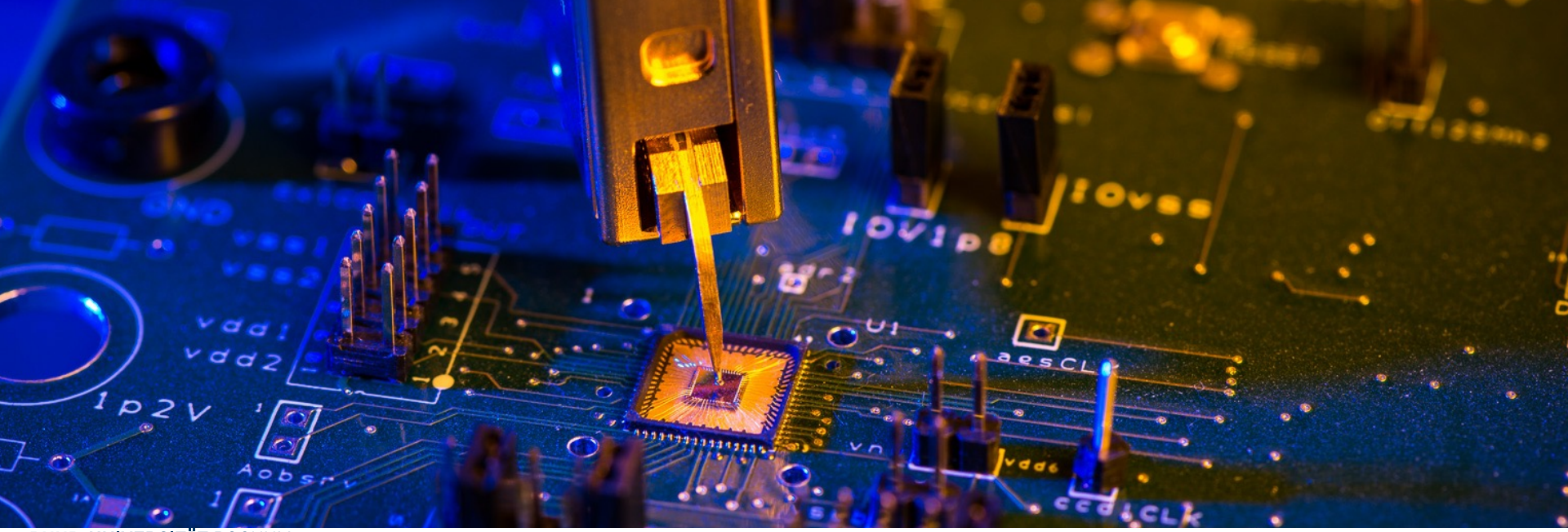


RUHR-UNIVERSITÄT BOCHUM

Fault Injections in System Design and Verification – Challenges and Outlook

Fault Diagnosis and Tolerance in Cryptography (FDTC)

Tim Güneysu
14. September 2025

Chair for Security Engineering
Faculty of Computer Science
Ruhr University Bochum

SECURITY ENGINEERING

Motivational Quote

„Research on reliable fault-tolerance systems is not required anymore -- the fault-tolerant community on security will solve their problems anyway.“

D.G.

Credits

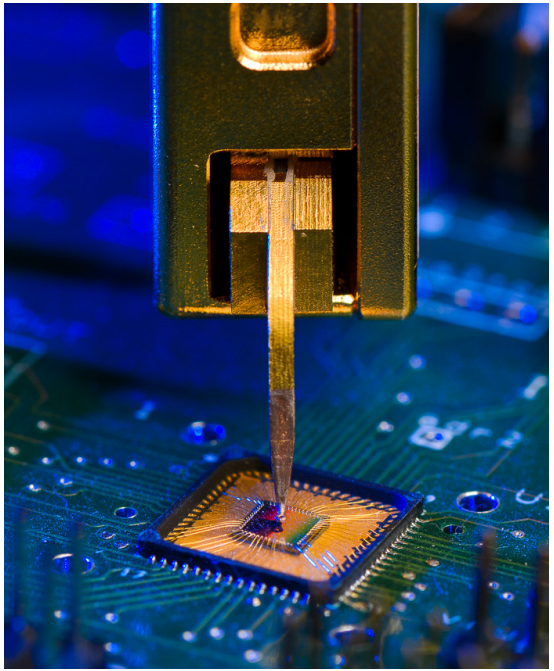
This talk is based on joint work with most of my team

Slide credit:

- Jan Richter-Brockmann
- Jakob Feldtkeller

Outline of this talk

- Motivation
- Some Observations
- Fault Injection Attacks Revisited
- Modelling Fault Injection
- Directions & Challenges
- Conclusions



Fault Injection Attacks – Tracing the Journey to Today

1996 **Seminal attack by Boneh, DeMillo and Lipton**
Breaking RSA with single fault (*aka* Bellcore attack)

1997 **Biham and Shamir**
Differential Fault Analysis (DFA, 1997)

until 2025 **according to ChatGPT (“Between 2009-2021 at least 50 targeted studies” 🤔)**
according to Google Scholar

→ Exact search on “Fault injection” – 55,800 hits
→ Search on “Fault injection” – 985,000 hits
→ Search on “Fault injection attacks” – 101,000 hits
→ Search on “Fault injection countermeasure” – 24,000 hits

Given the plethora of works, can we assume FIA in cryptography as (somewhat) tackled?

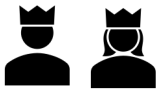
Observation A – Fault Injection Analysis (FIA) in Academic Settings

Cryptanalyst **C**lara‘s (*) idea:
Let’s amplify a cryptanalytic method via FIA
- → identifies requirements needed for improvements
- → assumes FIA can (magically) do the job
- → develops idea and submits paper

Reviewer **R**obert (*aka reviewer **B**=**B**ob*) receives paper for review
- → questions assumptions on FIA
- → requires practical experiments to demonstrate feasibility of FIA assumptions

Practitioner **P**aul to the rescue:
accepts to help with FIA demonstration
- → builds attacks setup
- → runs experiments
- → reports results to **C**lara

SUCCESS



→ finally accepted paper
→ fruitful collaboration

FAILURE

→ months of wasted research time

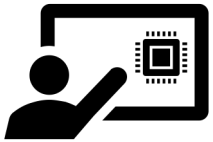
(*) All names are based on fictional characters.

Question: What are the **odds** to be successful? How to *increase* success probability *a priori* for Clara?

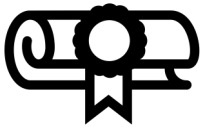
Observation B: Fault Injection Protection in Industrial Practice

Manager **M**artin requests FIA protection
- → competitor has certified FIA-secure product already in market
- → should be better than competitor; meet certification criteria
- → at best at no cost; with no delay for time t to market

Engineer **E**rin receives request
- → with no extra time, little flexibility to explore new paths
- → decides to throw TMR + lockstepping in
- → performs in-house experiments to pre-test protection
- → counts points required for certification process



Evaluation Lab **Val**ery validates product
- → runs large set of experiments (again) using own (different?) setup
- → reports (differing) results and requests changes

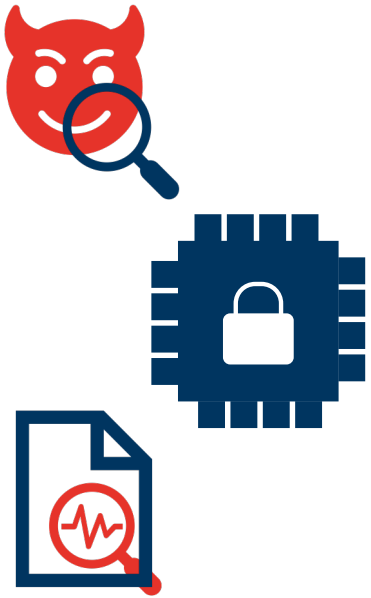


Awarded certificate

Questions: - Is there nothing better than TMR + lockstepping for quick implementation & certification?
- Are there universal, reproducible and verifiable proofs on FIA protections to simplfy Valery's work?

Observation Summary

What are key observations and research questions?



RQ1: Can we ***universally and reliably determine success probabilities*** of fault events on specific/structurally identical/similar hardware?

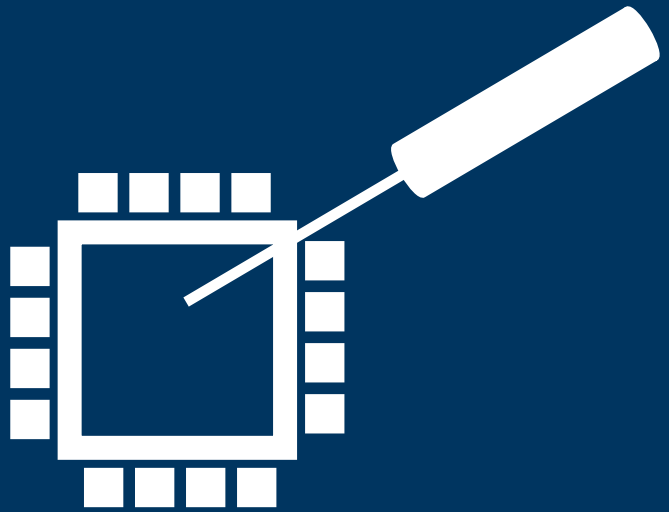
RQ2: How can we efficiently identify and model critical components (gates, wires, memories) affected by ***any type of fault event*****?**

RQ3: What are ***better generic design principles*** (beyond duplication) to eliminate the impact of fault events?

RQ4: How can we ***efficiently validate the correct operation and fault coverage*** of countermeasures?

Disclaimer: There are a number of additional relevant questions, but we unfortunately only have one day of FDTC…

FAULT INJECTION REVISTED



Fault Injection Revisited

RUB

Fault injection

Fault setup

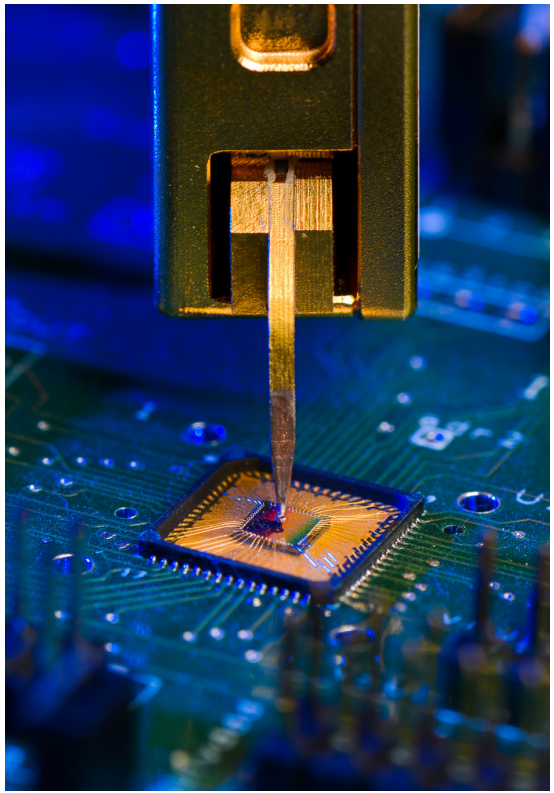
Fault target & technology

Fault characteristics

Fault types

Fault distinguishers

Fault combinations

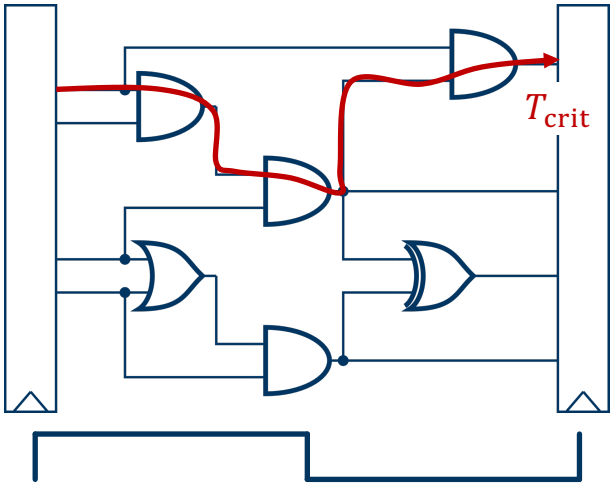


Different Paths for Injection

- Clock Glitching
- Voltage Glitching
- Electromagnetic Pulses
- Laser/Photon Injection
- Temperature Variation

Fault Injection Revisited [RBSG21]

Clock Glitches

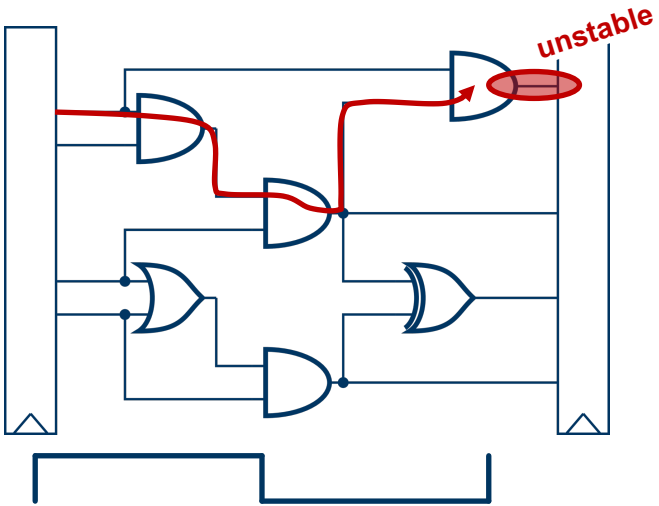


$$T_{\text{clk}} \geq T_{\text{crit}} + t_{\text{clkq}} + t_{\text{setup}} - \delta$$

[RBSG21] Jan Richter-Brockmann, Pascal Sasdrich, Tim Güneysu:
Revisiting Fault Adversary Models - Hardware Faults in Theory and Practice. IEEE Trans. Computers 72(2): 572-585 (2023)

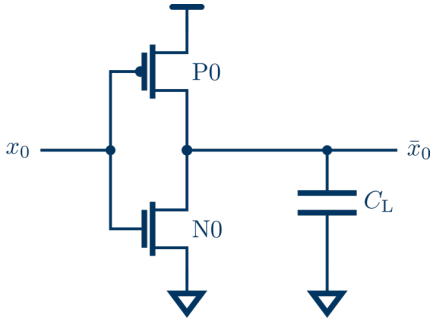
Fault Injection Revisited [RBSG21]

Clock Glitches



$$T'_{\text{clk}} < T_{\text{crit}} + t_{\text{clkq}} + t_{\text{setup}} - \delta$$

Underpowering and Voltage Glitches

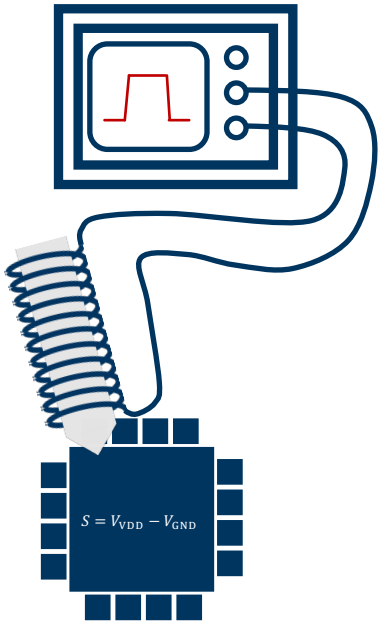


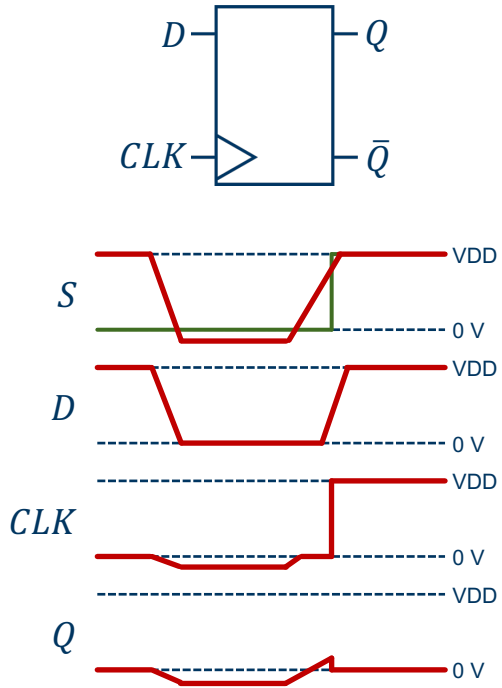
$$t_{pLH} = \frac{C_L \cdot \left[\frac{2|V_{TH2}|}{|V_{DD}| - |V_{TH2}|} + \ln\left(3 - \frac{4|V_{TH2}|}{V_{DD}}\right) \right]}{\mu_p C_{OX} \left(\frac{W}{L}\right)_2 (V_{DD} - |V_{TH2}|)}$$

[RBSG21] Jan Richter-Brockmann, Pascal Sasdrich, Tim Güneysu:
Revisiting Fault Adversary Models - Hardware Faults in Theory and Practice. IEEE Trans. Computers 72(2): 572-585 (2023)

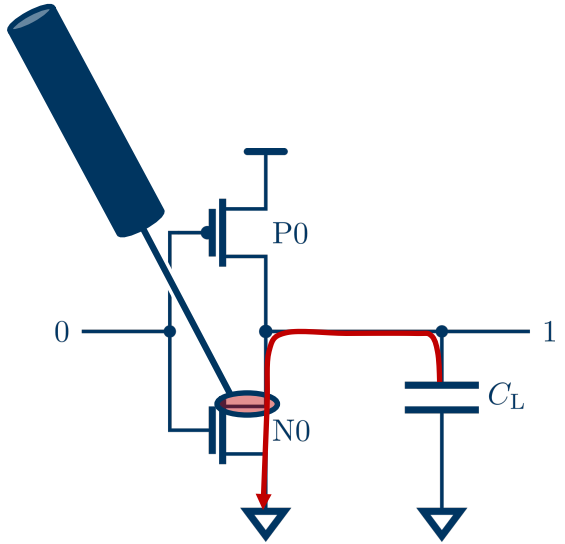
Fault Injection Revisited [RBSG21]

Electromagnetic Pulses



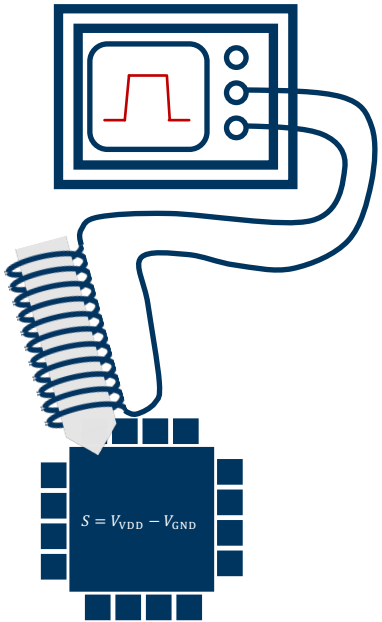


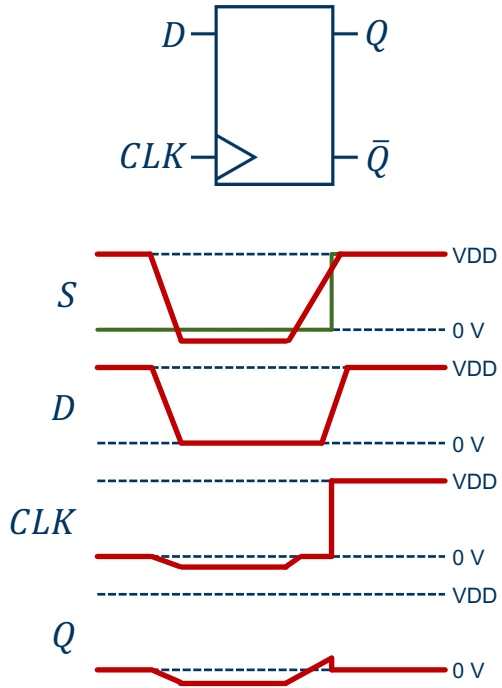
Laser Fault Injection



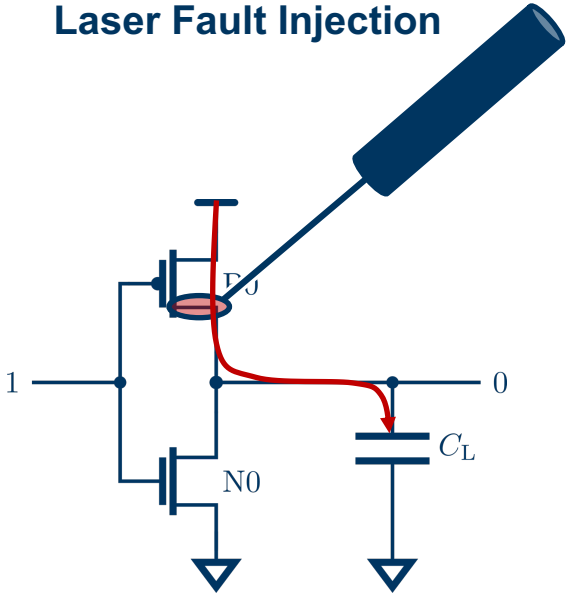
Fault Injection Revisited [RBSG21]

Electromagnetic Pulses



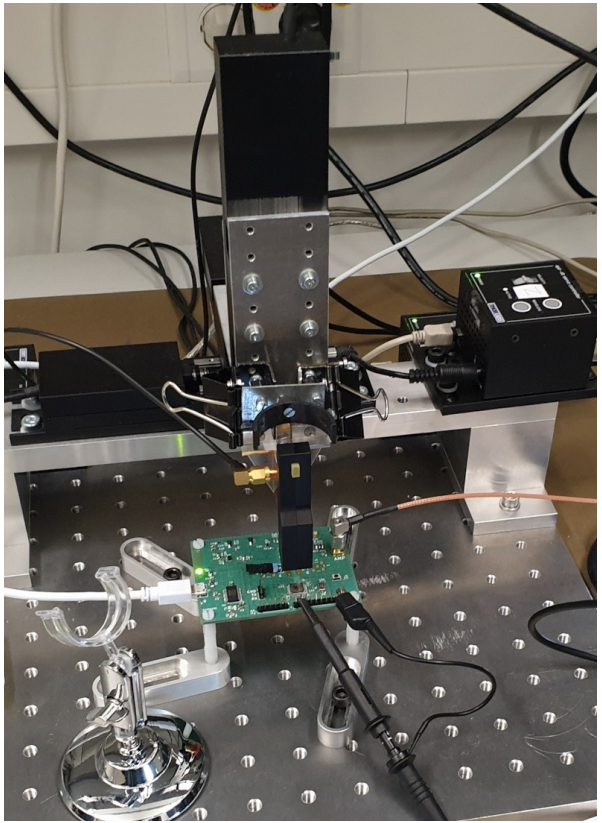


Laser Fault Injection



+ additional (less relevant) mechnisms such as temperature, x-ray beams, body bias…

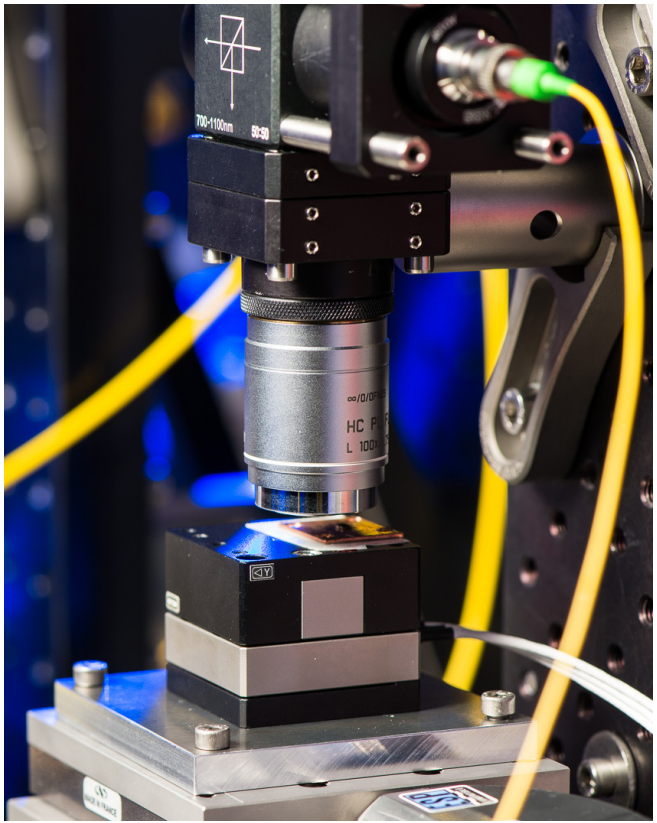
Fault Injection Revisited



Relevant Components in Fault Injection Setups

- Injection device (e.g., laser type & size, EM probe)
- Power supply (DUT/setup)
- Function generator (control of timings)
- Environmental influences (room situation, human noise)

Fault Injection Revisited



Targets

- Choice of input parameters (public vs. private), e.g. [KP+24]
- Processor vs. hardware circuits
- Logic style (CMOS, WDDL, ...)
- Technology feature size (1.2µm vs. 3nm)

[KP+24] E. Krahmer, P. Pessl, G. Land, T. Güneysu: Correction Fault Attacks on Randomized CRYSTALS-Dilithium. CHES 2024.

Fault Injection Revisited – Evolution of Laser Attacks

RUB

Skorobogatov

1200 nm

$\sim 20\ \mu m \times 20\ \mu m$ (6T)

$\sim 10\ \mu m$

Roscian et al.

250 nm

$9\ \mu m \times 4\ \mu m$ (5T)

$\sim 1\ \mu m$

Courbon et al.

90 nm

$3\ \mu m \times 1.5\ \mu m$ (6T)

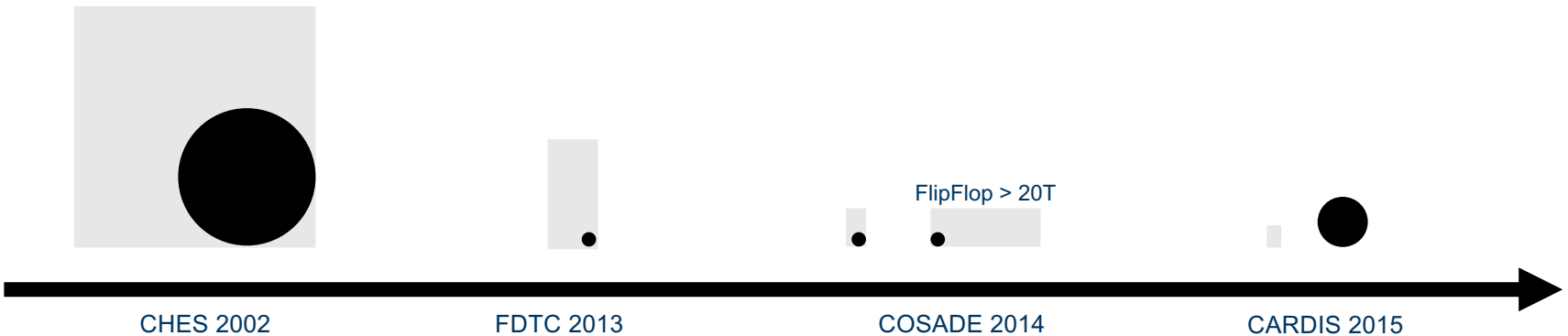
$\sim 1\ \mu m$

Selmke et al.

45 nm

$\sim 1.62\ \mu m \times 1\ \mu m$ (5T)

$\sim 4\ \mu m$



[SBHS15] Selmke, et al. "Precise Laser Fault Injections into 90 nm and 45 nm SRAM-Cells." International Conference on Smart Card Research and Advanced Applications (CARDIS), 2015.

[CLFT14] Courbon, et al. "Adjusting laser injections for fully controlled faults." International workshop on constructive side-channel analysis and secure design (COSADE), 2014.

[RSDT13] Roscian, et al. "Fault Model Analysis of Laser-Induced Faults in SRAM Memory Cells." *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2013*.

[SA02] Skorobogatov and Anderson, Optical Fault Induction Attacks. In: Kaliski, B.S., Koç, ç.K., Paar, C. (eds) Cryptographic Hardware and Embedded Systems (CHES), 2002.

Fault Injection Revisited

RUB

Fault injection

Fault setup

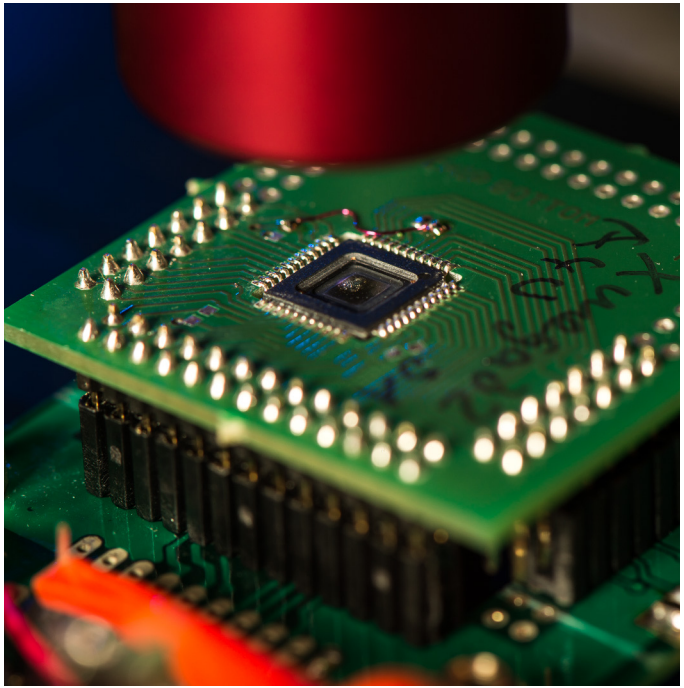
Fault target & technology

Fault characteristics

Fault types

Fault combinations

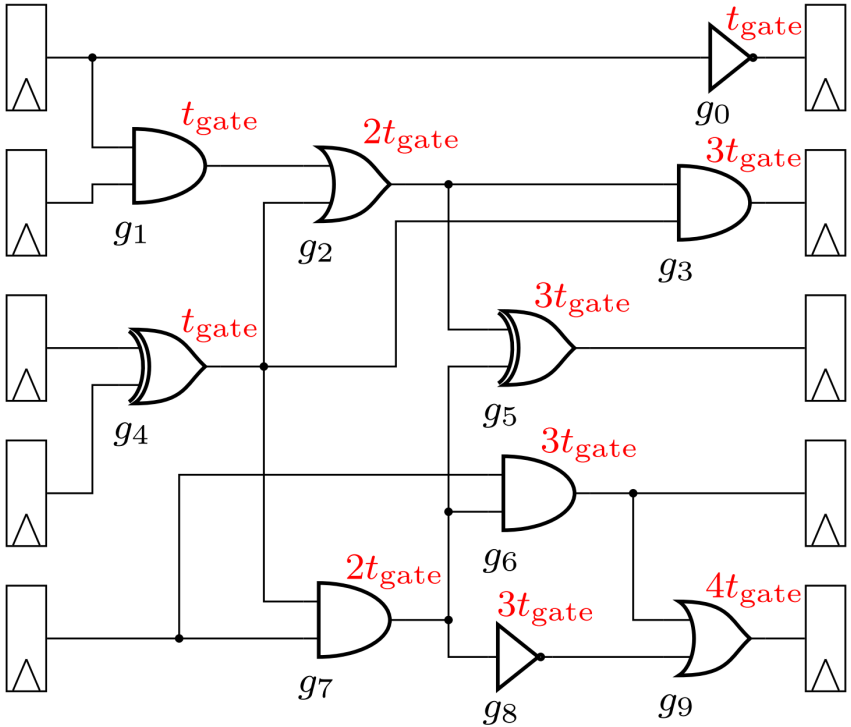
Fault distinguishers



Fault characteristics

- Range of impact
- Intensity
- Duration
- Timing
- Power domain/clock domain
- Location (EM, laser)

Fault Injection Revisited – Location Parameter



$$\mathcal{P} = \{t_o, t_1, \ldots, t_{T-1}\}$$

where $t_0 > t_1 > \cdots > t_{T-1}$ and $T \leq |\mathcal{G}_{\text{regin}}|$

$$\mathcal{G}_{\text{cluster},i} = \{g \in \mathcal{G}_{\text{regin}} \mid t(g) \geq t_i, t_i \in \mathcal{P}\}$$

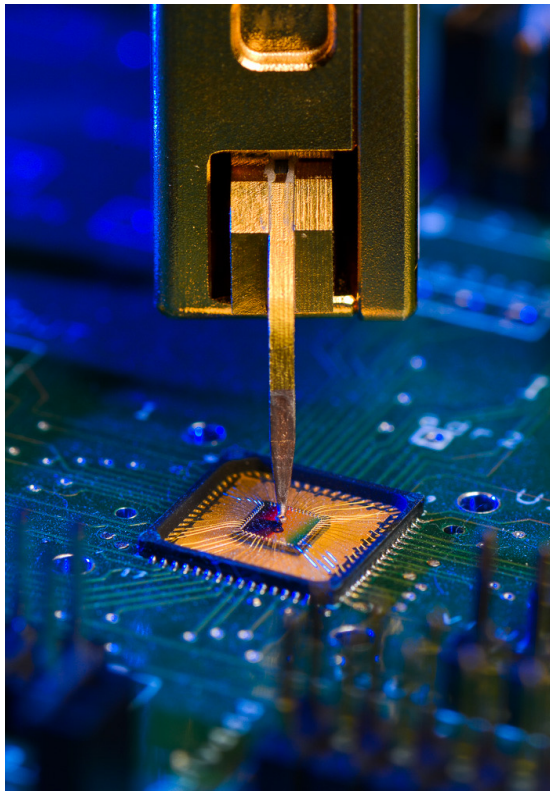
$$\mathcal{G}_{\text{cluster},0} = \{g_9\}$$

$$\mathcal{G}_{\text{cluster},1} = \{g_3, g_5, g_6\} \cup \{g_9\}$$

$$\mathcal{G}_{\text{cluster},2} = \{g_0\} \cup \{g_3, g_5, g_6\} \cup \{g_9\}$$

$$c_\infty = \{g_0, g_1, g_2, g_3, g_4, g_5, g_6, g_7, g_8, g_9\}$$

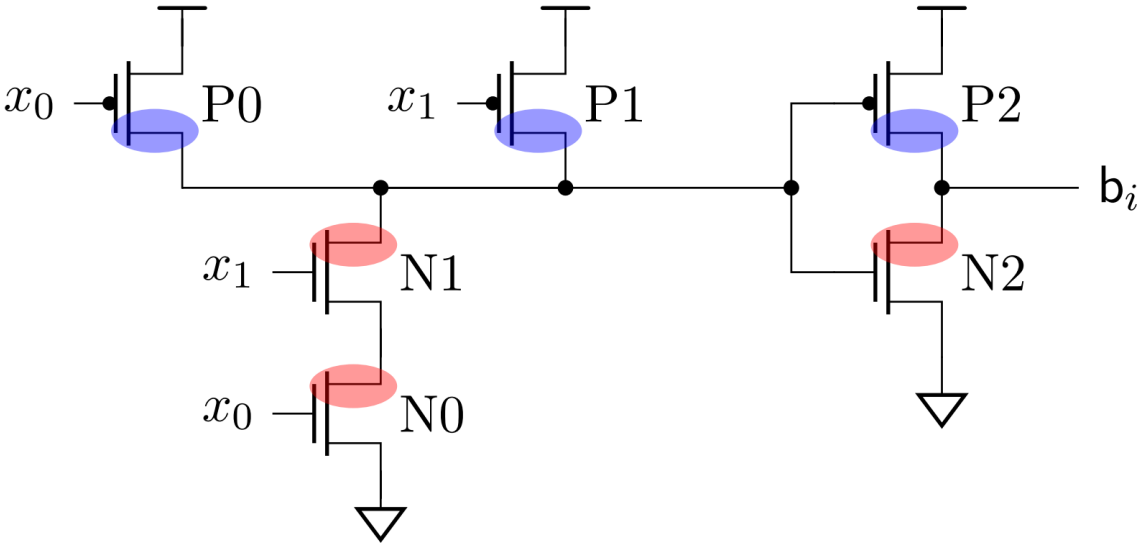
Fault Injection Revisited



Fault types

- Permanent vs. Transient
- Stuck-at-0 or bit reset (0), stuck-at-1 or bit-set (1), toggle or bit flip (0→1;1→0)
- Impacted components
  - Flip-flops (data corruption)
  - Gates (behavioral swap)
  - SRAM (data corruption)
  - DRAM (row hammering)

Fault Injection Revisited: Target CMOS Gates – AND



$$\{and\} \mapsto \{or, set, reset\}$$

Fault Injection Revisited

RUB

Fault injection

Fault setup

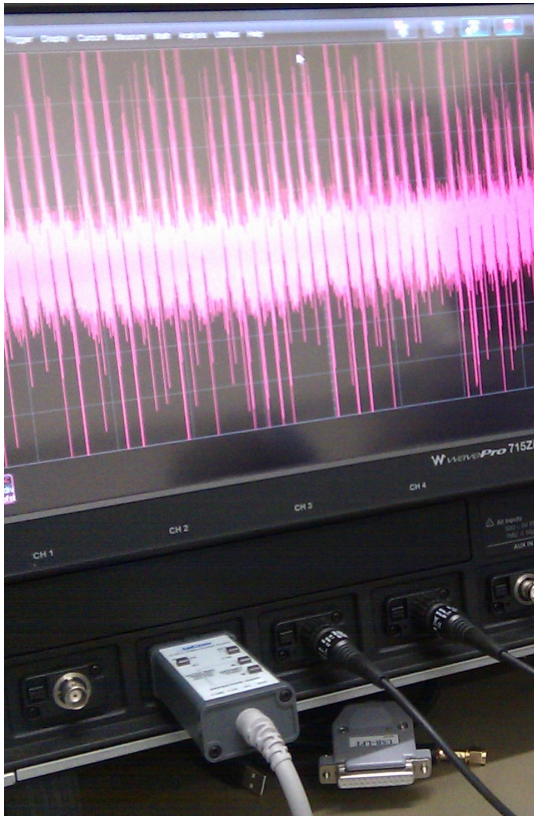
Fault target & technology

Fault characteristics

Fault types

Fault distinguishers

Fault combinations



Success criteria & distinguishers

- Differential Fault Analysis (DFA, 1997)
- Safe-Error Attacks (SEA, 2003)
- Fault Sensitivity Analysis (FSA, 2010)
- Algebraic Fault Analysis (AFA, 2012)
- Statistical Fault Analysis (SFA, 2013)
- Differential Fault Intensity Analysis (DFIA, 2014)
- Statistical Ineffective Fault Analysis (SIFA, 2018)
- Persistent Fault Attacks (PFA, 2018)
- Fault Template Attacks (FTA, 2020)

…

Fault Injection Revisited

RUB

Fault injection

Fault setup

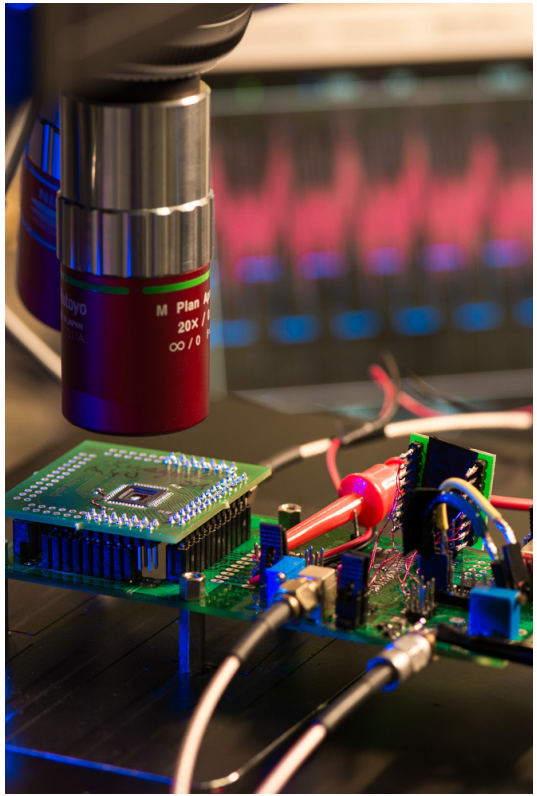
Fault target & technology

Fault characteristics

Fault types

Fault distinguishers

Fault combinations



Fault combinations

- Single fault vs. multiple fault injection
- Single bit random faults vs. burst or full row/column faults
- **Exhaustive simulation/verification** of all possible fault combinations in larger circuits is exponential (impossible)
- **Combined security** against multiple implementation attack vectors taking place **at the same time**
  - Reverse Engineering + FIA
  - **Side-Channel Analysis + FIA**

FAULT MODELLING

for Cryptographic Design and Verification

Design, Integration & Validation of FIA Countermeasures

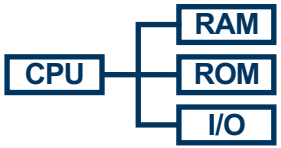
RUB

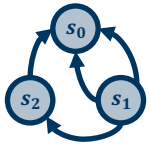
Countermeasure design & integration

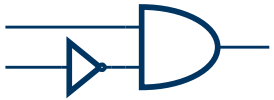
Fault Modelling

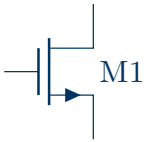
DESIGN PHASE

Abstraction









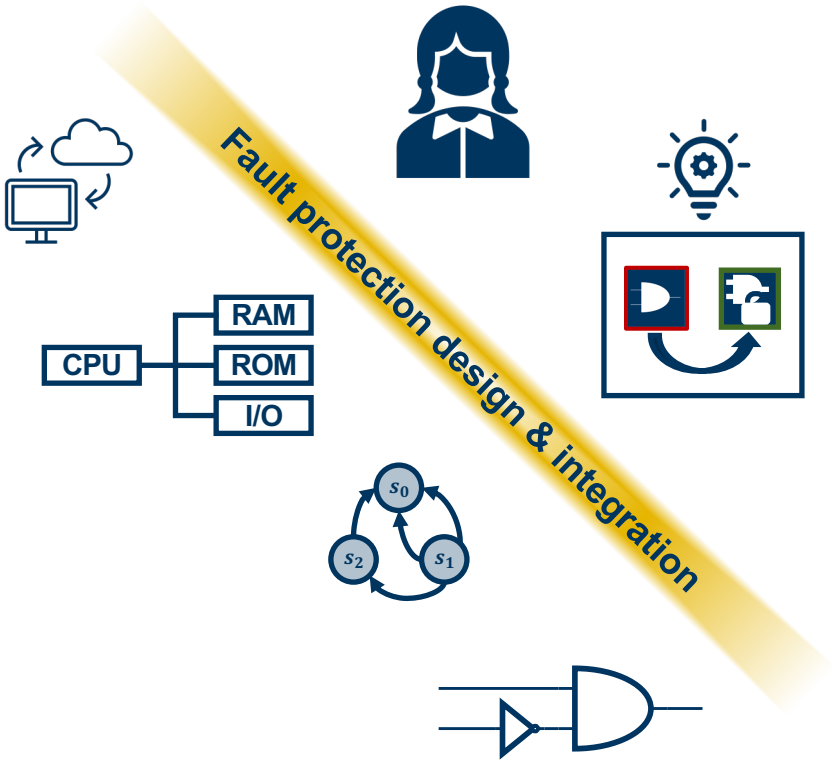
System | Algorithm | Register Transfer | Gate | Circuit | Material

Low-Level Detail

TEST PHASE

Practical Fault Injection & Validation

Design, Integration and Validation of Fault Injection Protections



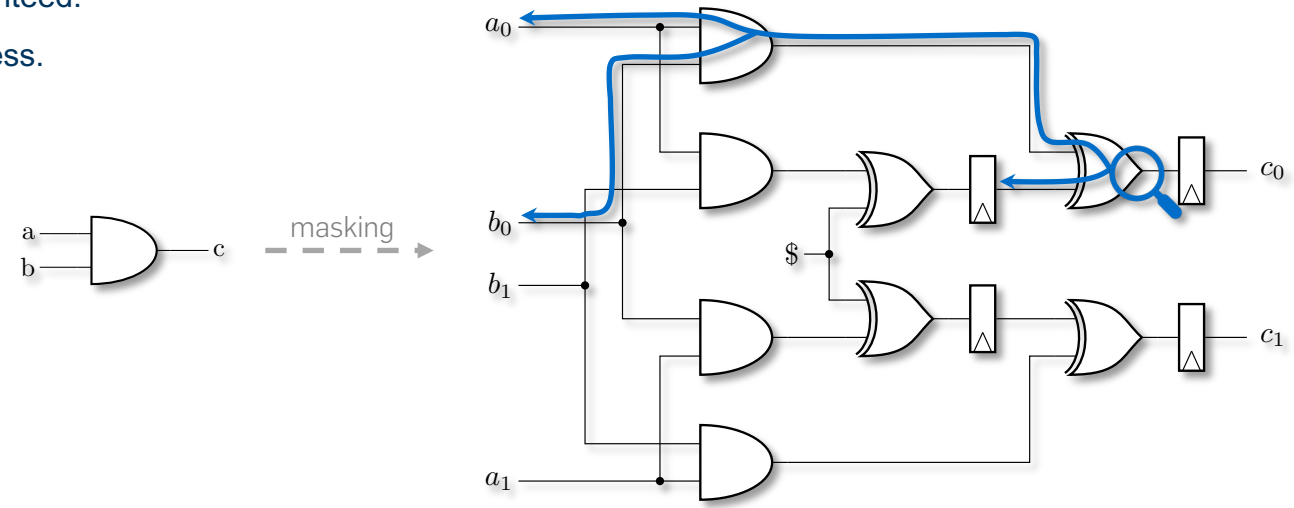
- **Most (security-related) fault injection countermeasures require manual human interaction**
  - Locations selected by criticality analysis
  - Protections chosen by coverage assumption
- **Computer-aided tools can** support the designer in an early stage of the design process
  - Automated fault protection instantiations defined by fault model and assumptions
  - Pre-silicon evaluation of target countermeasures
  - Proof of correct operation
- This requires to **model generic fault analysis and associated requirements**
- **How can we achieve this efficiently?**

Recall: Boolean Masking

Secret bit x is replaced by a vector of bits $\langle x_0, x_1, \ldots, x_{d-1}, x_d \rangle$, such that each true subset is independent of x but $x = \sum x_i$.

Circuits are transformed to compute over vectors such that probing security is guaranteed.

- Requires fresh randomness.

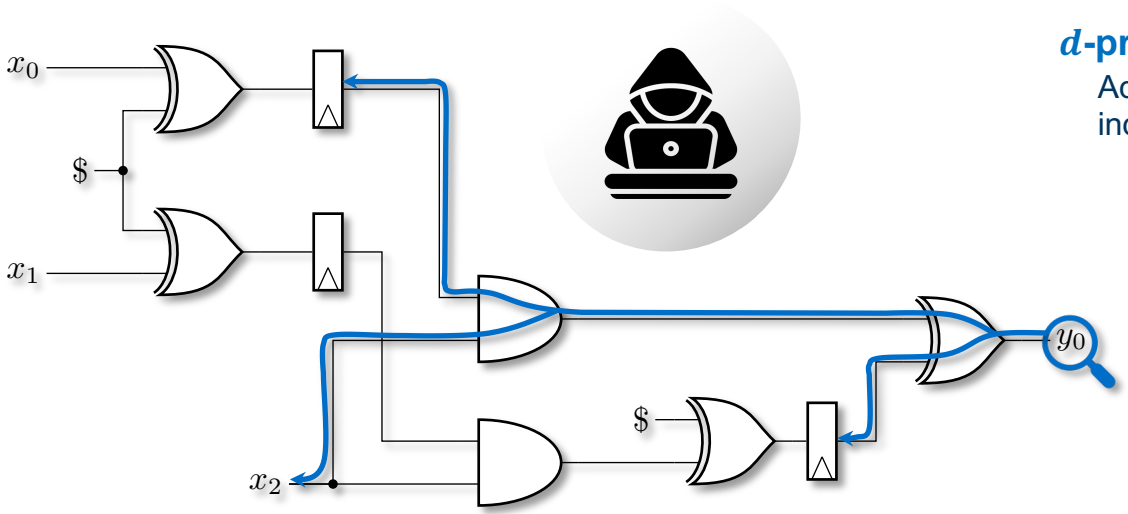


[Chari et al., Towards Sound Approaches to Counteract Power-Analysis Attacks, CRYPTO 1999]

Threshold Probing Model

Glitch-Extended

Free placement of up to d **probes** on wires, which leak the value of the last synchronization points.



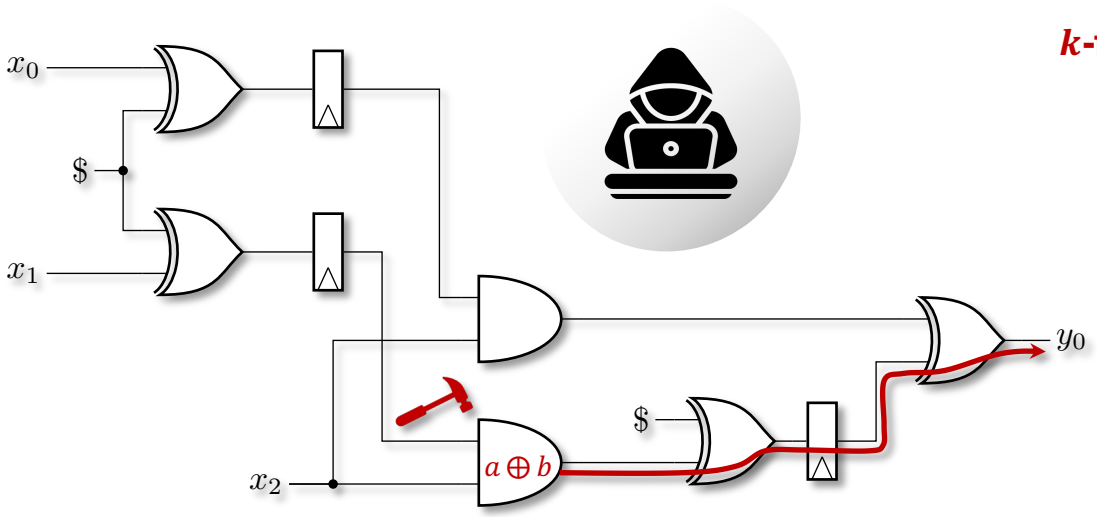
d-probing security

Adversarial observation is statistically independent of any secret input.

[Ishai et al., Private Circuits: Securing Hardware against Probing Attacks, CRYPTO, 2003]
[Faust et al., Composable Masking Schemes in the Presence of Physical Defaults & the Robust Probing Model, CHES, 2018]

Threshold Fault Model

Free selection of up to ***k* gates** which are **manipulated** according a chosen fault transformation (changing the underlying function of the gate).



***k*-fault security**

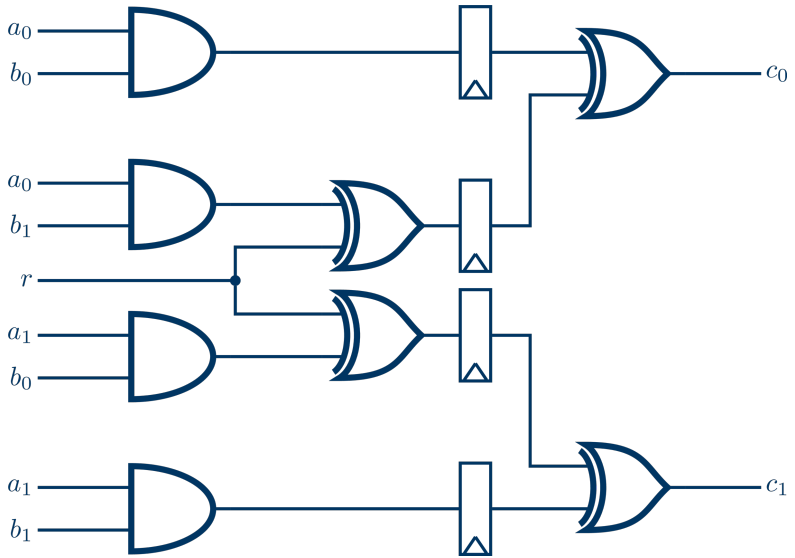
Faulty behavior can be detected or corrected at the circuit output.

This looks similar to a well-established model for side-channel security…

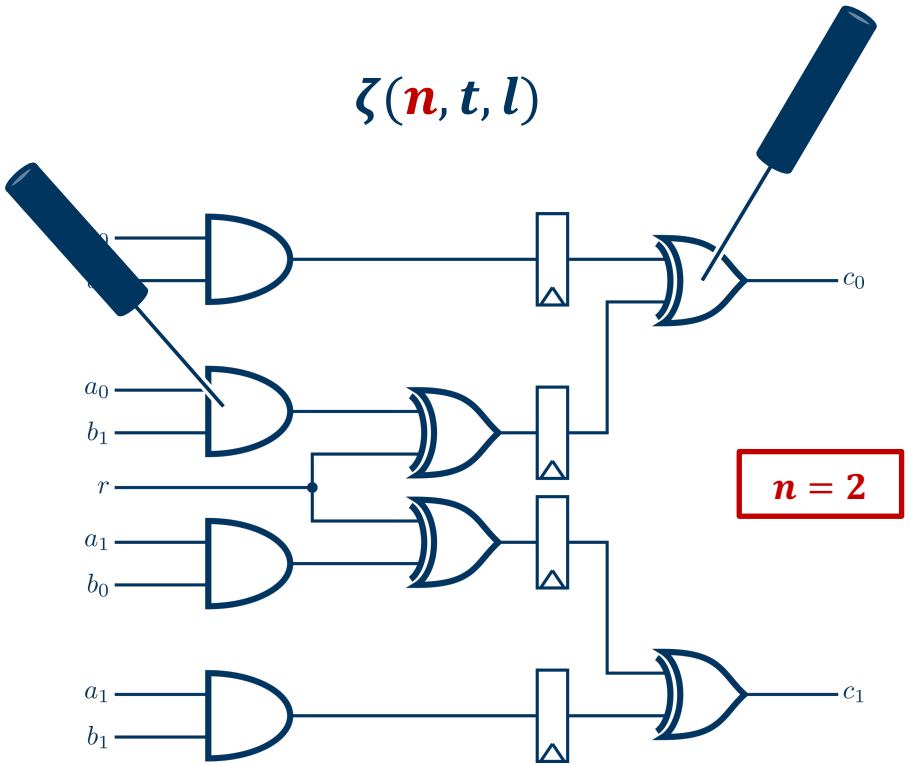
[Ishai et al., Private Circuits II: Keeping Secrets in Tamperable Circuits, EUROCRYPT, 2006]
[Richter-Brockmann, Sasdrich, **Güneysu**, Revisiting Fault Adversary Models - Hardware Faults in Theory and Practice, IEEE TC 2022]

Modeling Fault-Injection Attacks [RSG21]

Parametrizable adversary model $\zeta(n, t, l)$

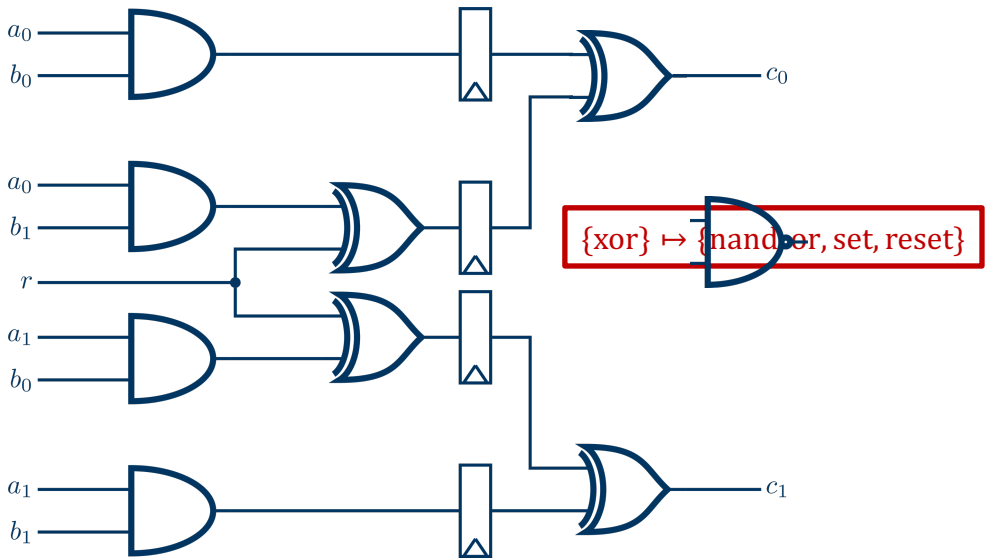


Modeling Fault-Injection Attacks [RSG21]



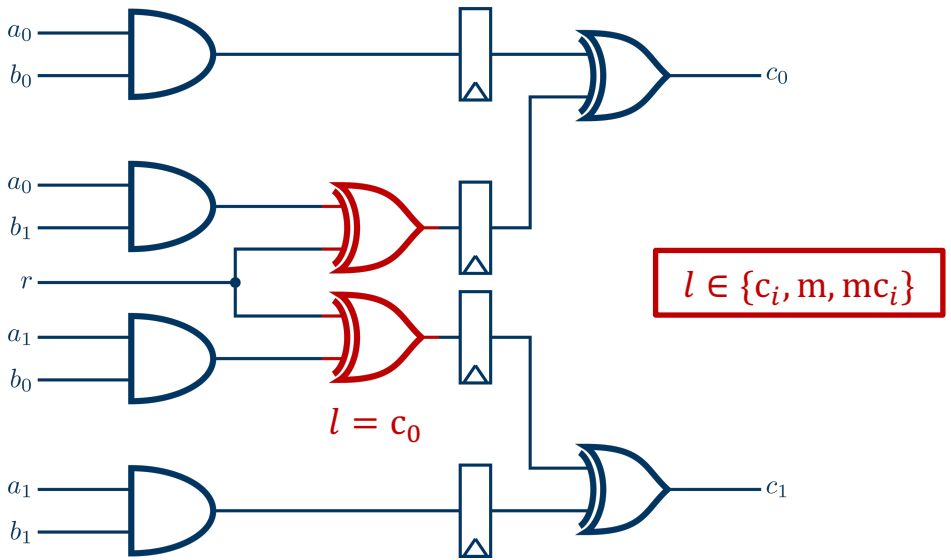
Modeling Fault-Injection Attacks [RSG21]

$$\zeta(n, t, l)$$



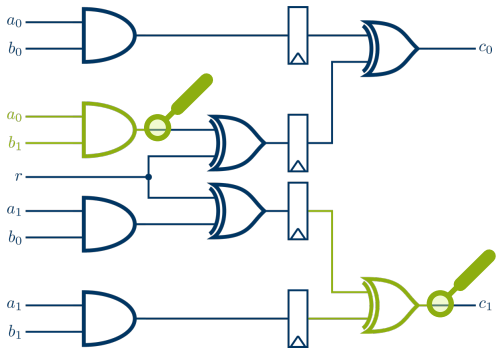
Modeling Fault-Injection Attacks [RSG21]

$$\zeta(n, t, l)$$

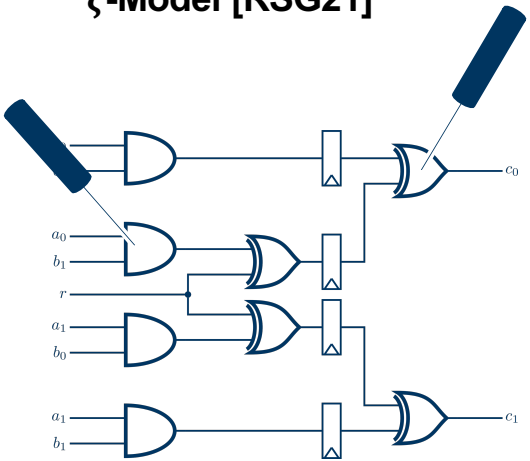


Formal Verification of Countermeasures

Glitch-Extended d-probing Model [FGP+18]



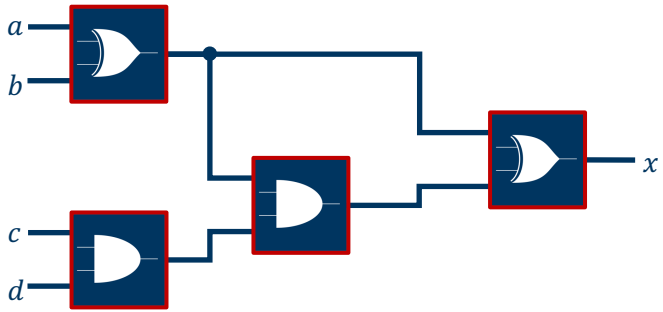
ζ-Model [RSG21]

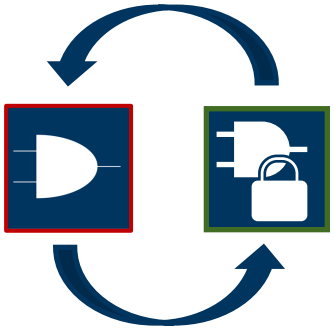


Both models can be used to guide computer-aided design and verification frameworks!

Advanced Models – Protection by Composable and Secure Gadgets

Insecure Circuit



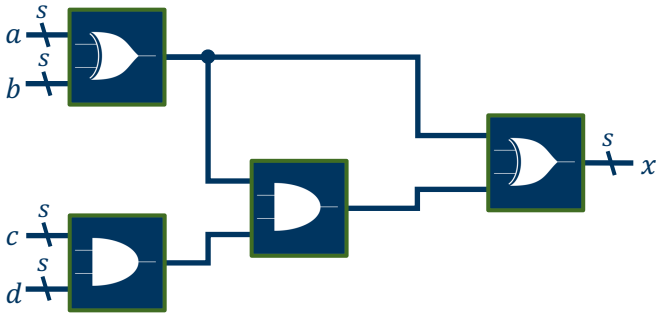


Replace insecure gates by secure gadgets

Share inputs and outputs

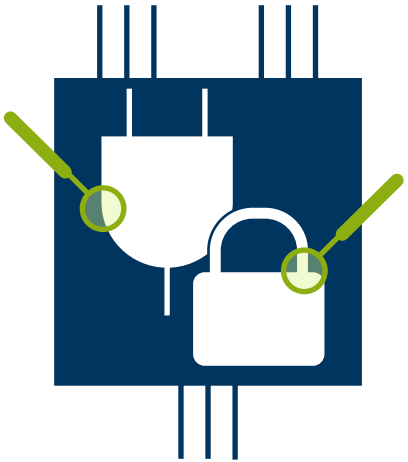
Maintain timing (pipelining)

Protected Circuit



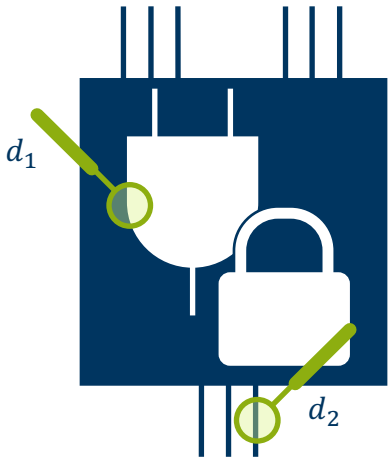
Modeling Side-Channel Attacks – Composability

PNI [BBD+15]
Probe Non-Interference



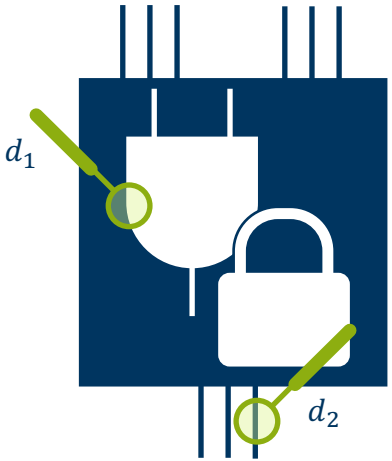
$d' \leq d$

PSNI [BBD+16]
Probe Strong Non-Interference



$d_1 + d_2 \leq d$

PINI [CS20]
Probe-Isolating Non-Interference

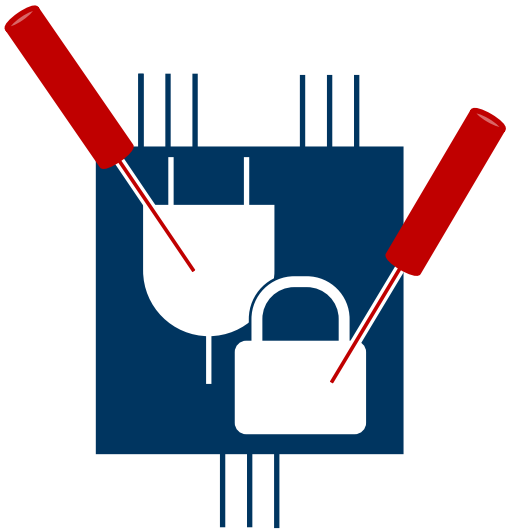


$d_1 + d_2 \leq d$

[BBD+15] Barthe, Belaïd, Dupressoir, Fouque, Grégoire, Strub: Verified Proofs of Higher-Order Masking. EUROCRYPT (1) 2015
[BBD+16] Barthe, Belaïd, Dupressoir, Fouque, Grégoire, Strub, Zucchini: Strong Non-Interference and Type-Directed Higher-Order Masking. CCS 2016
[CS20] Cassiers, Standaert: Trivially and Efficiently Composing Masked Gadgets With Probe Isolating Non-Interference. IEEE TIFS, 2022

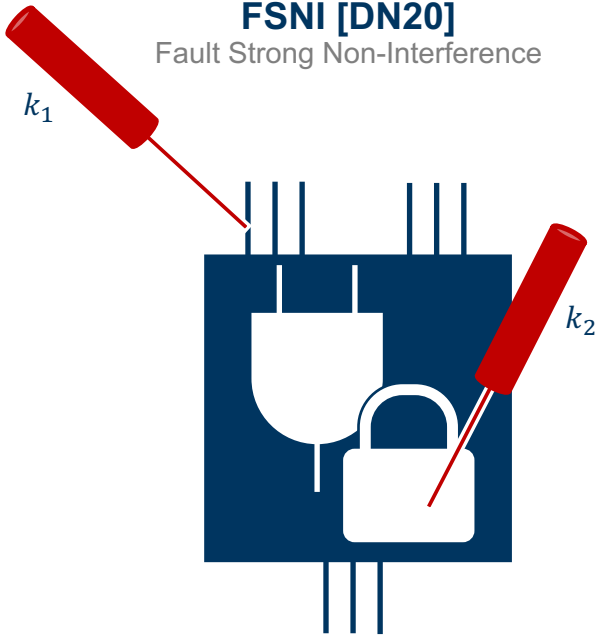
Modeling Fault-Injection Attacks – Composability

FNI [DN20]
Fault Non-Interference



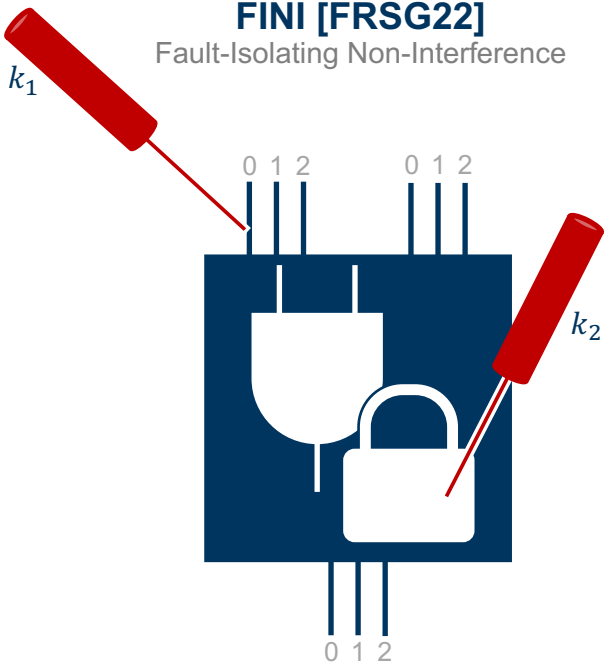
$k' \leq k$

FSNI [DN20]
Fault Strong Non-Interference



$k_1 + k_2 \leq k$

FINI [FRSG22]
Fault-Isolating Non-Interference

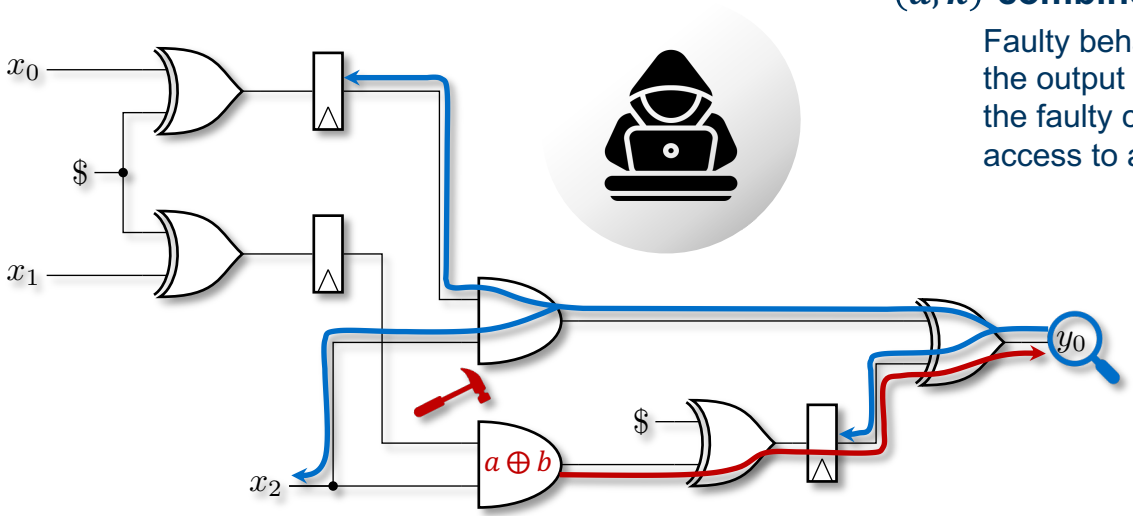


$k_1 + k_2 \leq k$

[DN20] Dhooghe and Nikova., Let's Tessellate: Tiling for Security Against Advanced Probe and Fault Adversaries, CARDIS 2020
[FRSG22] Feldtkeller, Richter-Brockmann, Sasdrich, Güneysu, CINI-MINIS: Domain Isolation for Fault and Combined Security, CCS 2022

Threshold Combined Model

Free placement of up to **k faults** on gates or randomness and up to **d probes** on wire.



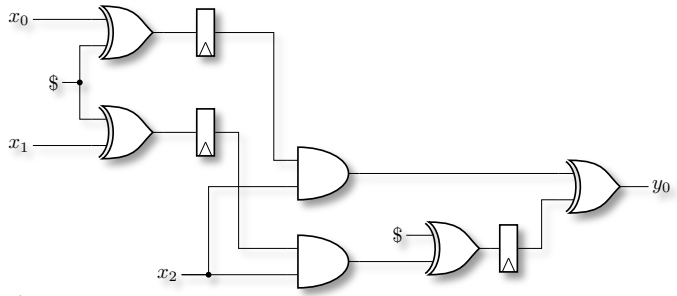
(d, k)-combined security

Faulty behavior can be detected/corrected at the output and the adversarial observation in the faulty circuit can be simulated without access to any secret.

[Dhooghe and Nikova, My Gadgets Just Cares for Me – How NINA Can Prove Security Against Combined Attacks, 2020]
[Richter-Brockmann, Feldtkeller, Sasdrich, Güneysu, VERICA – Verification of Combined Attacks, CHES 2022]

Trivial Approach

Masking & Replication



Unprotected Circuit

Masking

Replication

Combined Secure ?

Unfortunately, not…

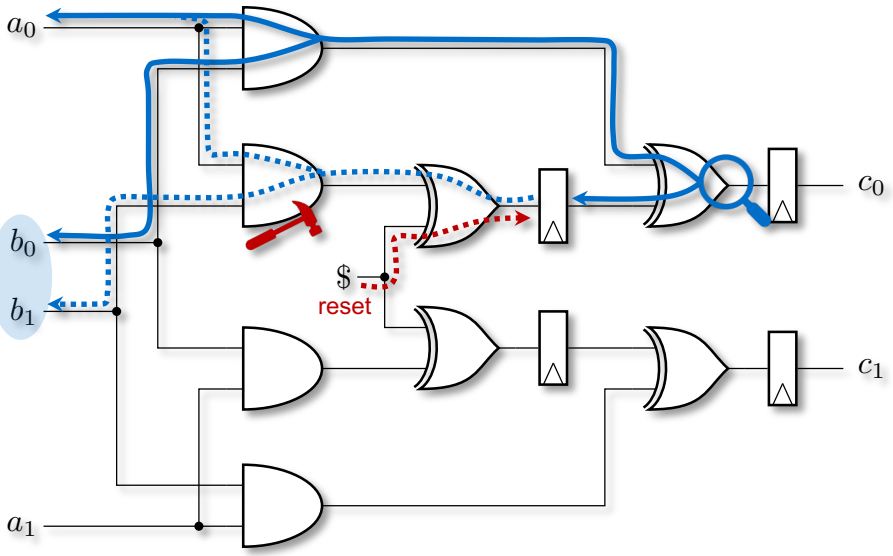
Why not?

Removal of Entropy

Fresh randomness is used to randomize intermediate values.

Would obviously be secure without faults.

When faulted, security guarantees of refreshing breaks down.

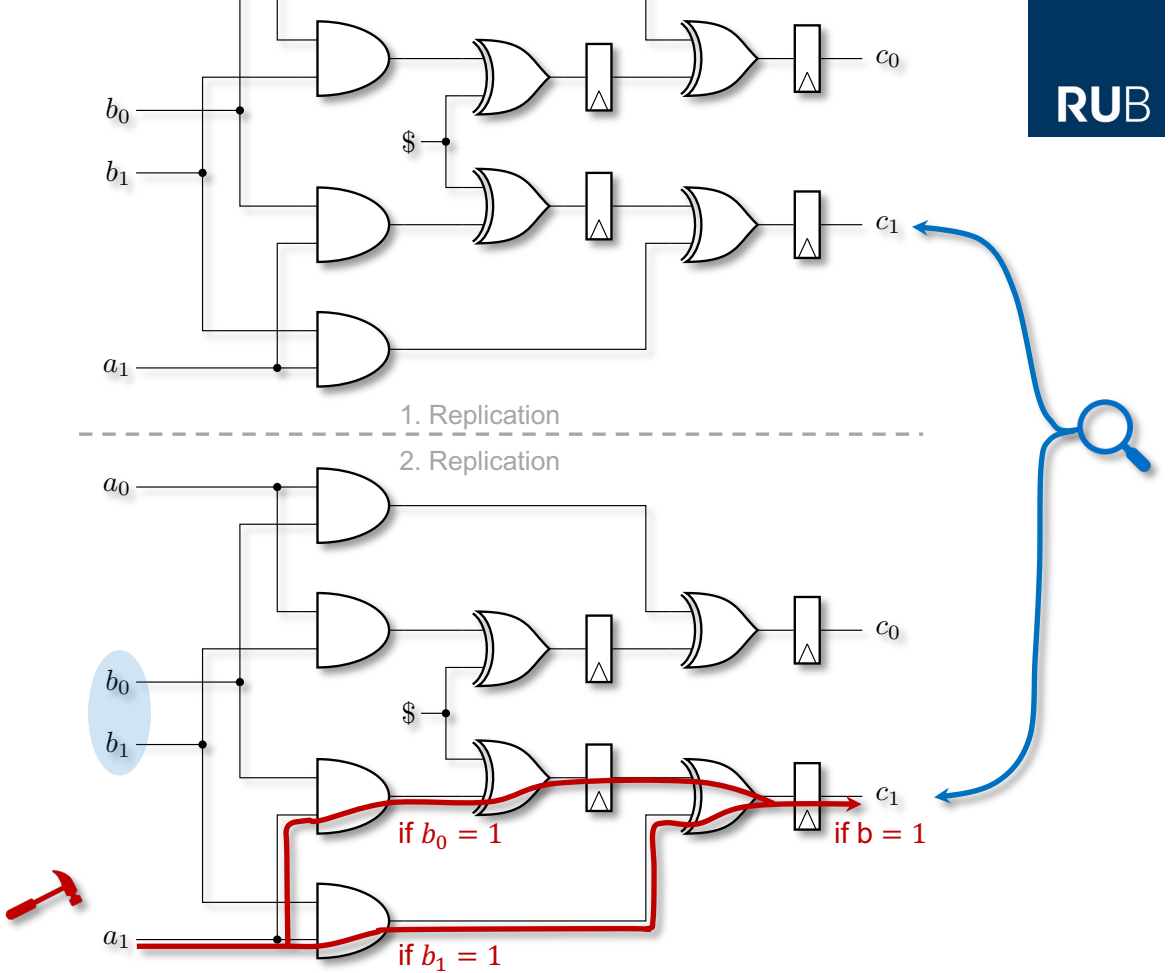


Why not?

Conditional Fault Propagation

Leakage through the observation of the effectiveness of faults.

Would be secure without probes.



Combined Security

General Approach

Masking & Replication

Intermediate Fault Handling

Where to place intermediate fault handling?

CINI: Combined-secure gadgets based on PINI

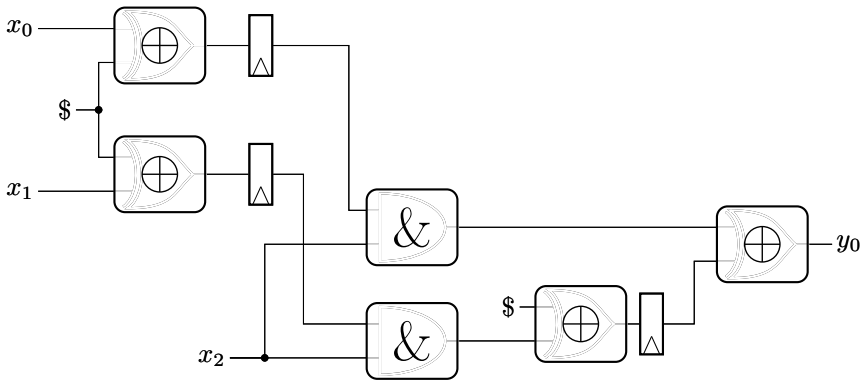
Feldtkeller, Richter-Brockmann, Sasdrich, Güneysu, CINI-MINIS: Domain Isolation for Fault and Combined Security, CCS 2022
Feldtkeller, Güneysu, Moos, Richter-Brockmann, Saha, Sasdrich, Standaert, Combined Private Circuits – Combined Security Refurbished, CCS 2023

CTI: Combined-secure circuits from TI

Feldtkeller, Richter-Brockmann, Sasdrich, **Güneysu**, Combined Threshold Implementation, CHES 2024

Combined Isolated Non-Interference

Gadgets - Principle



Replace every gate by a corresponding gadget

Gadget:
Small circuit that implements some functionality adhering to some security and compositional property.

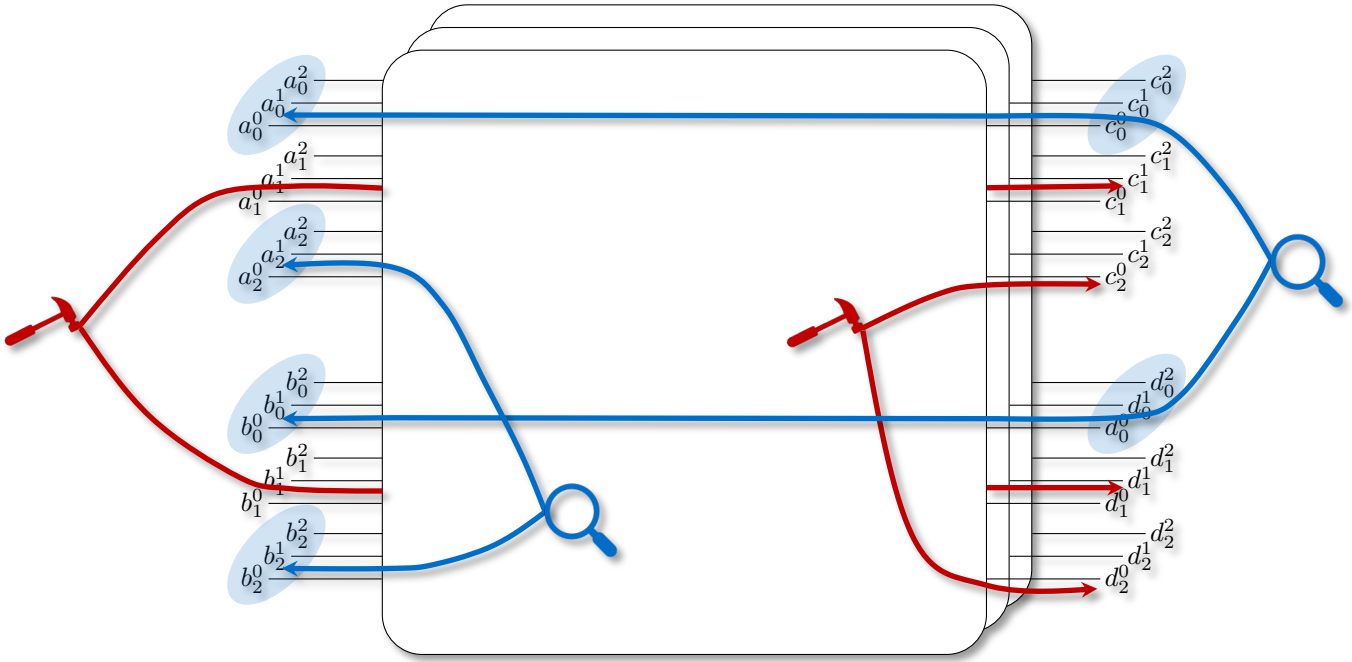
Focus on security analysis for small building blocks.

[Barthe et al., Verified Proofs of Higher-Order Masking, 2015]

Combined Isolated Non-Interference

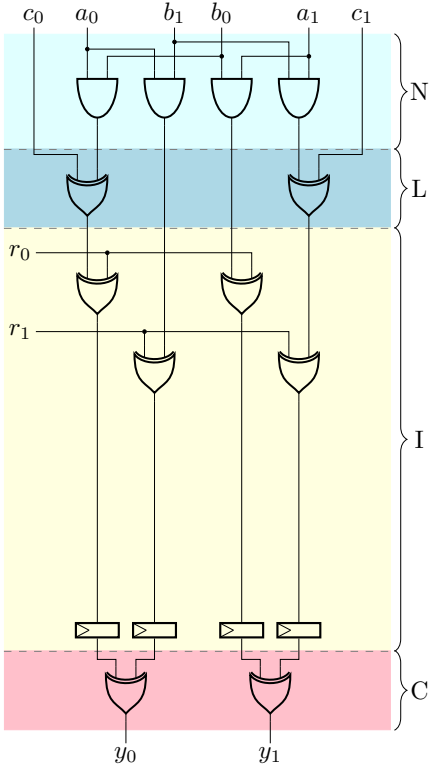
Gadget Properties

Isolation of **probe propagation within share domains** and **fault propagation within shared redundancy domain**, i.e., shares with same share and replication index.



Combined Threshold Implementations

Transformation



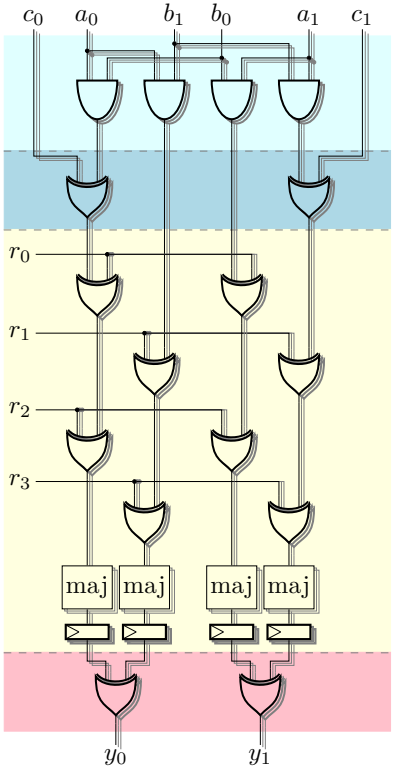
Starting from a CMS design
(generalization of TI – structured into layers)

Transformation:

1. Refresh Replication
   Each value is refreshed with at least $k + 1$ random bits.
   - Faults cannot remove entropy

2. Circuit Replication
   Entire circuit is replicated $2k + 1$ times, using the same randomness.
   - Stand-alone fault security.

3. Corrections
   k consecutive corrections after refreshing.
   - Fault isolation



Performance

AES S-box

Secure against one fault and one probe simultaneously and implemented using the 45 nm OpenCell library.

Can be efficiently automated and built into cryptographic design tools

simple to use

Design			Performance		
Version	s	n	Area [GE]	Lat. [Cycle]	Rand. [Bit]
λ-detection M&M	3	-	29300.0	6	903
Let's Tessellate†	3	2	17550.0	7	216
$\widehat{\mathbf{CPC_1}}$	2	3	10882.0	6	144
CCMS	2	3	6576.0	5	62
CTI	4	3	11690.0	3	0

s - number of shares n - number of replications

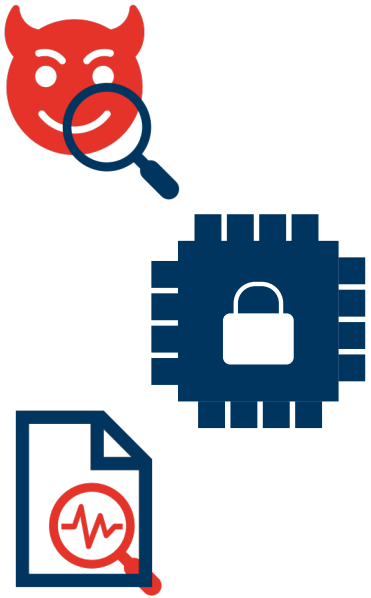
† Without abort handling

[Hirata et al., All You Need Is Fault: Zero-Value Attacks on AES and New λ-Detection M&M, 2024]
[Dhooghe and Nikova., Let's Tessellate: Tiling for Security Against Advanced Probe and Fault Adversaries, CARDIS 2020]
[Feldtkeller, Richter-Brockmann, Sasdrich, Güneysu, Combined Threshold Implementation, CHES 2024]

DIRECTIONS & CHALLENGES

Directions and Challenges for Research

Recall: What were key observations and research questions from the beginning?



RQ1: Can we **universally and reliably determine success probabilities** of fault events on specific/structurally identical/similar hardware?

RQ2: How can we efficiently identify and model critical components (gates, wires, memories) affected by **any type of fault event?**

RQ3: What are **better generic design principles** (beyond duplication) to eliminate the impact of fault events?

RQ4: How can we **efficiently validate the correct operation and fault coverage** of countermeasures?

Solutions to Date: Fault Protection Design and Validation

Models how to applying composable security gadgets to crypto circuits

- **Threshold models** (d, k): expensive in terms of #gates, latency, #random bits
- **Random models** (ε, μ) [BFG+24]: more efficient but hard interpretation of results

RQ2

Verification approaches: Validation of generic implementations via tools (selection):

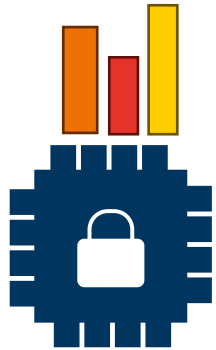
- verFI [AWMN20] – Cryptographic fault diagnosis simulator
- FIVER & VERICA [RSS+21, RFSG22] – threshold fault model verification
- FaultDetective [LSS24] – Fault tracing tool from processor design to software
- FIESTA [UMM25] at CHES 2025 – approximation of generalized random fault model

RQ4

[BFG+24] Belaid, Feldtkeller, Güneysu, Guinet, Richter-B., Rivain, Sasdrich, Taleb, Formal Definition and Verification for Combined Random Fault […], ASIACRYPT 2024
[AWMN20] Arribas, Wegener, Moradi, Nikova, Cryptographic Fault Diagnosis using VerFI, HOST 2020
[RFSG22] Richter-Brockmann, Feldtkeller, Sasdrich, Güneysu: VERICA - Verification of Combined Attacks Automated formal verification of security […], CHES, 2022
[LSS24 Liu, Shanmugam, Schaumont, FaultDetective Explainable to a Fault, from the Design Layout to the Software, CHES, 2024
[UMM25 Uhle, Feldtkeller, Moradi, Formal Definition and Verification for Combined Random Fault and Random Probing Security, ASIACRYPT 2024

Open Challenges: Fault Protection, Determination and Replication

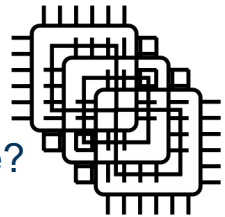
Most fault injection analyses and countermeasures are bound to a specific target…



RQ1: How can we achieve generic reproducibility of fault injection results?

- How can we efficiently investigate success probabilities for a specific fault event?
- Can we build a universal/unified fault setup and injection procedures?

RQ3: Are there other generic and efficient countermeasures than duplication?

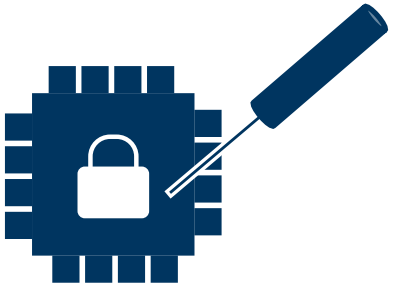


- Which detection/correction mechanisms can be universally applied (e.g., codes)?
- Can these be less expensive (latency, acrea) than duplication at better fault coverage?
- What are efficient + universal design patterns for cover fault protection in EDA tools?

CONCLUSIONS

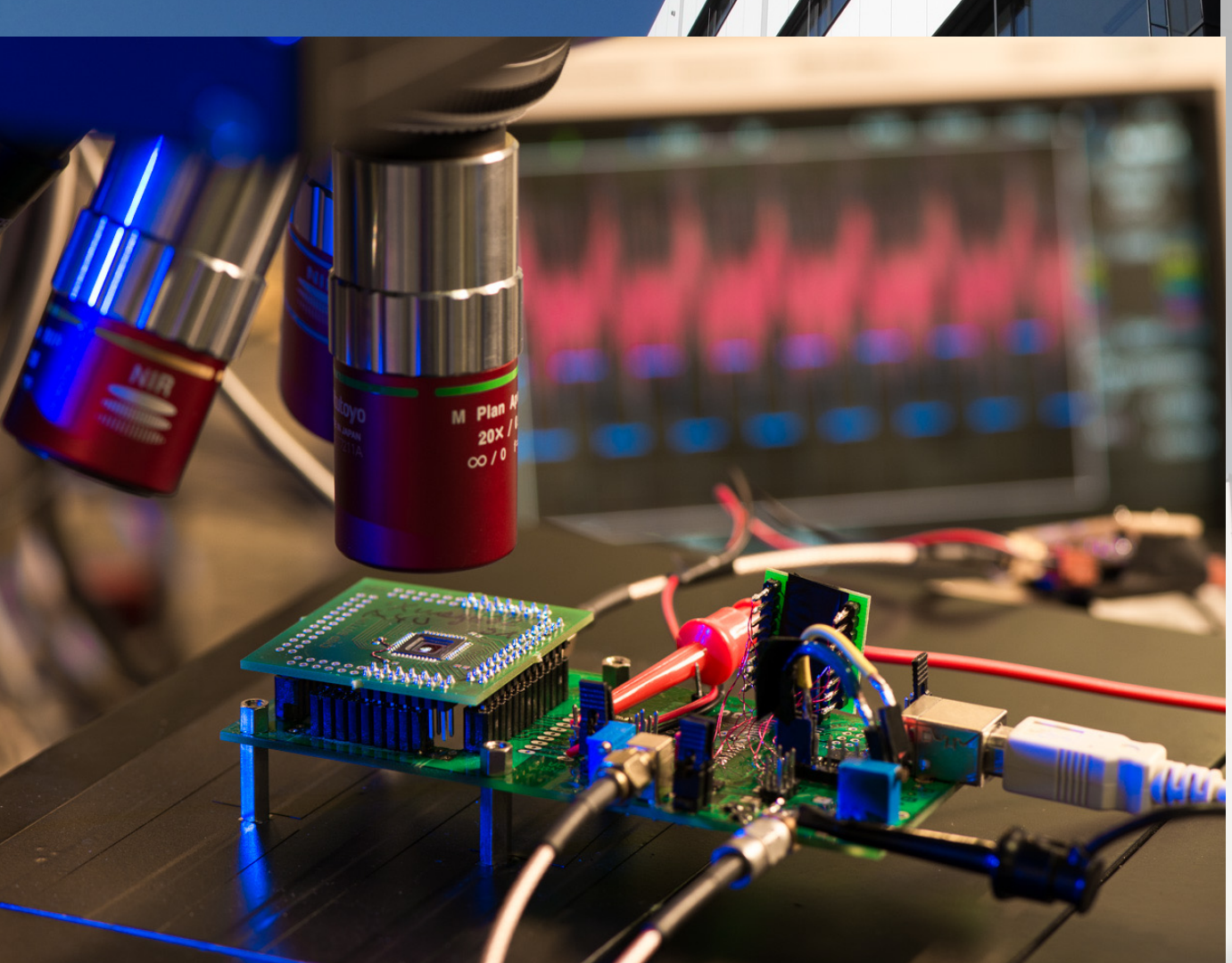
Conclusions

FDTC community has still a few open challenges to tackle



- **Build frameworks to guide early detection and prevention of fault event at early-stage and design time**
  - Not only by FIA experts but also for use by theorists/cryptanalysts
  - Generic assessment of probabilities of fault events in specific/structurally similar targets
  - Reproducible results in different evaluation setups

- **Develop structural countermeasures for integration in EDA tools**
  - Better coverage against multi-fault/biased attacks than duplication
  - Efficient and (at best) generically applicable



Thank you!

Tim Güneysu
Security Engineering

tim.gueneysu@rub.de

RUB

SECURITY ENGINEERING