CAPABARA: A Combined Attack on CAPA

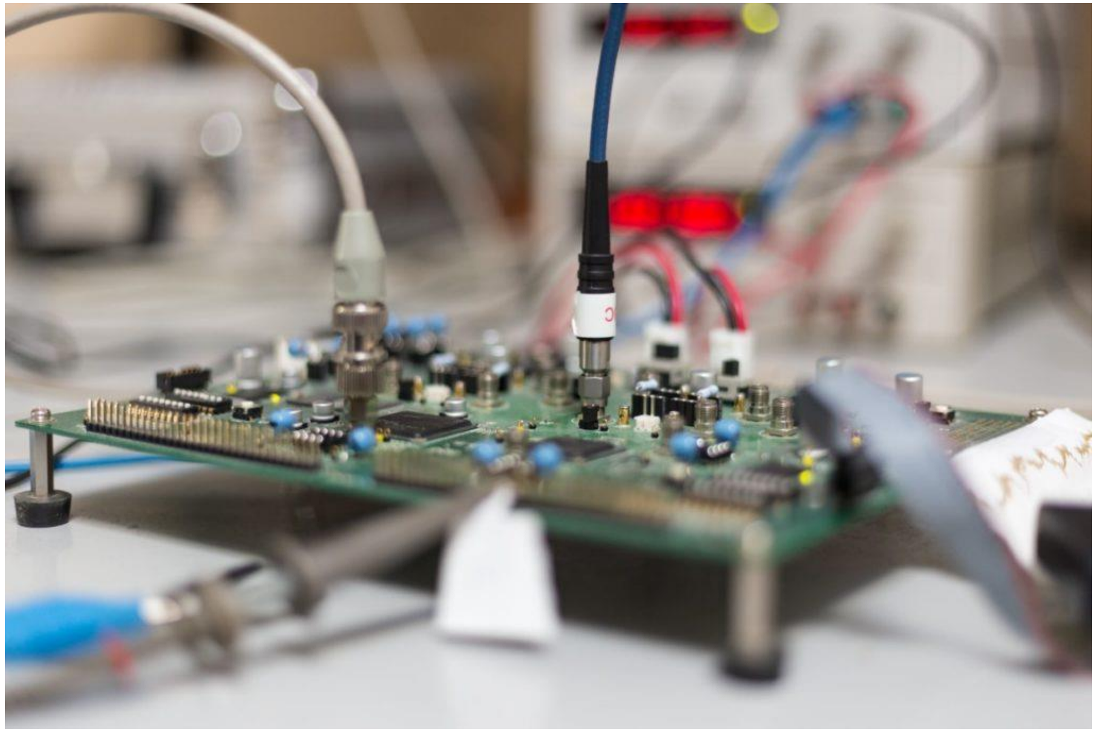
Dilara Toprakhisar, Svetla Nikova, Ventzislav Nikov

COSADE’24 09.04.2024

Physical Attacks

1- Passive Attacks

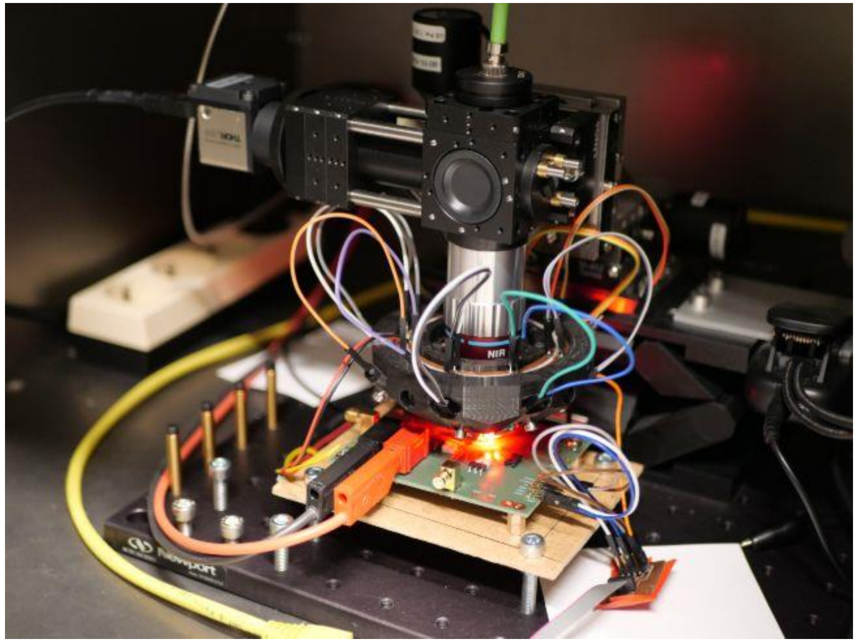
- Side-Channel Analysis (SCA)
  - Exploits the observable leakage
- Masking
  - Statistically independent random shares



Physical Attacks

2- Active Attacks

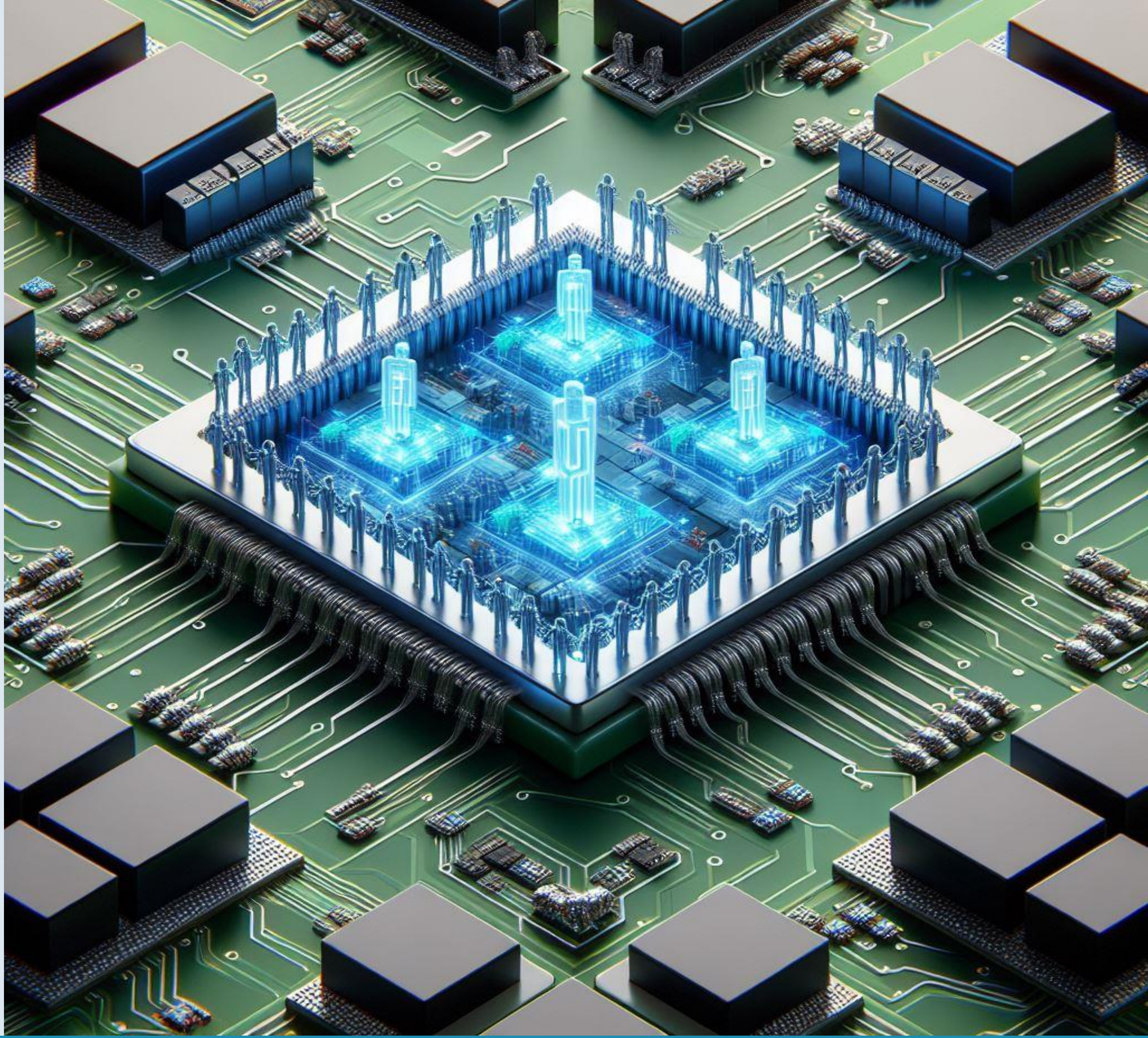
- Fault Attacks (FA)
  - Intentionally disturb computations
  - Initially exploited wrong ciphertexts, *e.g.*, DFA, SFA
- Redundancy
  - SIFA (targeting registers/linear operations)
- Redundancy + masking
  - SIFA-2 (targeting nonlinear operations)



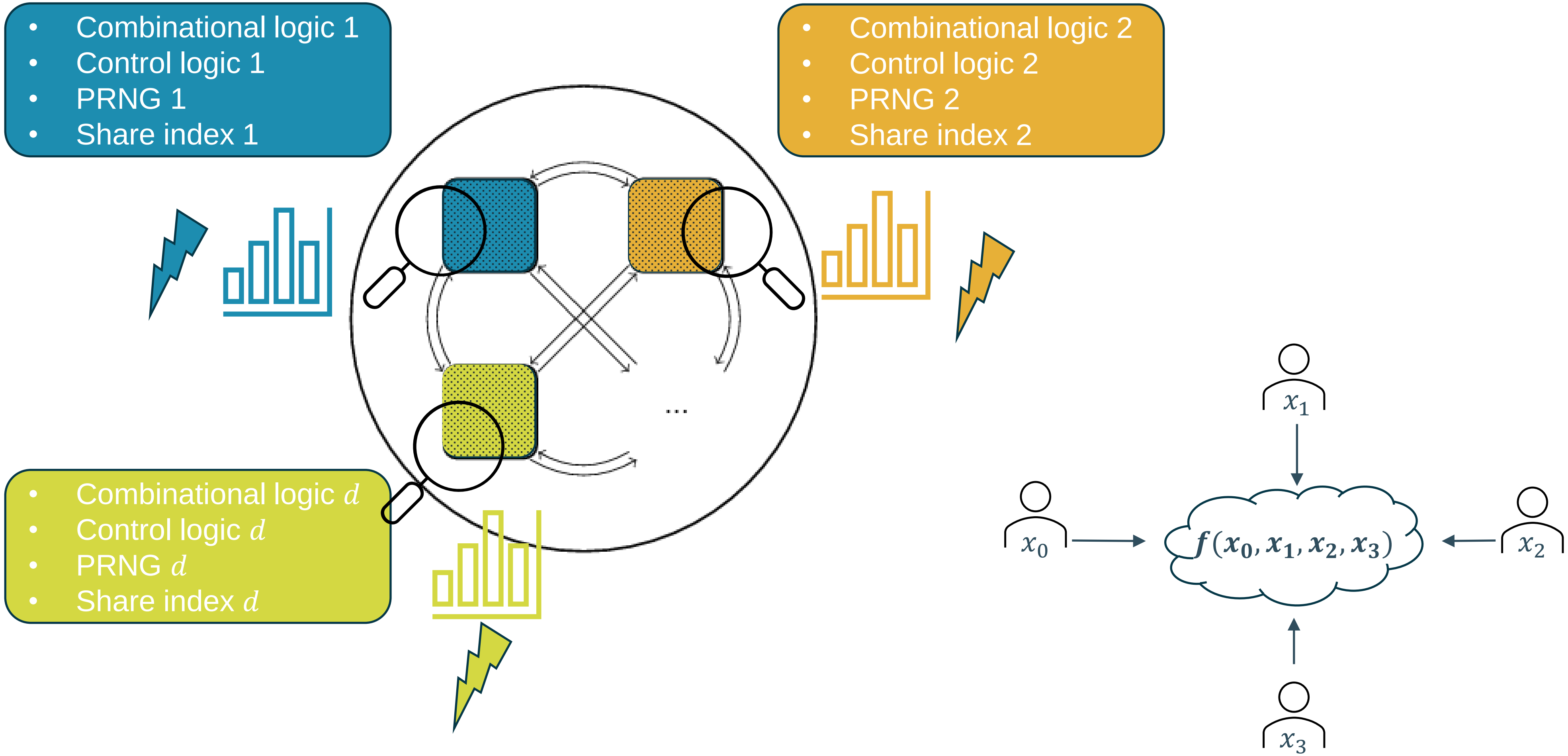
3- Combined Attacks

- SCA + FA

The CAPA Countermeasure



The Tile-Probe-and-Fault Model



The Tile-Probe-and-Fault Model

Probing

1. d_p-probing
   - All intermediate variables within d_p tiles
     - From beginning to the end
     - With a probability one

Faulting

1. d_f-faulting
   - Chosen value faults
   - Any number of precisely chosen variables within d_f tiles
2. ϵ-faulting
   - Random value faults
   - Any variable within any tile

The Tile-Probe-and-Fault Model

Adversary $\mathcal{A}_1$

- d_p-probing + d_f-faulting
- $d_p, d_f \leq d - 1$
- At least one share/tile is unaccessed

Adversary $\mathcal{A}_2$

- d_p-probing + ϵ-faulting
- $d_p \leq d - 1$

The CAPA Design

- Preprocessing stage
  - Auxiliary data
  - Denoted with $a, b, c, \ldots$
- Evaluation stage
  - Denoted with $x, y, z, \ldots$
- Works over $\mathbb{F}_{2^n}$
- α is the MAC key
  - $\tau_x = \alpha x$ denotes the tag of x
- Boolean masking
  - $x = x_0 + x_1 + \cdots + x_{d-1}$

Multiplication ($z = xy, \tau_z = \alpha xy$)

$x_0, \tau_{x_0}, y_0, \tau_{y_0}$
a_0, b_0, c_0
$\tau_{a_0}, \tau_{b_0}, \tau_{c_0}$

$\mathcal{T}_0$

Beaver triple
$c_0 + c_1 = (a_0 + a_1)(b_0 + b_1)$
$c = ab$

$\mathcal{T}_1$

$x_1, \tau_{x_1}, y_1, \tau_{y_1}$
a_1, b_1, c_1
$\tau_{a_1}, \tau_{b_1}, \tau_{c_1}$

Step 1: blinding

$\mathcal{E}_0 = x_0 + a_0, \eta_0 = y_0 + b_0$
$\tau_{\mathcal{E}_0} = \tau_{x_0} + \tau_{a_0}, \ \tau_{\eta_0} = \tau_{y_0} + \tau_{b_0}$

$\mathcal{E}_1 = x_1 + a_1, \eta_1 = y_1 + b_1$
$\tau_{\mathcal{E}_1} = \tau_{x_1} + \tau_{a_1}, \ \tau_{\eta_1} = \tau_{y_1} + \tau_{b_1}$

Step 2: partial unmasking

$\mathcal{E} = \mathcal{E}_0 + \mathcal{E}_1, \eta = \eta_0 + \eta_1$

$\mathcal{E} = \mathcal{E}_0 + \mathcal{E}_1, \eta = \eta_0 + \eta_1$

Step 3: MAC tag check

$\mathcal{E}, \tau_{\mathcal{E}_0}, \eta, \tau_{\eta_0}$

$\mathcal{E}, \tau_{\mathcal{E}_1}, \eta, \tau_{\eta_1}$

Step 3: Beaver computation

$z_0 = c_0 + \mathcal{E}b_0 + \eta a_0 + \mathcal{E}\eta$
$\tau_{z_0} = \tau_{c_0} + \mathcal{E}\tau_{b_0} + \eta\tau_{a_0} + \alpha_0\mathcal{E}\eta$

$z_1 = c_1 + \mathcal{E}b_1 + \eta a_1$
$\tau_{z_1} = \tau_{c_1} + \mathcal{E}\tau_{b_1} + \eta\tau_{a_1} + \alpha_1\mathcal{E}\eta$

MAC Tag Check

$\mathcal{E}, \tau_{\mathcal{E}0}, \eta, \tau_{\eta 0}$ — $\mathcal{T}_0$

$\mathcal{T}_1$ — $\mathcal{E}, \tau_{\mathcal{E}1}, \eta, \tau_{\eta 1}$

$\alpha_0\mathcal{E} + \tau_{\mathcal{E}_0}$
$\alpha_0\eta + \tau_{\eta_0}$

$\alpha_1\mathcal{E} + \tau_{\mathcal{E}_1}$
$\alpha_1\eta + \tau_{\eta_1}$

$\alpha_0\mathcal{E} + \alpha_1\mathcal{E} = \tau_{\mathcal{E}} = \tau_{\mathcal{E}_0} + \tau_{\mathcal{E}_1}$

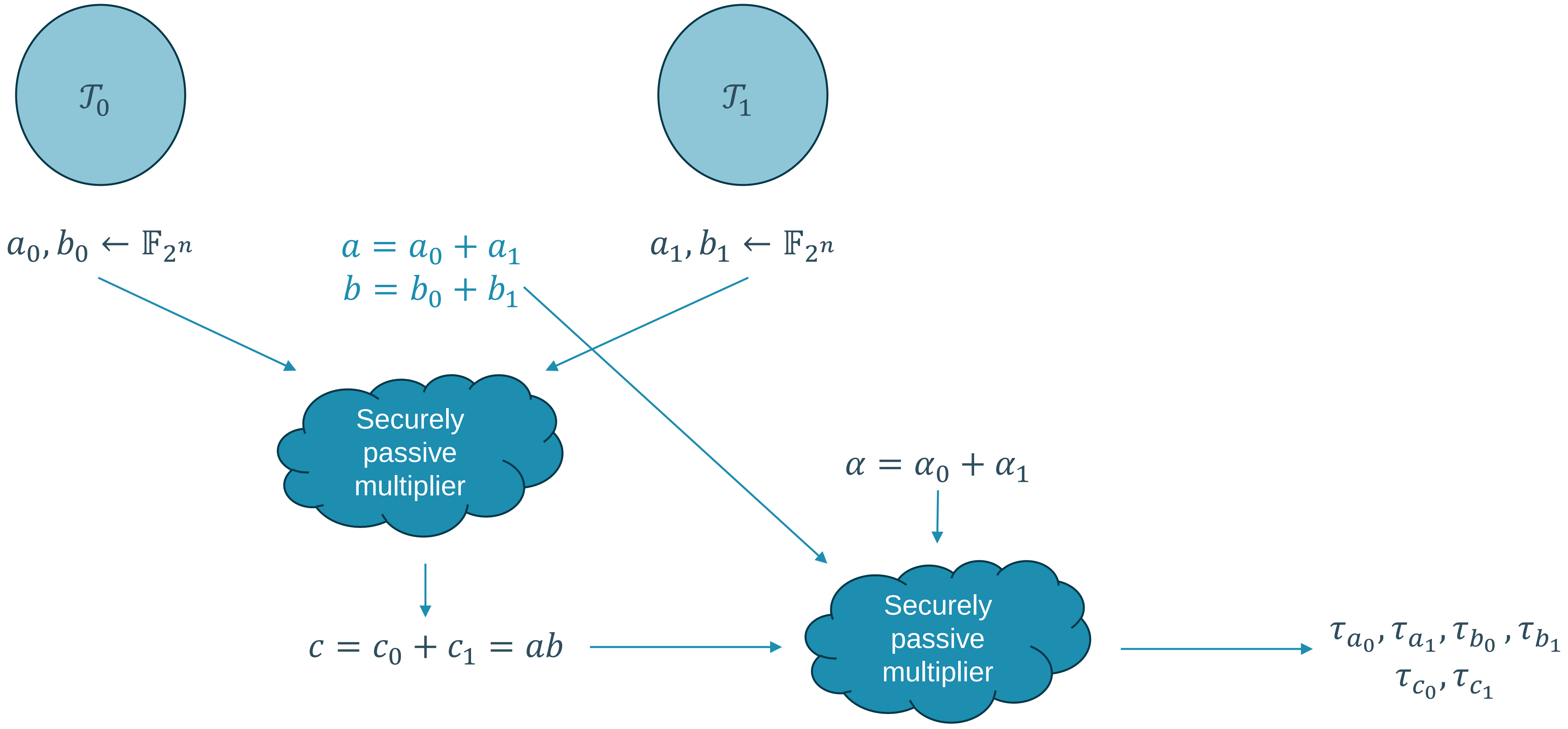
$\alpha_0\mathcal{E} + \tau_{\mathcal{E}_0} + \alpha_1\mathcal{E} + \tau_{\mathcal{E}_1} =?\, 0$
$\alpha_0\eta + \tau_{\eta_0} + \alpha_1\eta + \tau_{\eta_1} =?\, 0$

$\alpha_0\mathcal{E} + \tau_{\mathcal{E}_0} + \alpha_1\mathcal{E} + \tau_{\mathcal{E}_1} =?\, 0$
$\alpha_0\eta + \tau_{\eta_0} + \alpha_1\eta + \tau_{\eta_1} =?\, 0$

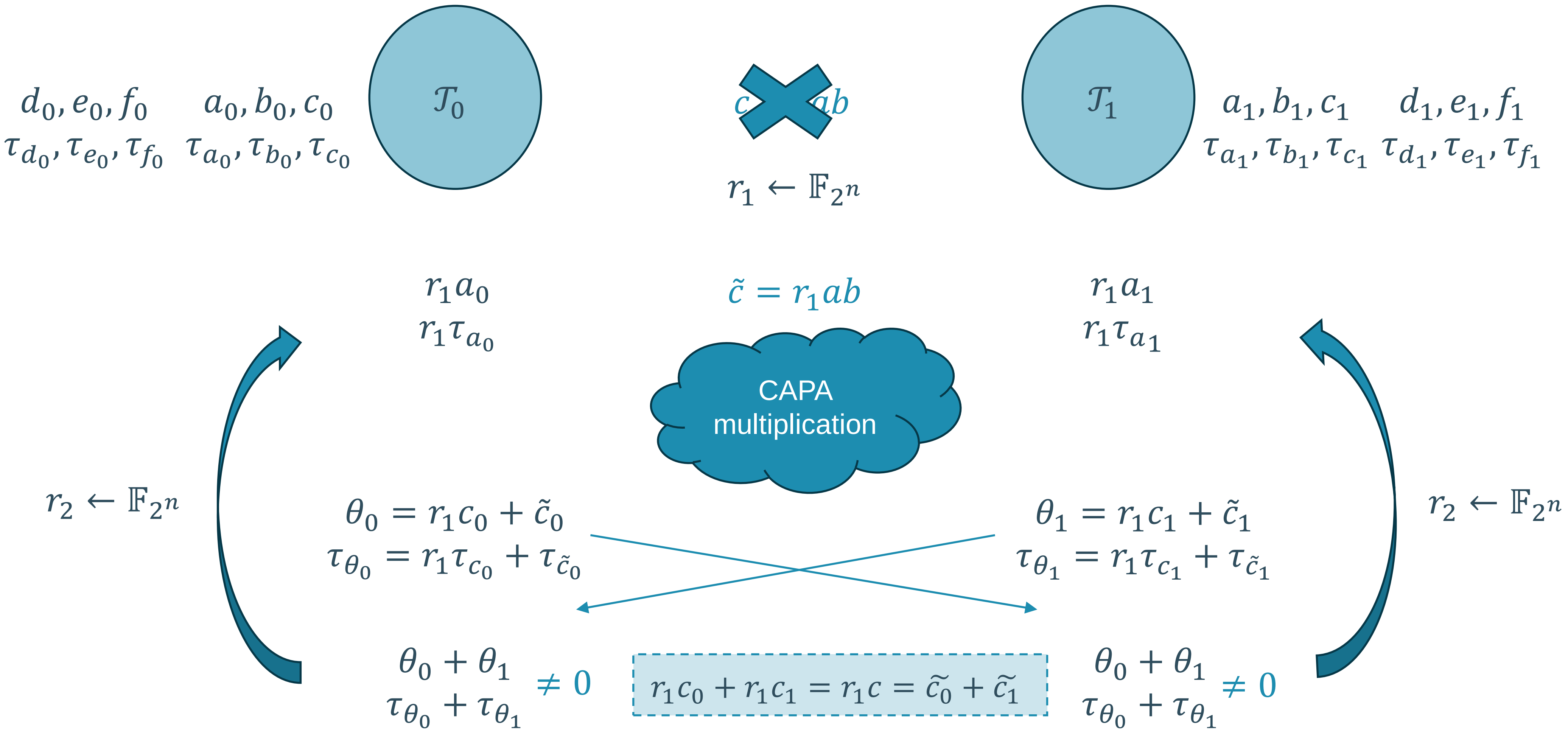
$\neq 0 \rightarrow$ ABORT

$\neq 0 \rightarrow$ ABORT

Beaver Triples (a, b, c)



Relation Verification of Beaver Triples

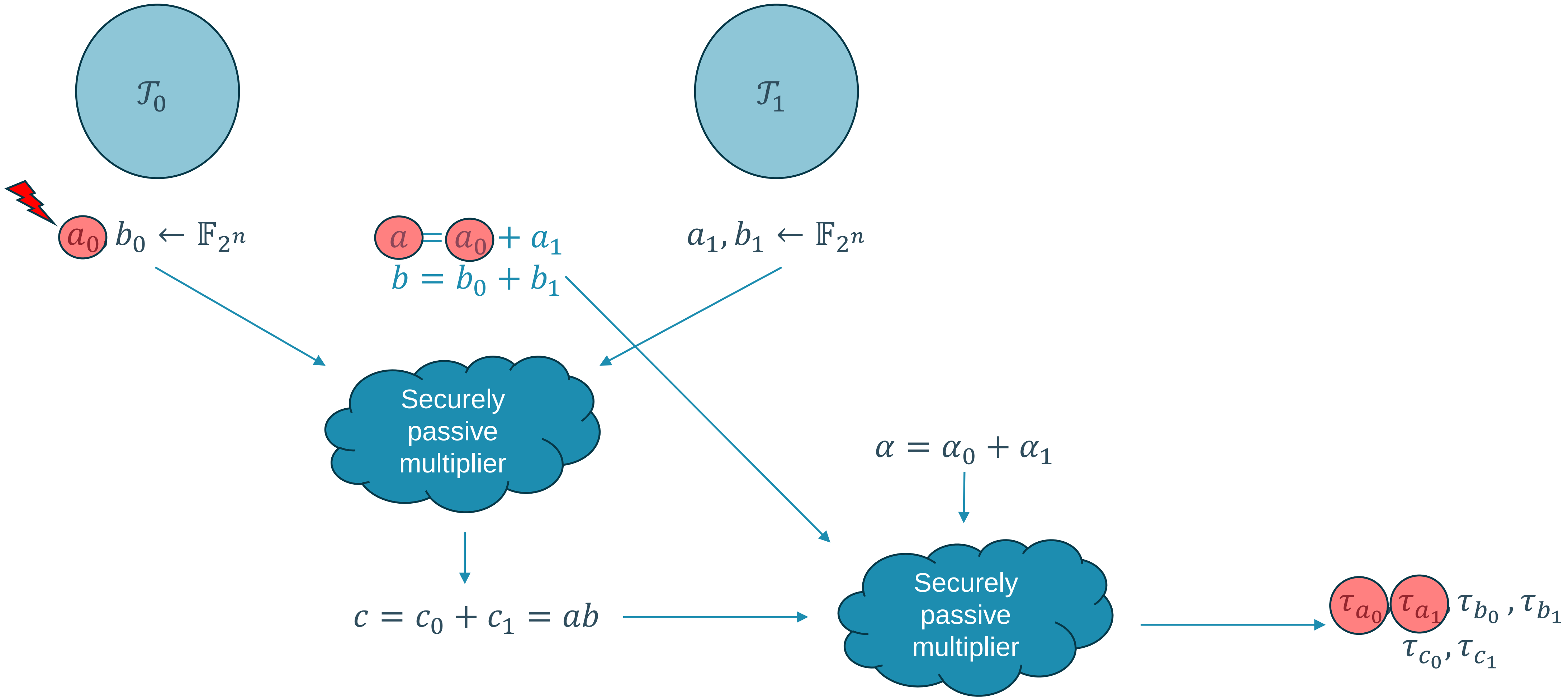


CAPABARA:
The Combined Attack Description

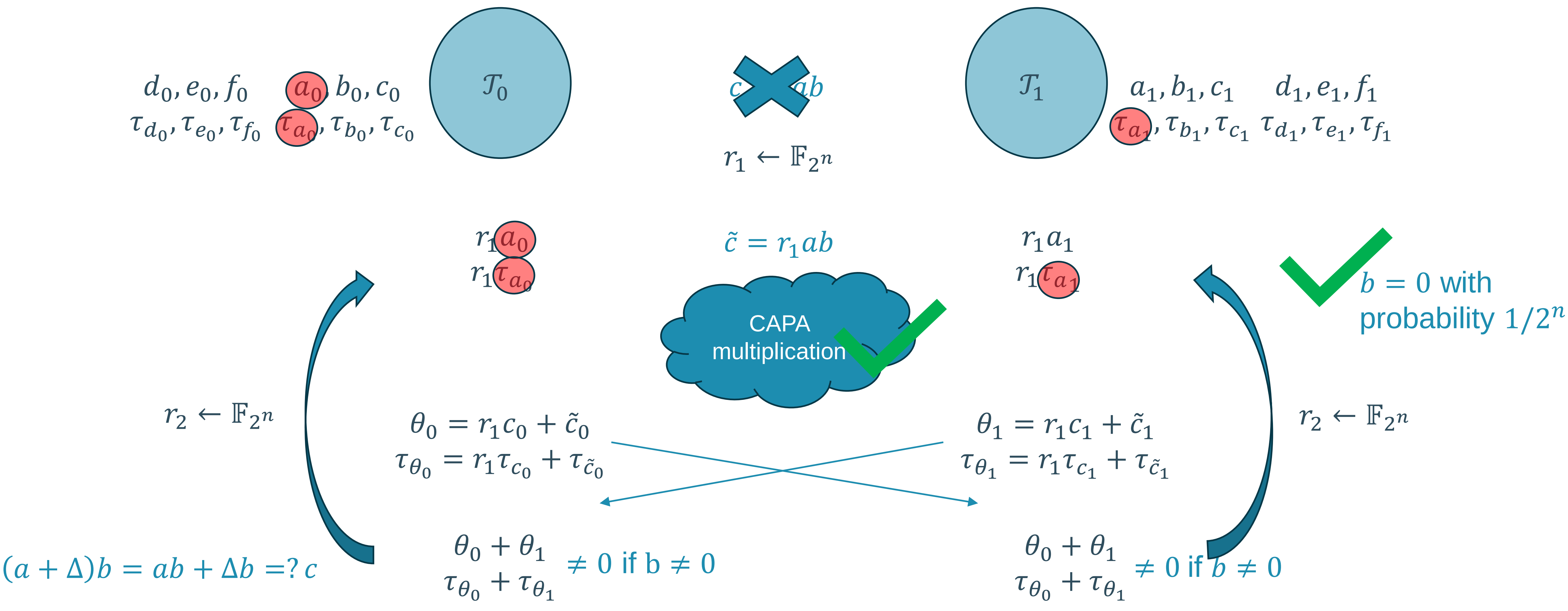
Adversarial Model of CAPABARA

- Single-shot transient fault to a variable in $\mathbb{F}_{2^n}$
- Loose fault location
- Precise fault timing
- Any type of fault injected to a register
- Probing a chosen variable
- Stays within the tile-probe-and-fault model
  - Also works with t-probing and gate/register faulting models

Fault Injection Step



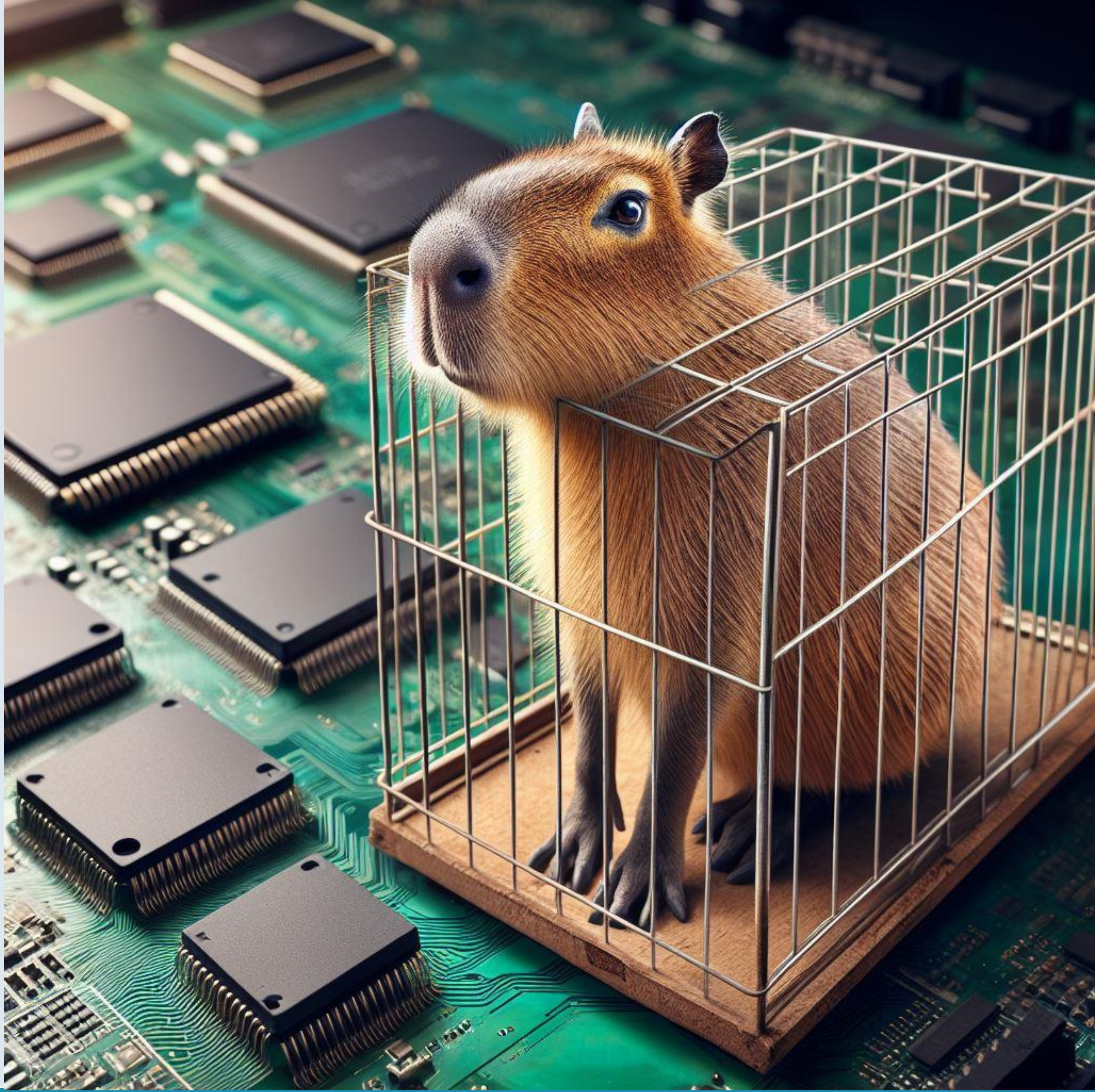
Fault Injection Step



Probing Step

- (a', b, c) passes the relation verification
  - This implies $b = 0, c = 0$
- $b = 0$ is used to blind one of the inputs in CAPA multiplication
  - $\eta = y + b = y$ is unmasked

Fixes Against the Proposed Attack



Fixes Against the Proposed Attack

1. Computing the tags of a and b prior to forming the triple
   - *CAPABARA: a is faulted after c is computed, before the tags are computed*
   - **Three** fault injections with the **same** success probability
     - A fault is injected to a (a') before its tag is computed
     - After the tag is computed, the same fault is injected to a' again to revert it (a)
       - c is computed using correct a and b
     - The same fault is injected to a (a') again

Fixes Against the Proposed Attack

2. Randomly choosing the Beaver triple to be used in the multiplication
    - CAPA can choose between (a, b, c) and (d, e, f) to be used for blinding
        - **Single** fault injection with **half** of the initial success probability
        - **Two** fault injections for the **same** success probability
    - Multiple (m) Beaver triples can be generated, and two of them can be chosen for the relation verification
        - **Single** fault injection with $\mathbf{1/m}$ of the initial success probability
        - $\boldsymbol{m}$ fault injections for the **same** success probability

Fixes Against the Proposed Attack

3. Zero check on c
   - Indirectly checks if $a = 0$ or $b = 0$, preventing ineffective faults
   - Compromises the uniformity of the unmasked blinded multiplication inputs

Summary

- Single fault injection in a Beaver triple $(\boldsymbol{a}, b, c)$ + single probe
- The attack is successful $\leftrightarrow b = 0$
  - Probability $1/2^n$
  - $b = 0 \rightarrow$ an unmasked variable occurs some cycles after the injection
- The fault does not need to be repeatable
- Proposed fixes
  - Increased attack complexity
  - New vulnerability

Thank you!