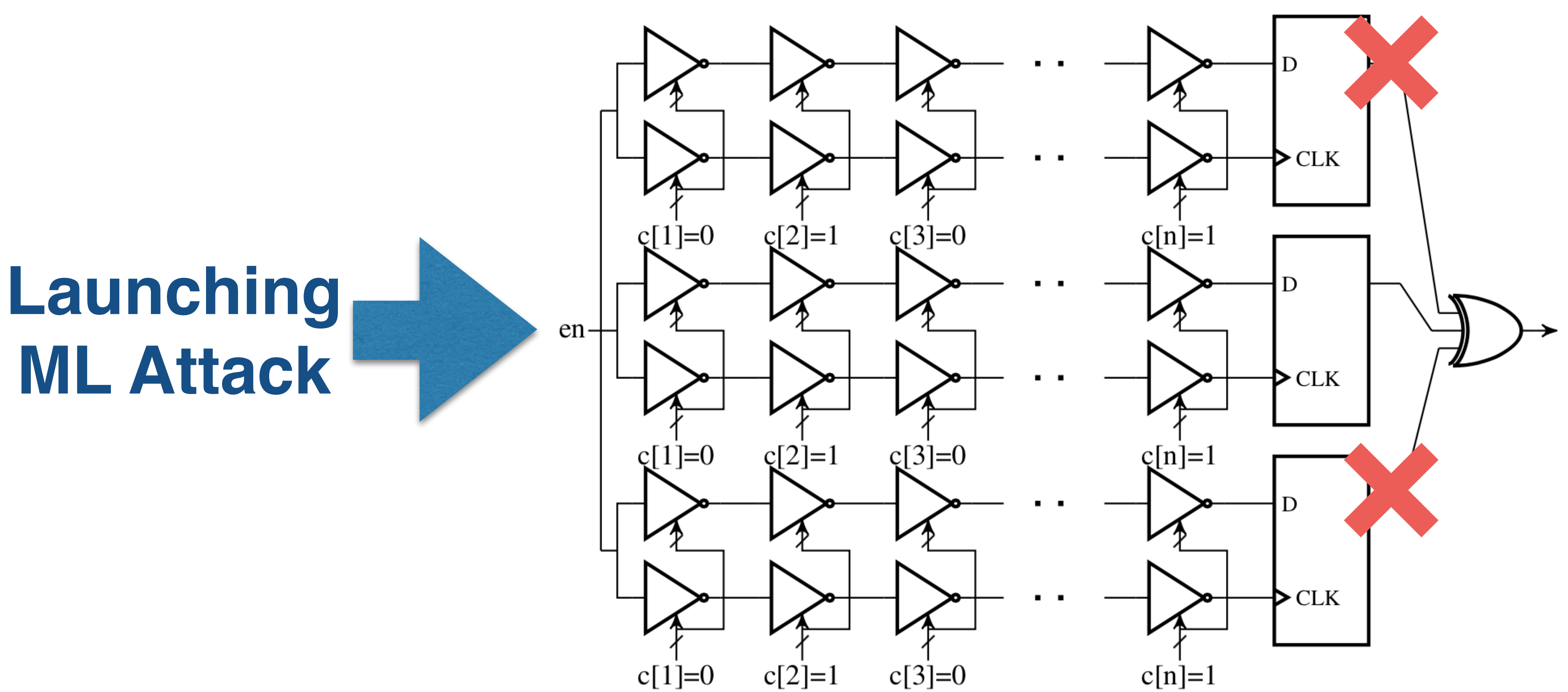
Simplifying ML attacks by Deactivating all Arbiter Chains Except One!

Launching ML Attack



e.g., 3-XOR arbiter PUF

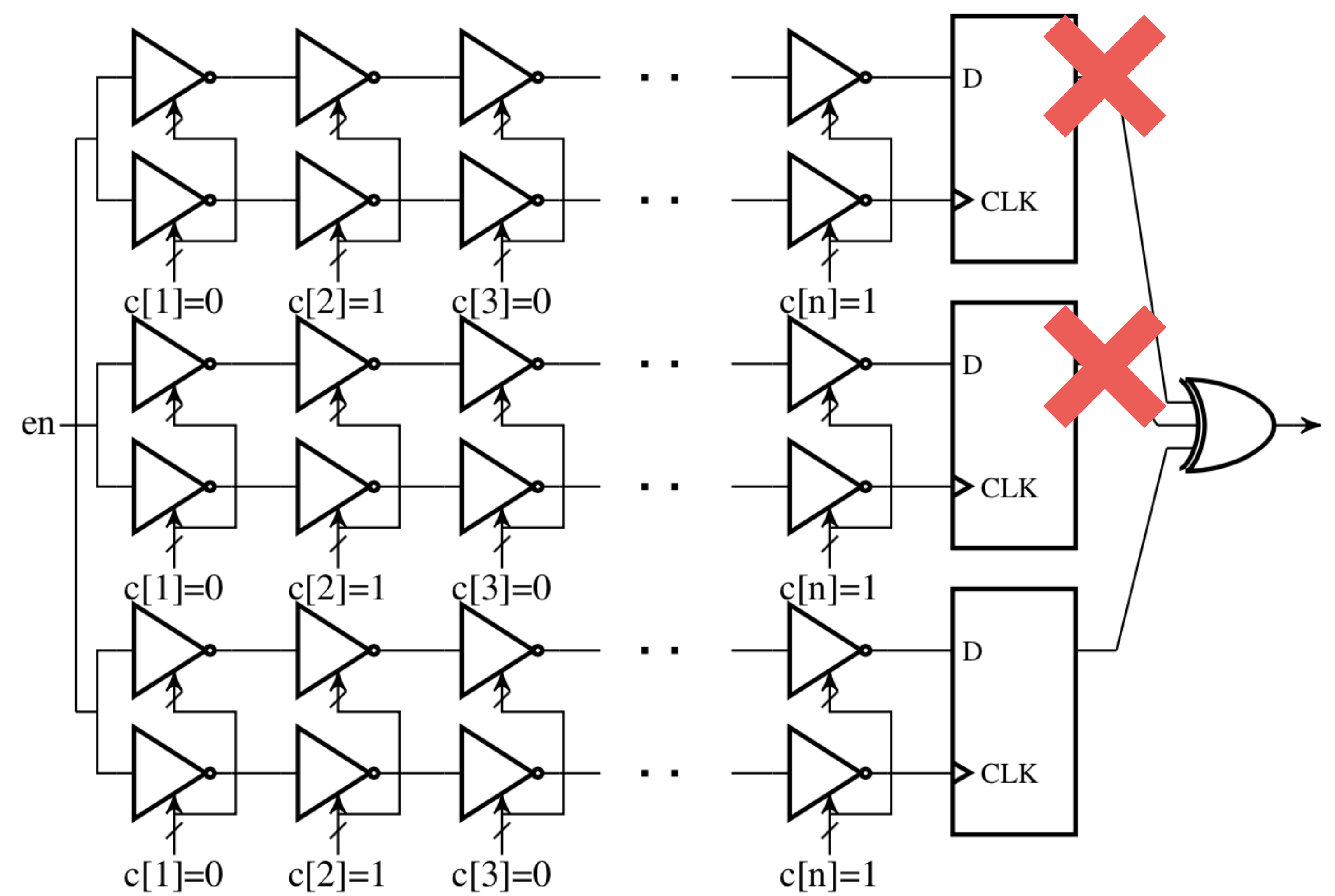
Simplifying ML attacks by Deactivating all Arbiter Chains Except One!



e.g., 3-XOR arbiter PUF

Simplifying ML attacks by Deactivating all Arbiter Chains Except One!

Launching ML Attack

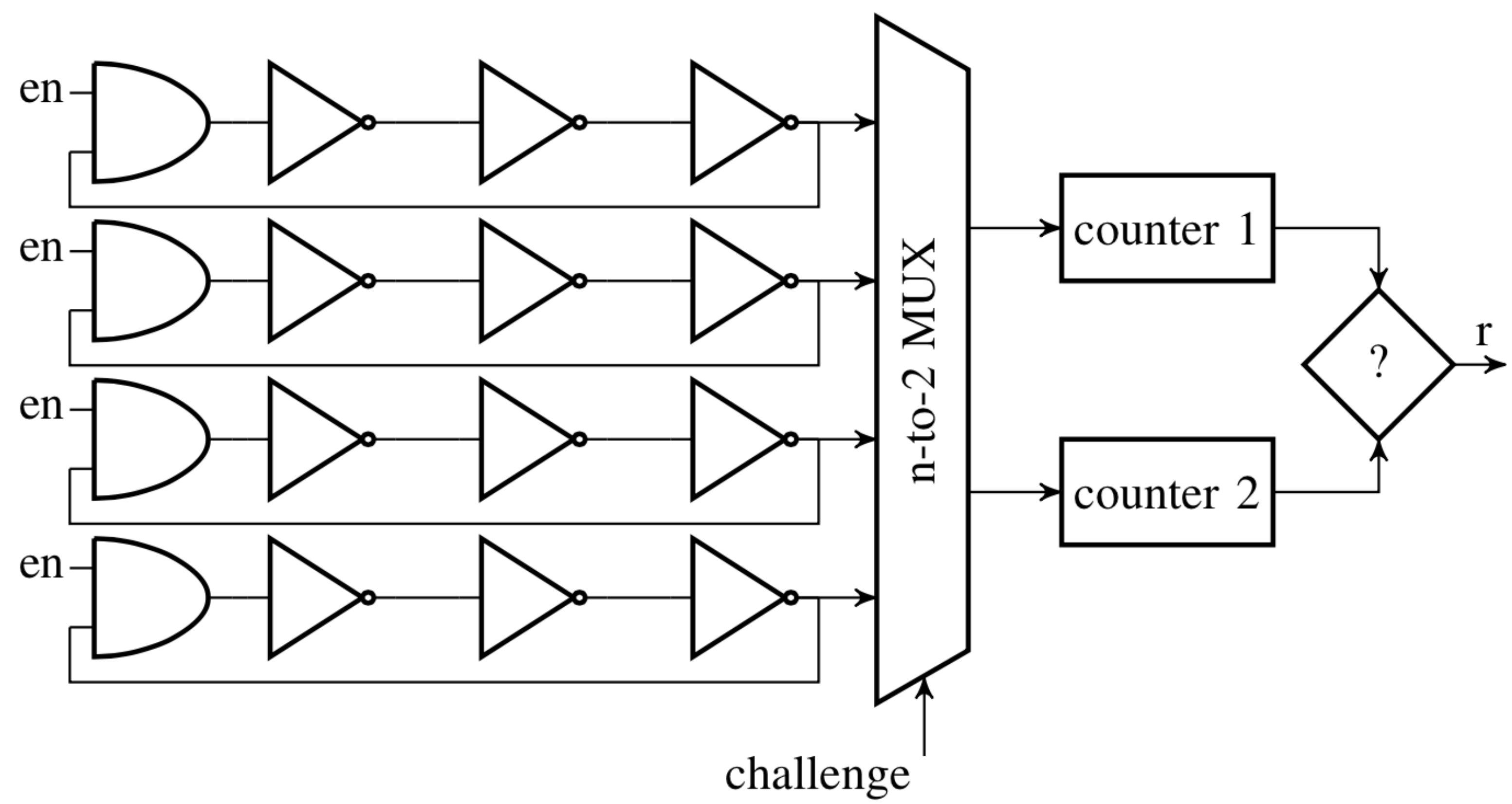


e.g., 3-XOR arbiter PUF

Key Generation Scenario

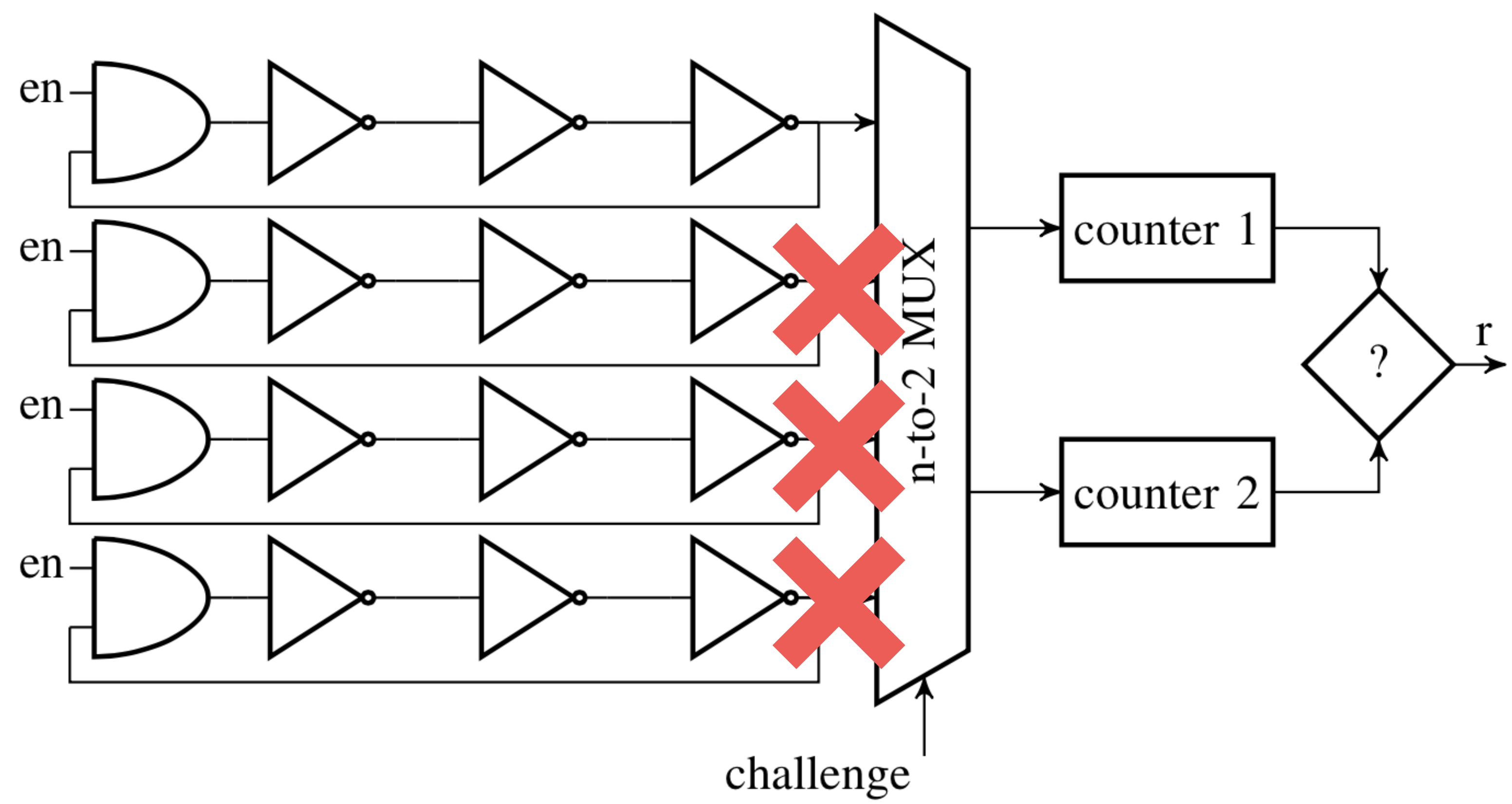
Ring-oscillator PUF

- N ring-oscillators
- Entropy density in the PUF response >> $\boldsymbol{\log_2(N!)}$



Reducing the Entropy of the PUF responses

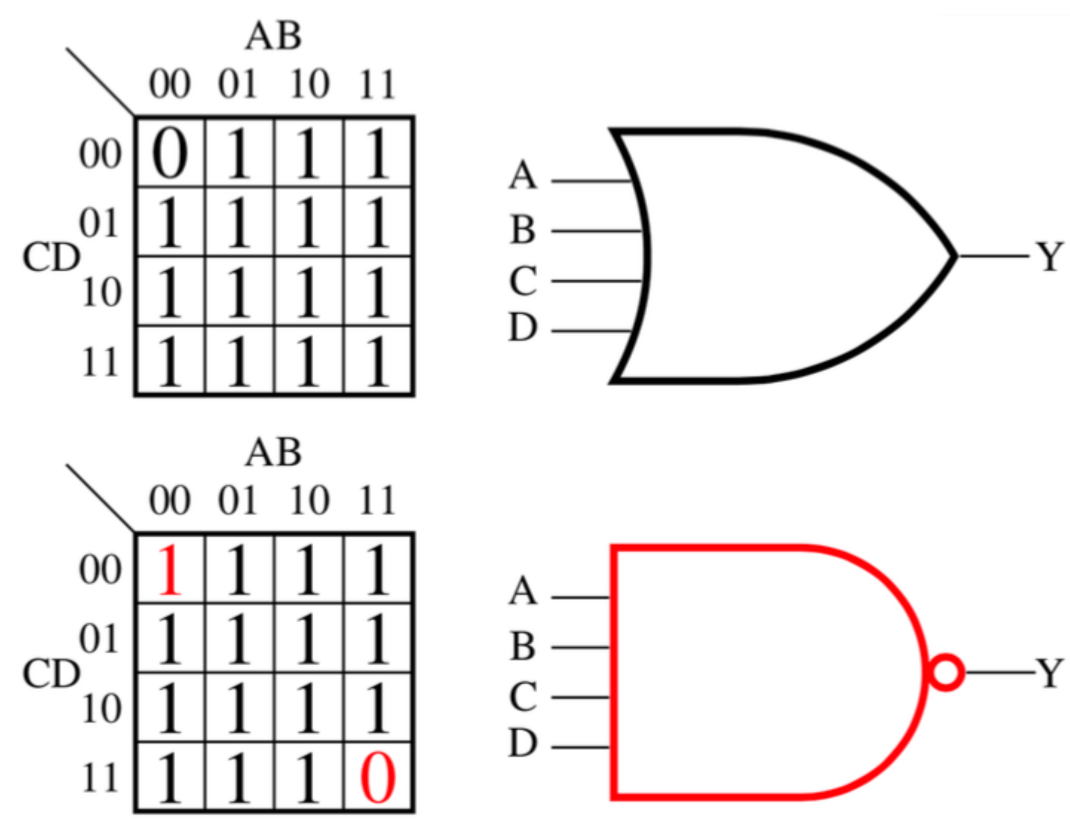
- Reduction of the entropy of the generated random numbers: ***$log_2(N!) >> log_2((N-x)!)$***



Fault Injection into the Configuration Memory of LUTs

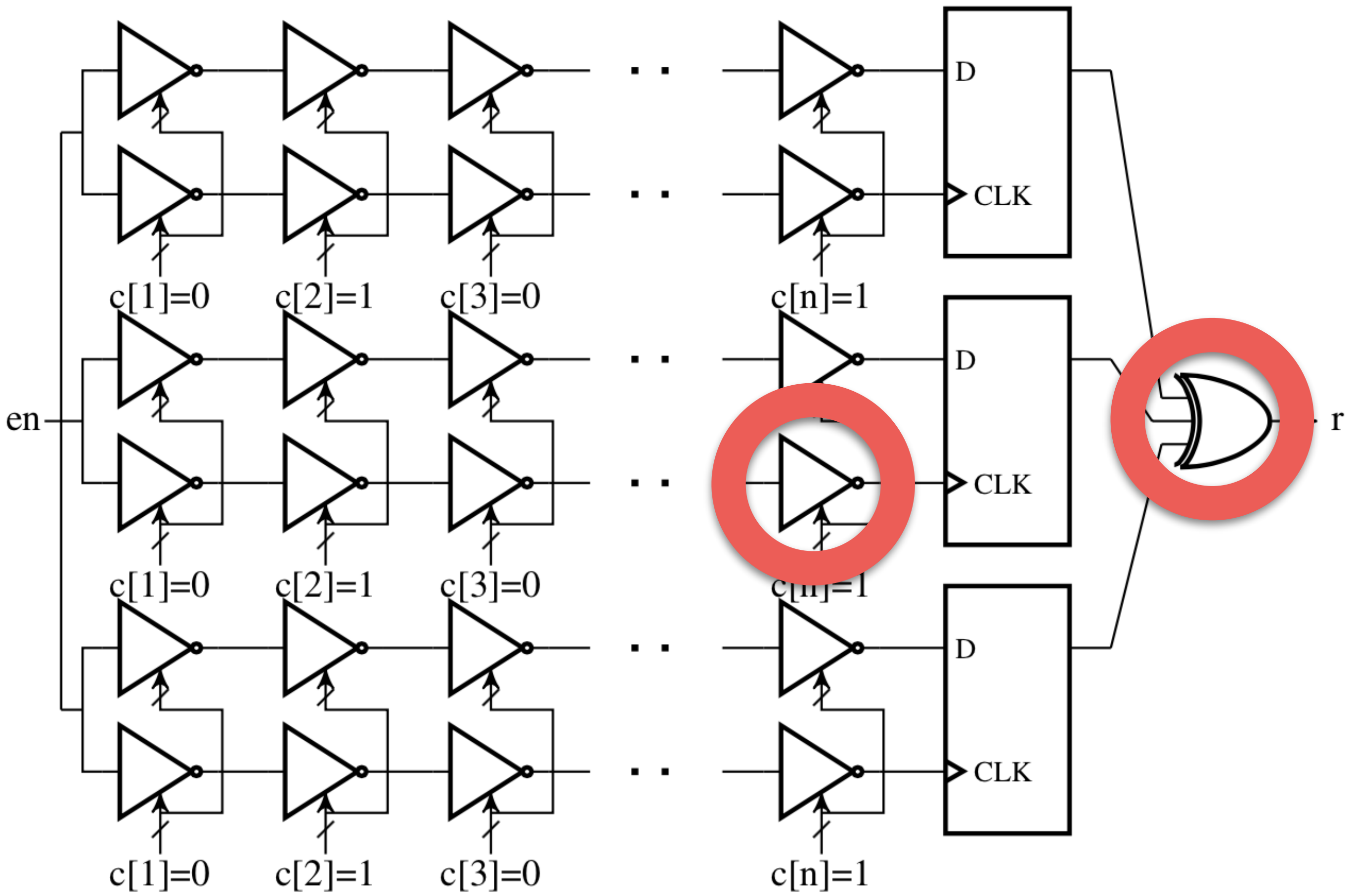
- n input LUT >> $2^{(n)}$ SRAM cells >> $2^{(2^n)}$ configurations

- Any Faulty SRAM cell in the LUT change the logical combinatorial function

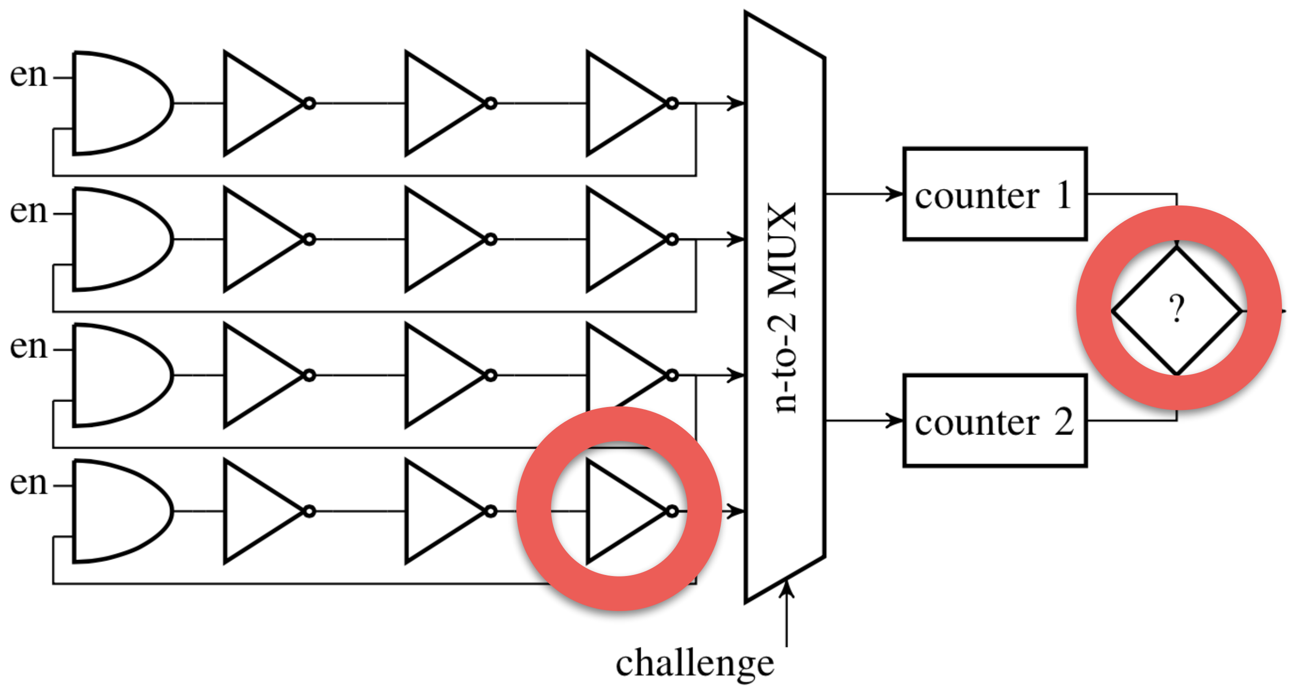


Possible Targets

XOR arbiter PUF

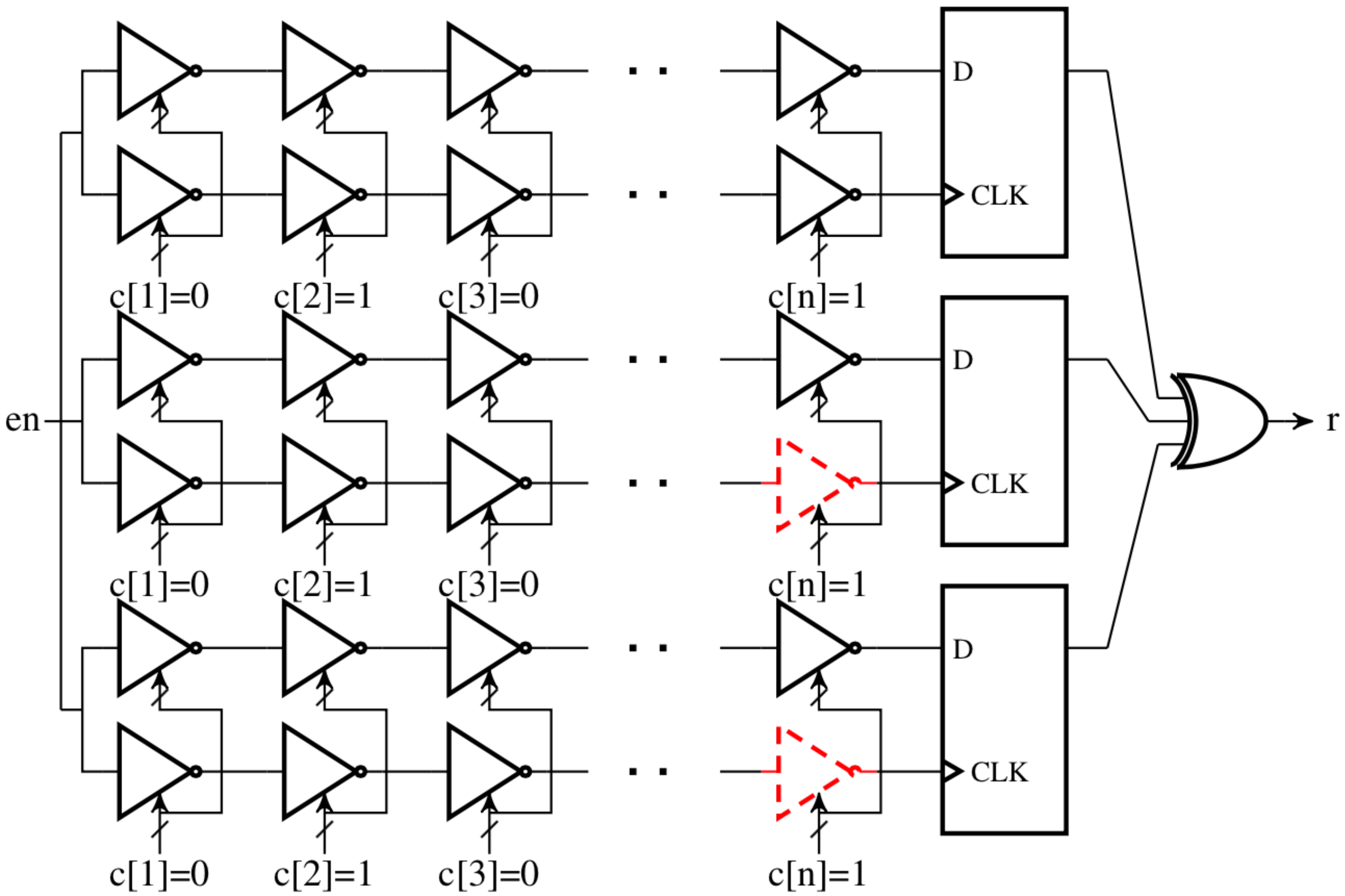


RO PUF

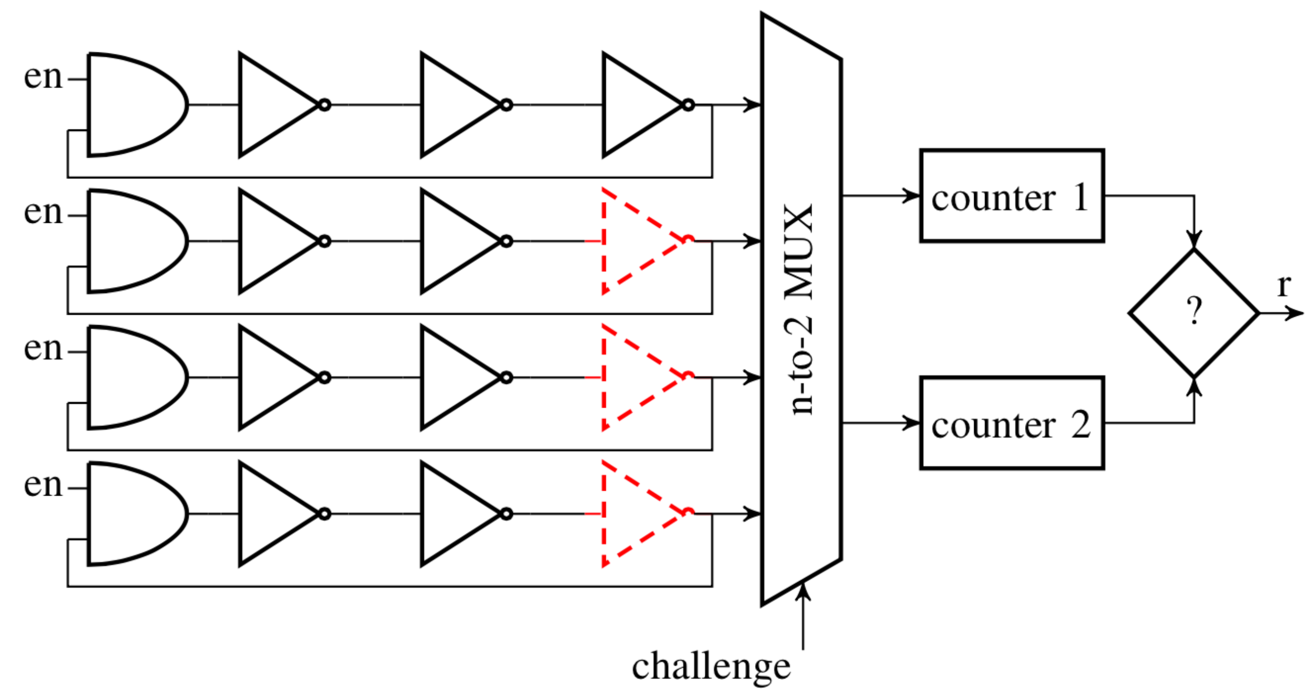


Inverters as Easiest Targets

XOR arbiter PUF

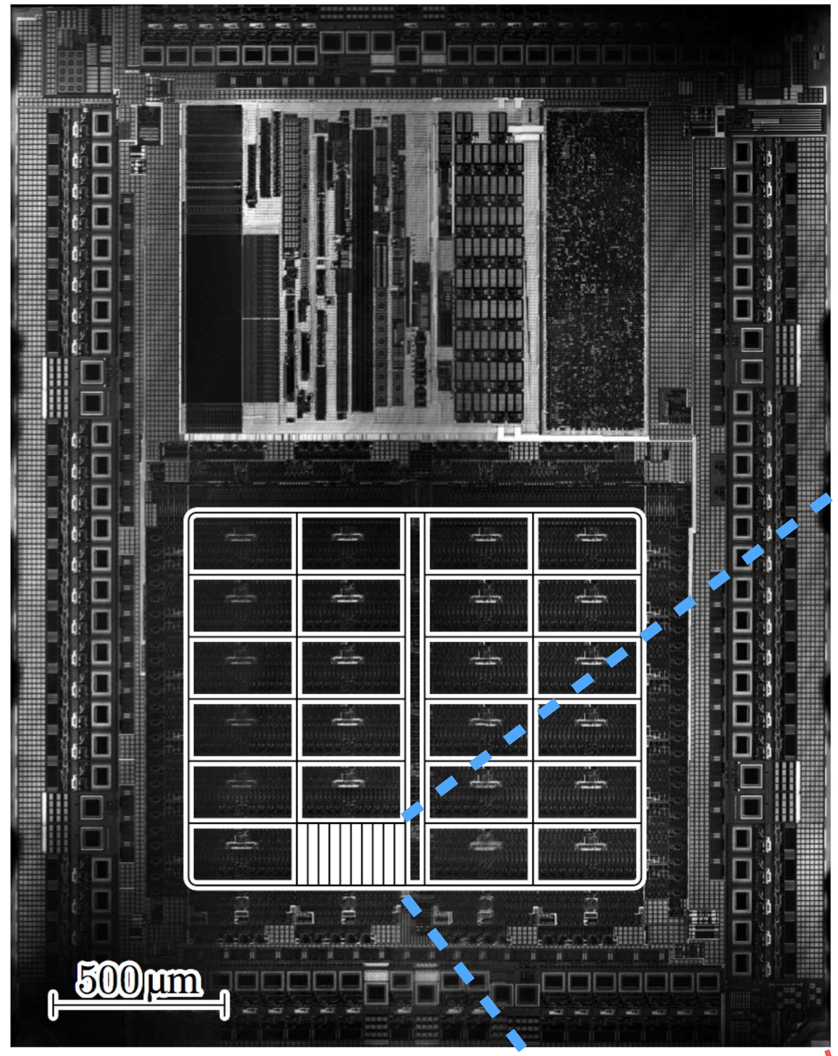


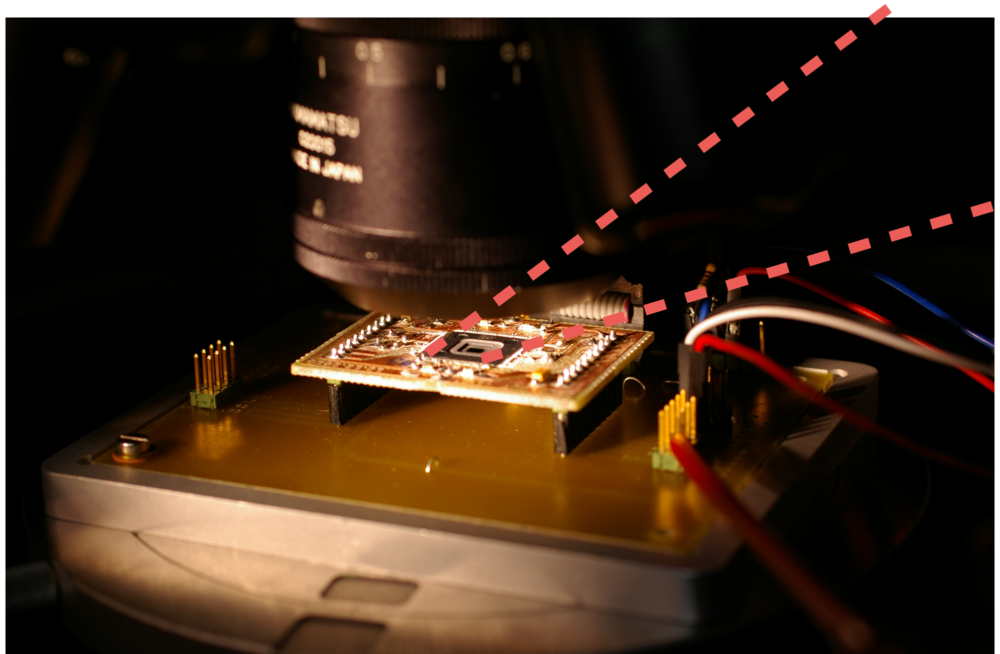
RO PUF

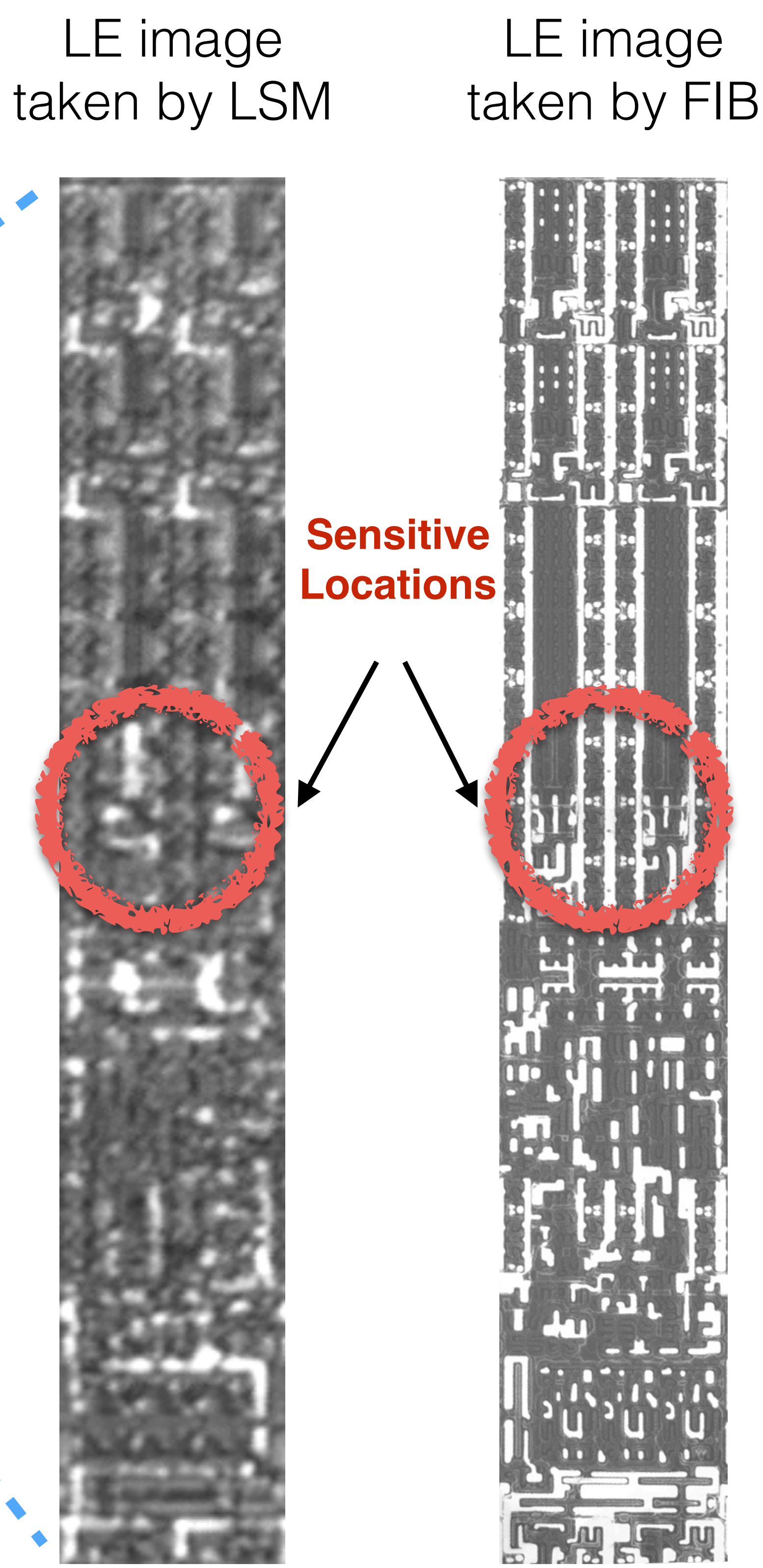


DUT:
Altera MAX V (180 nm)

Optical Setup:
HAMAMATSU PHEMOS

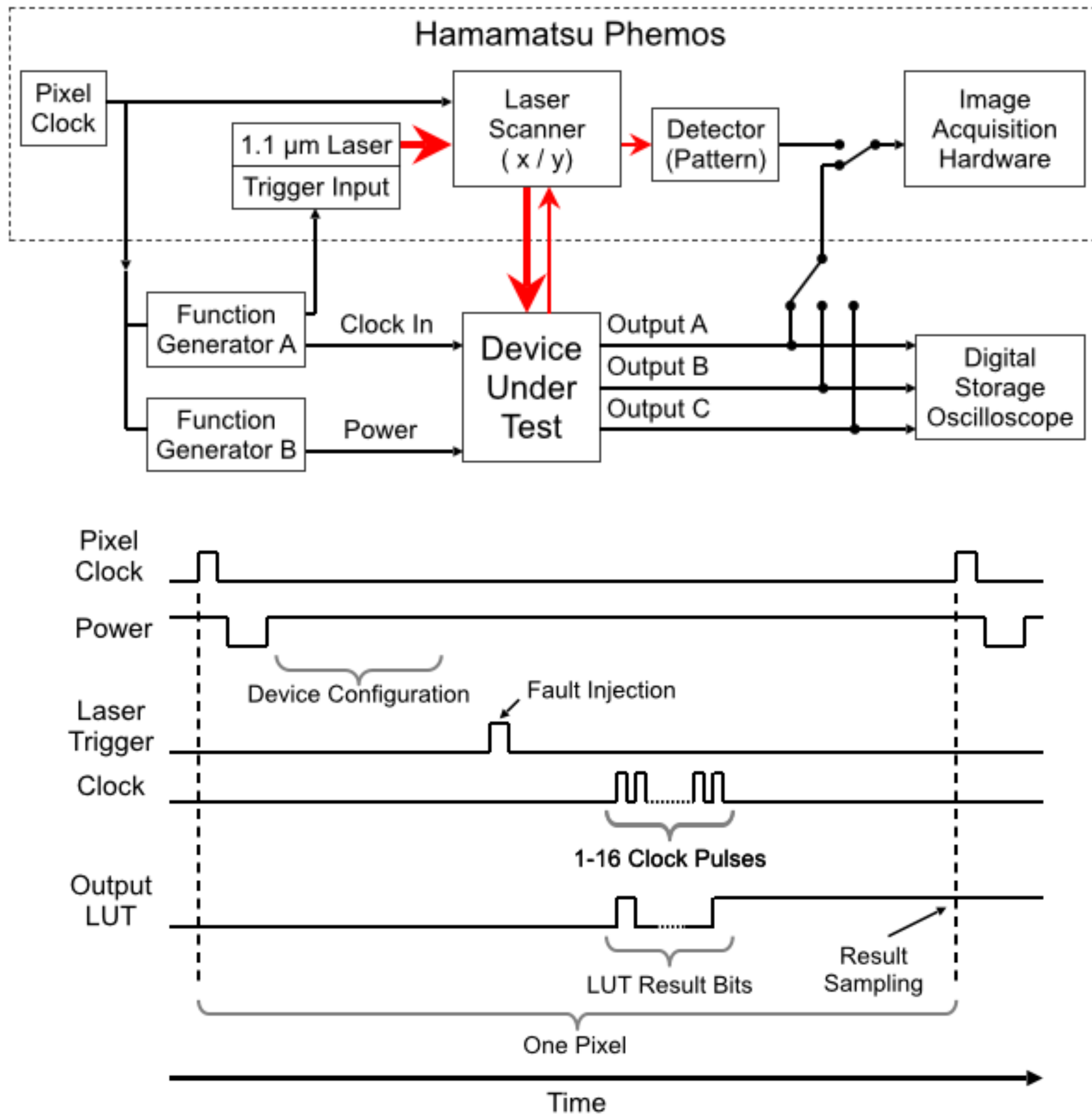




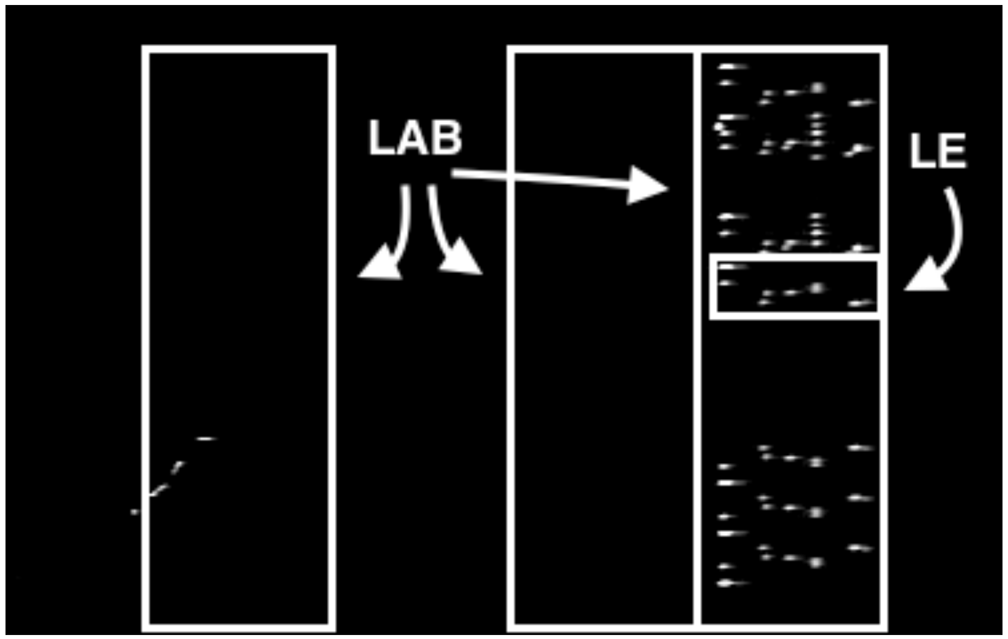


Experimental Setup

- Finding the sensitive locations by scanning the whole LE with the laser scanning microscope (LSM)
- Addressing all SRAMs of a LUT after the laser shot to observe the faults

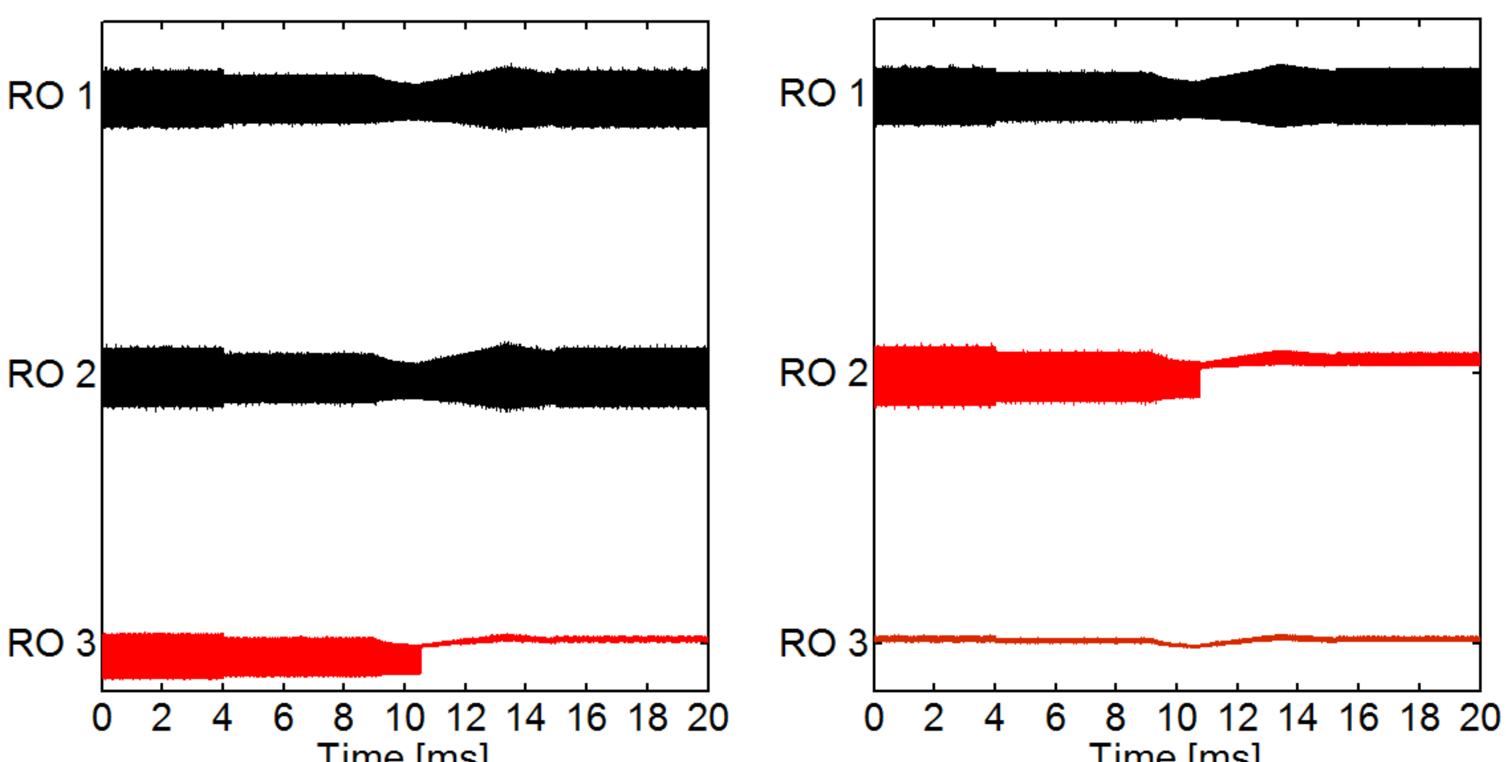


Results

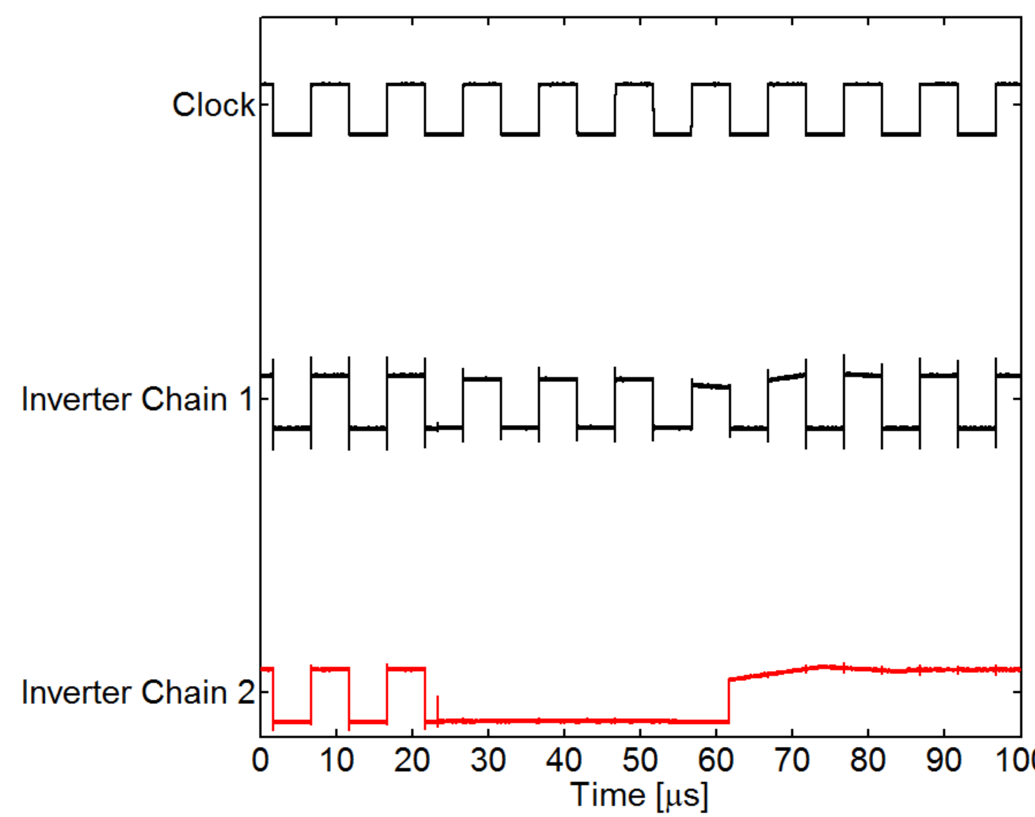


← Finding PUFs by photonic emission analysis

RO PUF with 3 oscillators

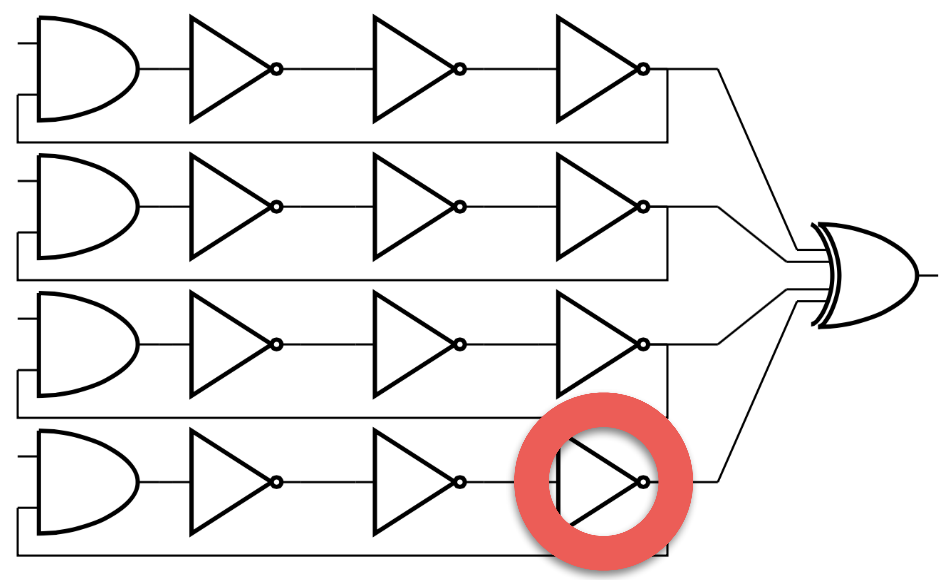


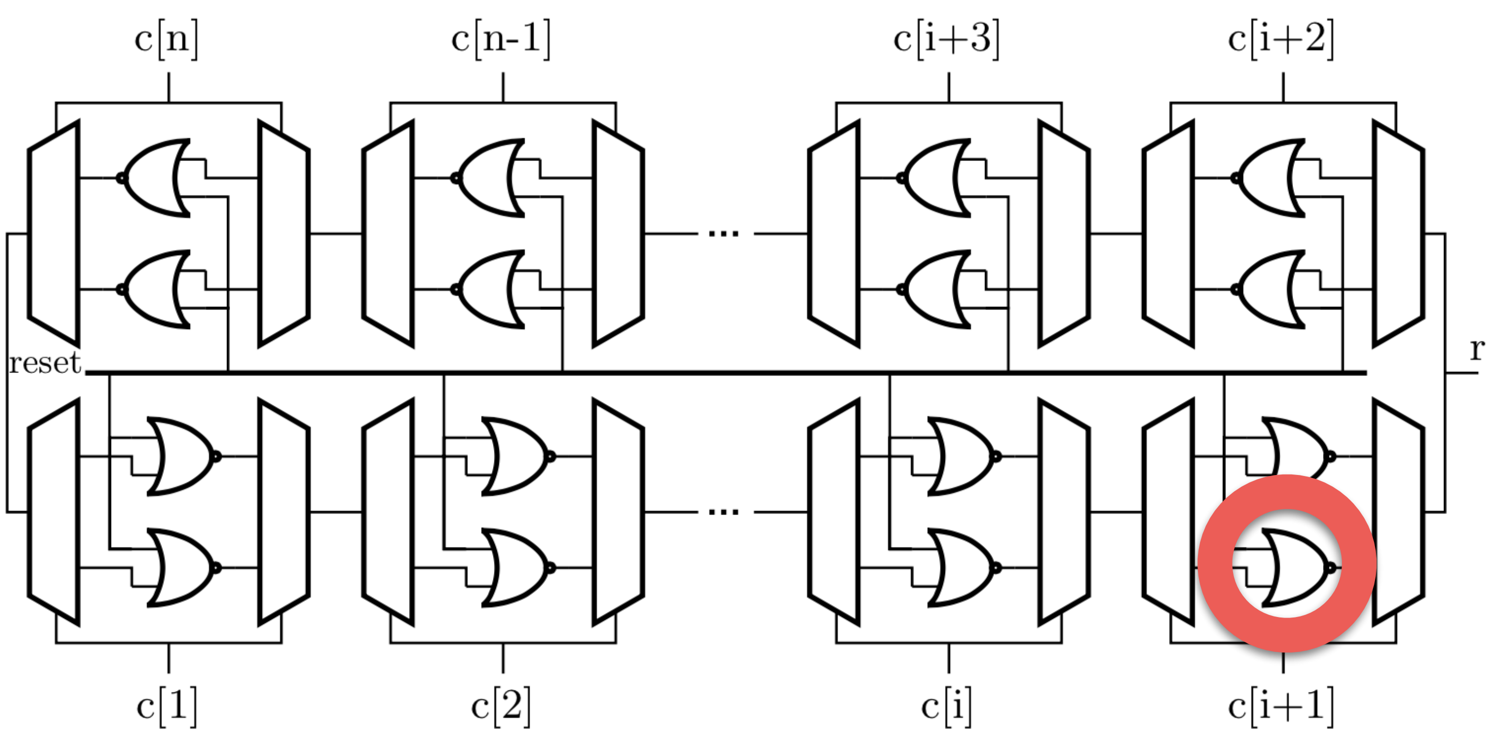
XOR arbiter PUF with 2 arbiter chains



Same Attack on other Intrinsic Primitives

Delay-based TRNG →



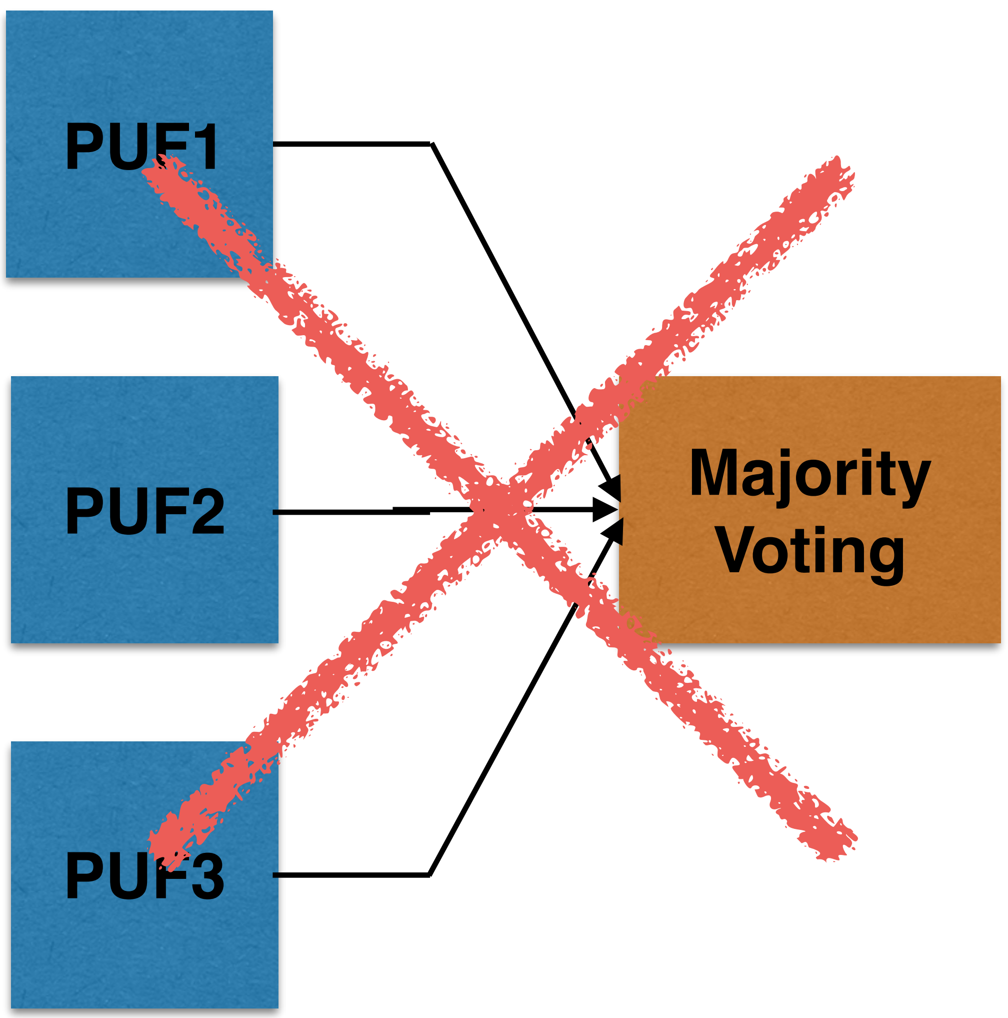


← BR PUF

Classical Countermeasures?

- Protecting arithmetic operations using redundancy: e.g., Triple Modular Redundancy (TMR), Duplication with Comparison (DWC)

- **Duplication (i.e., physical cloning) of one PUF instance is nearly impossible**



Conclusion

- Reducing the complexity of the PUFs in authentication and key generation applications using Laser Fault injection:
  - Learning XOR PUFs in polynomial time
  - Entropy reduction of PUF responses
- Launching the same attack on other platforms, such as ASICs by deactivating the registers
- Launching the same attack on similar intrinsic primitives such as TRNGs and BR PUFs
- Classical countermeasures cannot be effective for PUFs

Thanks for your Attention!

Questions?