Divided We Stand, United We Fall: Security Analysis of Some SCA+SIFA Countermeasures Against SCA-Enhanced Fault Template Attacks

Sayandeep Saha, Arnab Bag, Dirmanto Jap, Debdeep Mukhopadhyay, Shivam Bhasin

ASIACRYPT 2021

Motivation

Crypto-implementations in real life leak sensitive information:

Motivation

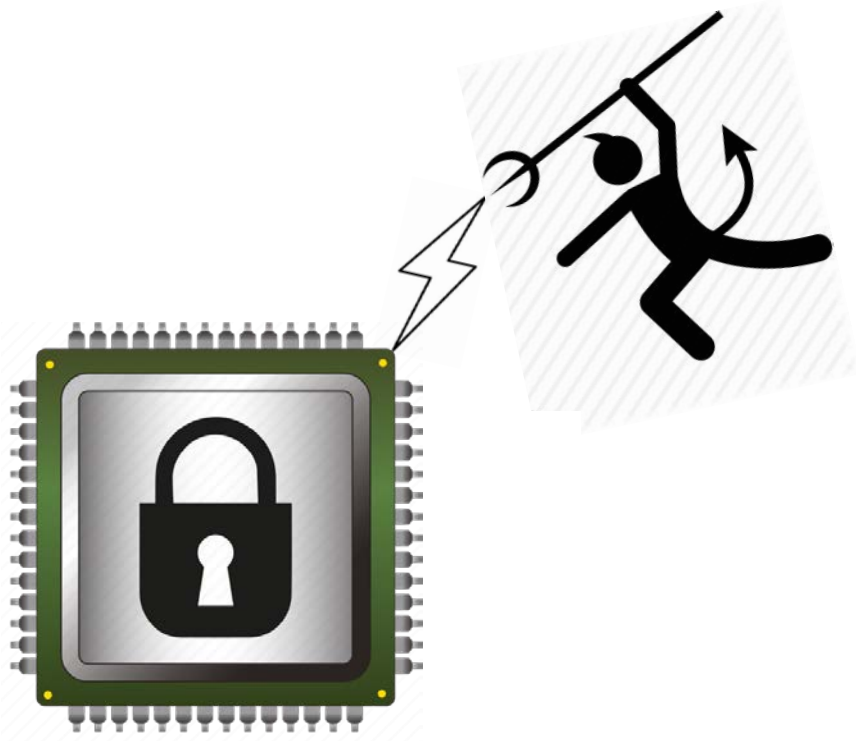
Crypto-implementations in real life leak sensitive information:

- Side-Channel Attacks (SCA)

Motivation

Crypto-implementations in real life leak sensitive information:

- Side-Channel Attacks (SCA)
- Fault Attacks (FA)



Motivation

Crypto-implementations in real life leak sensitive information:

- Side-Channel Attacks (SCA)
- Fault Attacks (FA)

- Countermeasures are required

Motivation

Crypto-implementations in real life leak sensitive information:

- Side-Channel Attacks (SCA)
- Fault Attacks (FA)

- Countermeasures are required
  - Designing countermeasure is tricky
  - Individual efforts for SCA and FA
  - Several failure stories

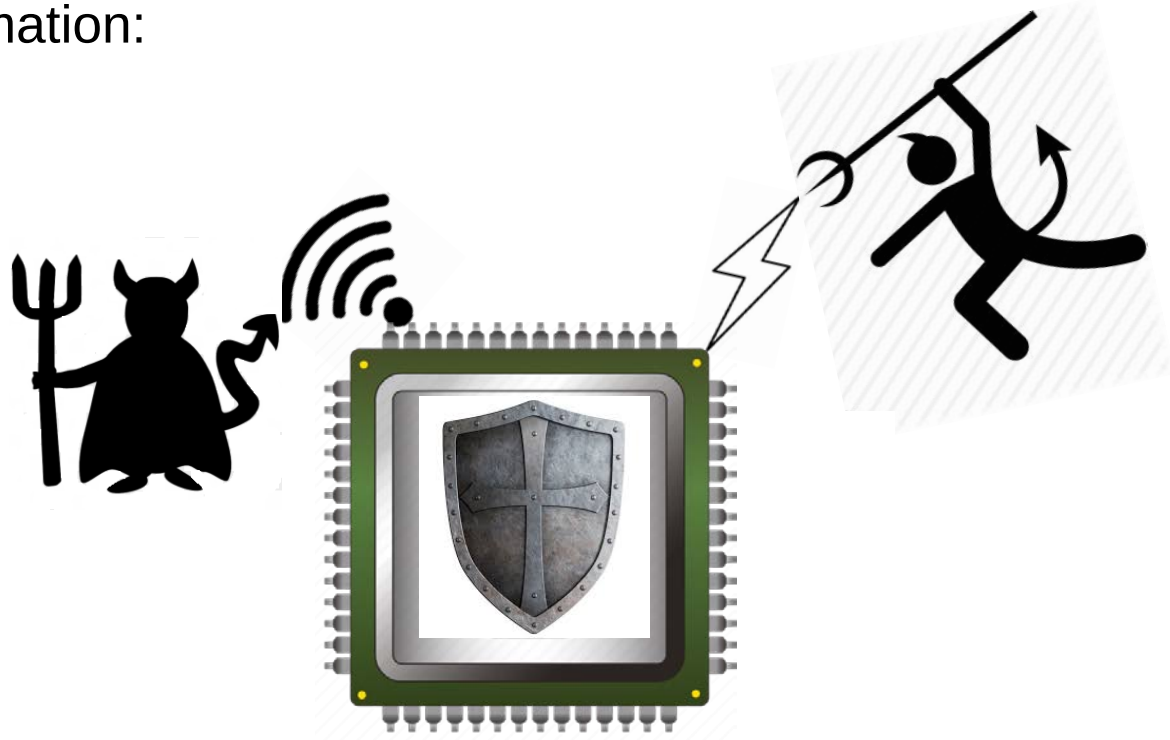
Motivation

Crypto-implementations in real life leak sensitive information:

- Side-Channel Attacks (SCA)
- Fault Attacks (FA)

- Countermeasures are required.
  - Designing countermeasure is tricky
  - Individual efforts for SCA and FA
  - Several failure stories

<u>What if both adversaries are present together?</u>

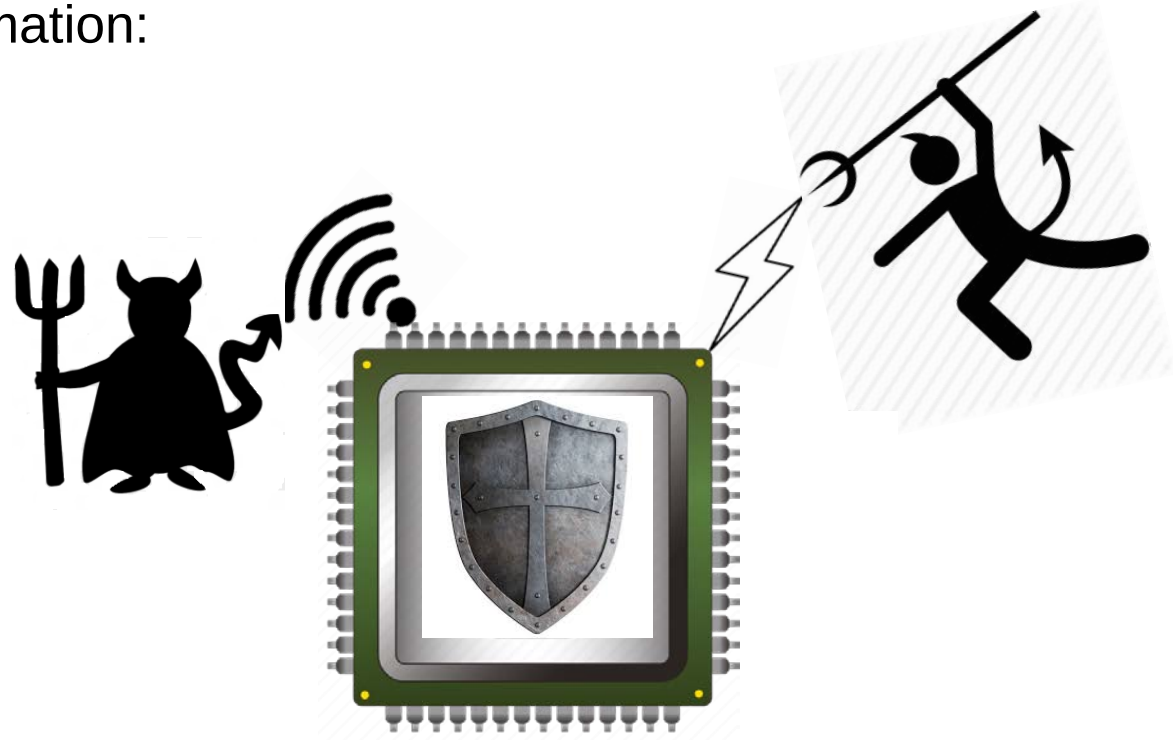


Motivation

Crypto-implementations in real life leak sensitive information:

- Side-Channel Attacks (SCA)
- Fault Attacks (FA)

- Countermeasures are required.
  - Designing countermeasure is tricky
  - Individual efforts for SCA and FA
  - Several failure stories

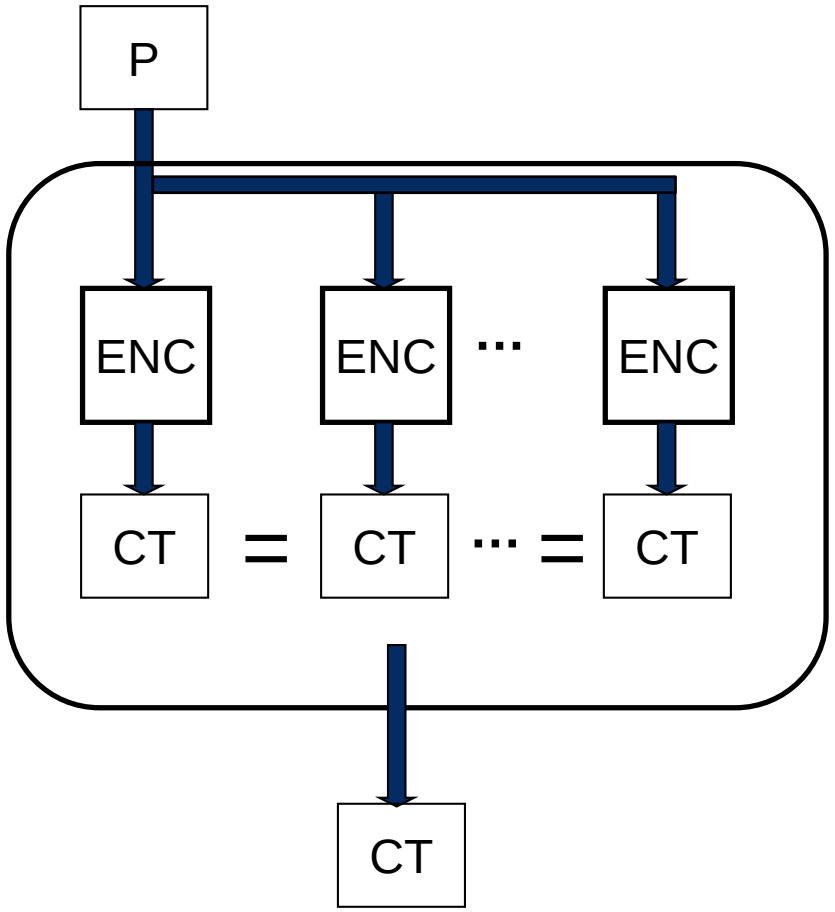


<u>What if both adversaries are present together?</u>

- Quite feasible with modern attack setups
- Relatively less explored
- What happens to the countermeasures?

FA Countermeasures

- Exploiting :
  - Time/space/information redundancy
- If fault detected:
  - No ciphertext / random ciphertext
- Redundancy at every round



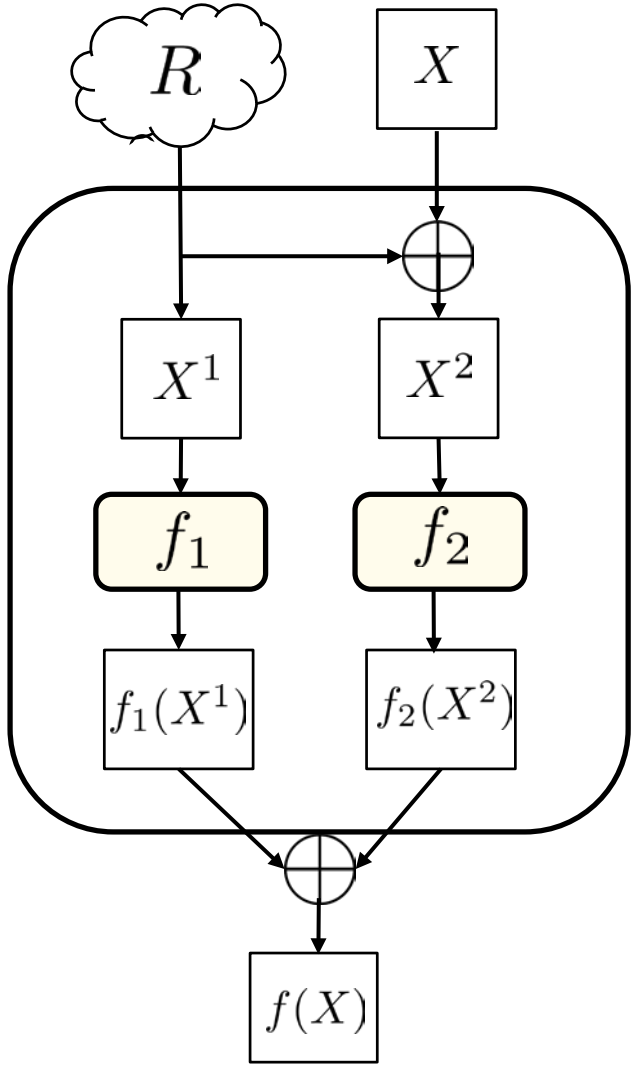
SCA Countermeasures

Masking:

- Makes power consumption independent of processed data
- Requires fresh randomness at every execution
- Split a value X into multiple shares $\langle X^1, X^2, \cdots, X^n \rangle$ such that $X^1 + X^2 + \cdots + X^n = X$
- Function $f(X)$ is split into functions $\langle f_1, f_2, \cdots, f_n \rangle$ such that $f_1(X^1) + f_2(X^2) + \cdots + f_n(X^n) = f(X)$
- Splitting is trivial for linear functions
- Nonlinear functions require special attention

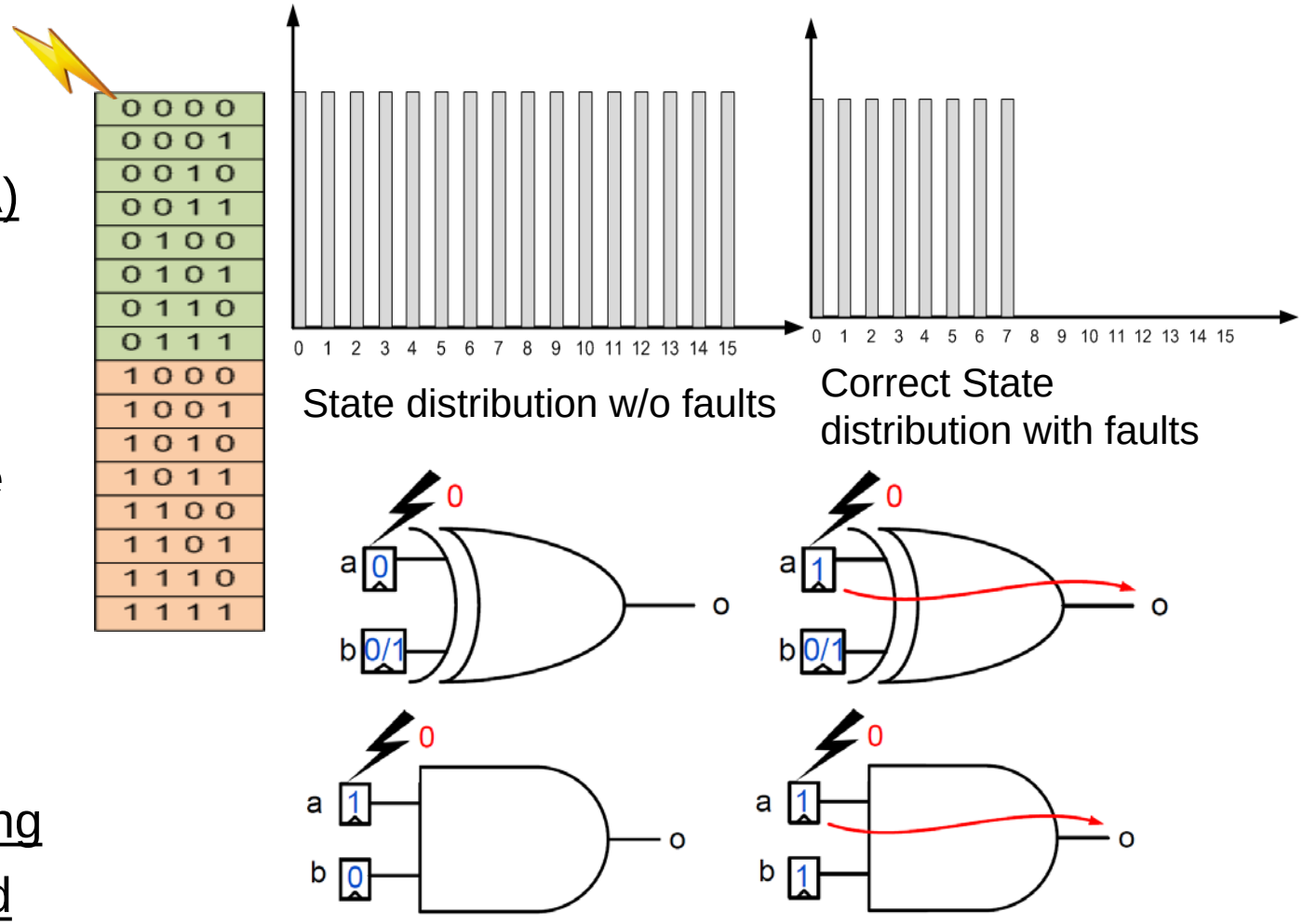
Hiding, Shuffling:

- Randomly alters program sequence without changing end result



Ineffective Faults and Countermeasures

- Attacks using correct responses from a fault campaign:
  - Statistical Ineffective Fault Attacks (SIFA)
  - Fault Template Attacks (FTA)
- Bypasses existing FA countermeasures and masking
  - Correctness information adds bias in the intermediate state
  - Correctness information can expose intermediate state altogether
- Countermeasures
  - Fine-grained error-correction and masking
  - Error detection on specially implemented masking schemes (e.g. Toffoli gates)



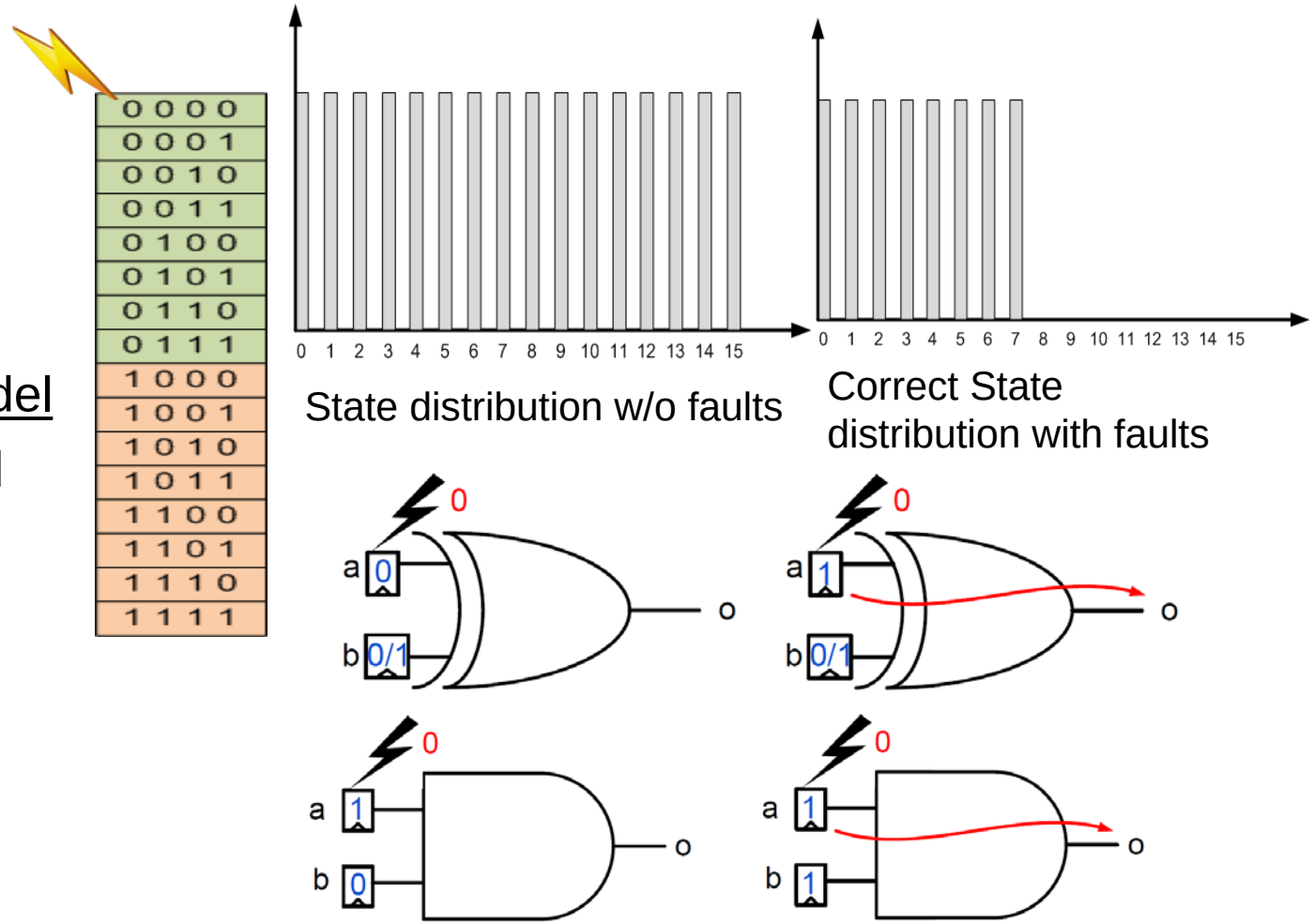
C. Dobraunig. M. Eichlseder, T. Korak, S. Mangard, F. Mendel, R. Primas, "SIFA: Exploiting Ineffective Fault Inductions on Symmetric Cryptography,.". *IACR TCHES* 2018

C. Dobraunig. M. Eichlseder, H. Gross, S. Mangard, F. Mendel, R. Primas, Statistical Ineffective Fault Attacks on Masked AES with Fault Countermeasures,.". *IACR ASIACRYPT* 2018

S. Saha, A. Bag, D.B. Roy, S. Patranabis, D. Mukhopadhyay, "Fault Template Attacks on Block Ciphers Exploiting Fault Propagation,.". *IACR EUROCRYPT* 2020

Ineffective Faults and Countermeasures

- Countermeasures against SIFA exploits both error detection/correction and most of them employ masking
  - Masking prevents some SIFA attacks
- The countermeasures employed makes combined SCA+FA a potential adversary model
- Can the existing proposal prevents combined attacks?
- Or we need something more?



Our Proposal: SCA-FTA

- SCA-enhanced Fault Template Attack
- Profiled technique:
  - Requires offline template building
- Bypasses some recent SIFA countermeasures
- Exposes few intricacies associated with masking while realizing SIFA countermeasures
- Middle round attacks
- No ciphertext access; no direct plaintext access

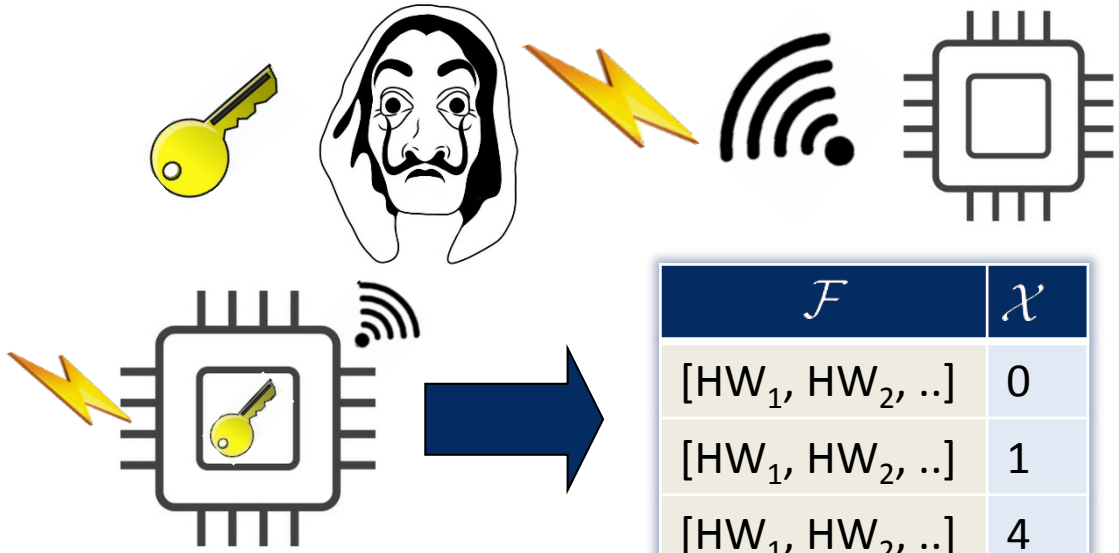
Attack	Breaks Simple SCA-FA Countermeasure	Breaks SCA-SIFA Countermeasure	Remarks
SIFA	✓	✗	No middle round attack
FTA	✓	✗	Middle round attack possible
SCA-FTA	✓	✓	**Middle round attack possible**

SCA-Enhanced Fault Template Attack (SCA-FTA)

Two Phases:

- Template Building (Offline):
  - Attacker has full access to a device similar to the target
    - Knows the key and the mask.
  - Constructs a template $\mathcal{T} : \mathcal{S}_{\mathcal{F}} \rightarrow \mathcal{X}$
    - $\mathcal{S}_{\mathcal{F}}$: Set containing the outcomes of some function over attacker observable SCA signal per fault location
    - $\mathcal{X}$: Part of the secret

Template Building



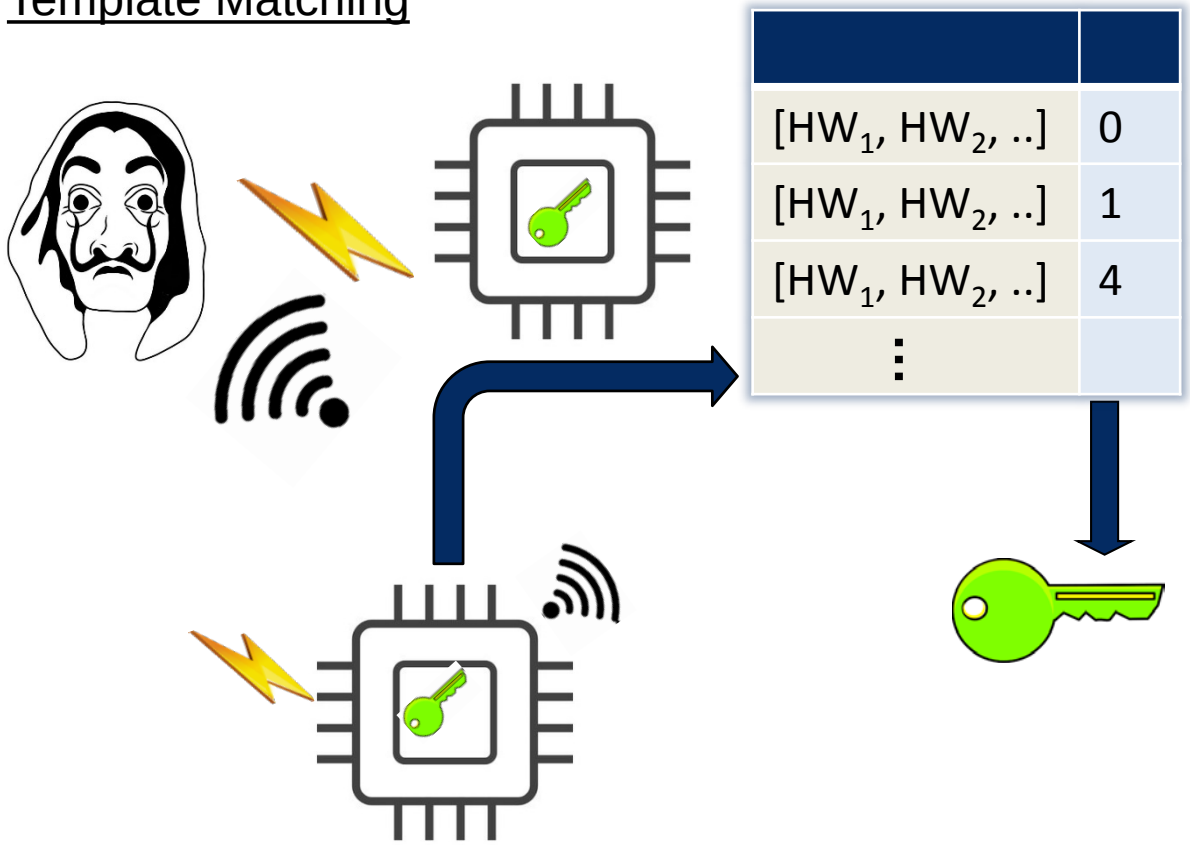
$\mathcal{F}$	$\mathcal{X}$
[HW_1, HW_2, ..]	0
[HW_1, HW_2, ..]	1
[HW_1, HW_2, ..]	4
⋮	⋮

SCA-Enhanced Fault Template Attack (SCA-FTA)

Two Phases:

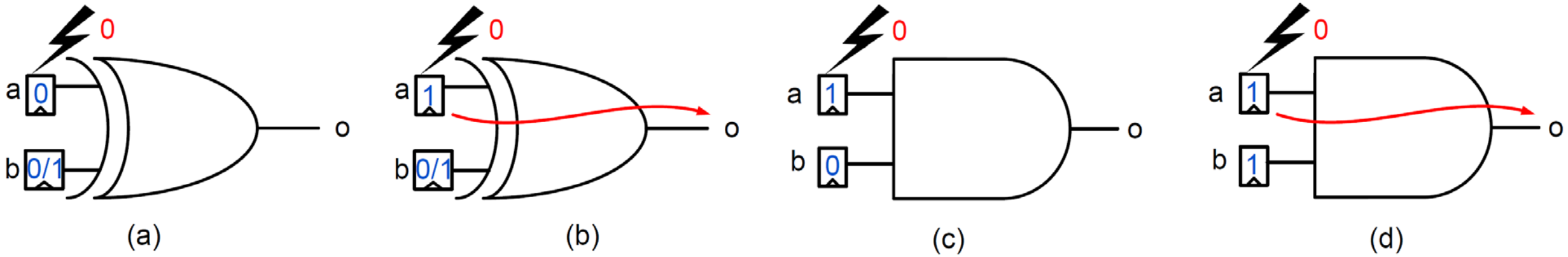
- Template Building (Offline):
  - Attacker has full access to a device similar to the target
    - Knows the key and the mask.
  - Constructs a template $\mathcal{T} : \mathcal{S}_{\mathcal{F}} \rightarrow \mathcal{X}$
    - $\mathcal{S}_{\mathcal{F}}$: Set containing the outcomes of some function over attacker observable SCA signal per fault location.
    - $\mathcal{X}$: Part of the secret
- Template Matching (Online):
  - Attacker injects faults at some predefined locations, measures SCA leakage and extract secret by using the (SCA) template

Template Matching



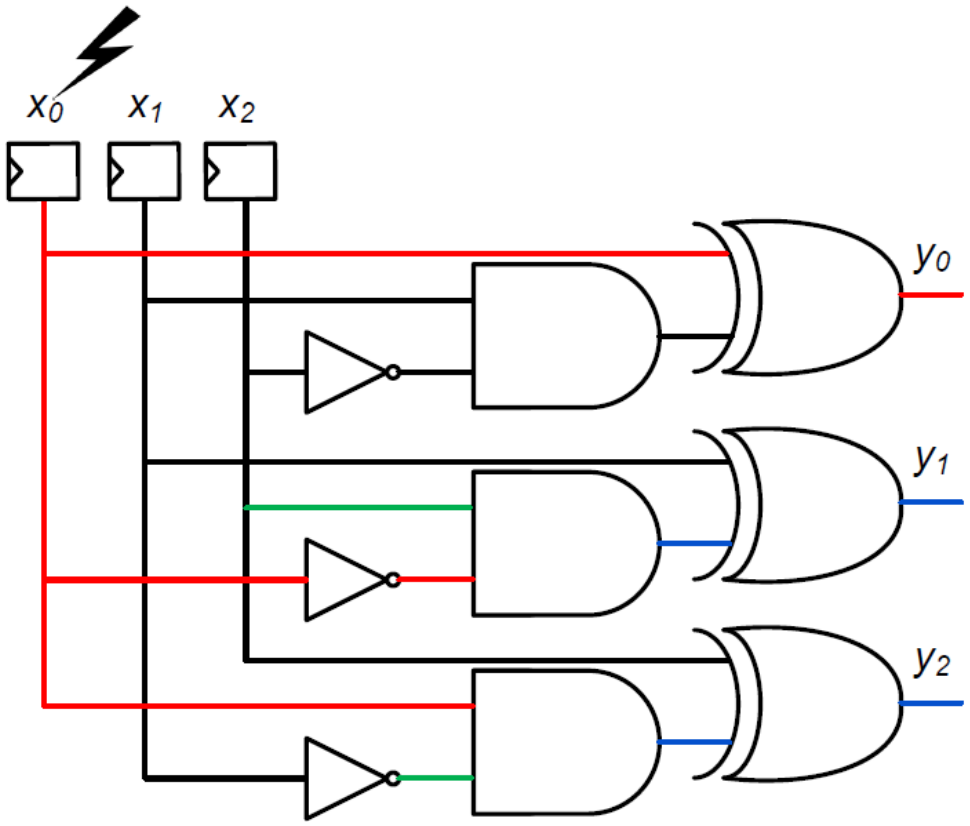
The model is similar to FTA, but can use some additional information other than correct/incorrect, due to SCA

Fault Propagation and Error Detection



- Fault activation and propagation through gates are data dependent
- If some circuit changes its activity depending on the presence of faults, it leads to SCA leakage
  - Error detection/ correction modules

Fault Propagation and Error Detection



- Fault propagation through S-Boxes leaks the S-Box input
- Error detection logic at the end of S-Boxes may leak the fault propagation information

Fault Pattern vs State

y_0	y_1	y_2	Input
F	C	F	(0,4)
F	F	F	(1,5)
F	C	C	(2,6)
F	F	C	(3,7)

Hamming weight of fault patterns vs State

HW	State
1	(2,6)
2	(0,4,3,7)
3	(1,5)

SCA-FTA on FA-Protected PRESENT

$$y_0 = x_0x_1x_3 + x_0x_2x_3 + x_0 + x_1x_2x_3 + x_1x_2 + x_2 + x_3 + 1$$
$$y_1 = x_0x_1x_3 + x_0x_2x_3 + x_0x_2 + x_0x_3 + x_0 + x_1 + x_2x_3 + 1$$
$$y_2 = x_0x_1x_3 + x_0x_1 + x_0x_2x_3 + x_0x_2 + x_0 + x_1x_2x_3 + x_2$$
$$y_3 = x_0 + x_1x_2 + x_1 + x_3$$

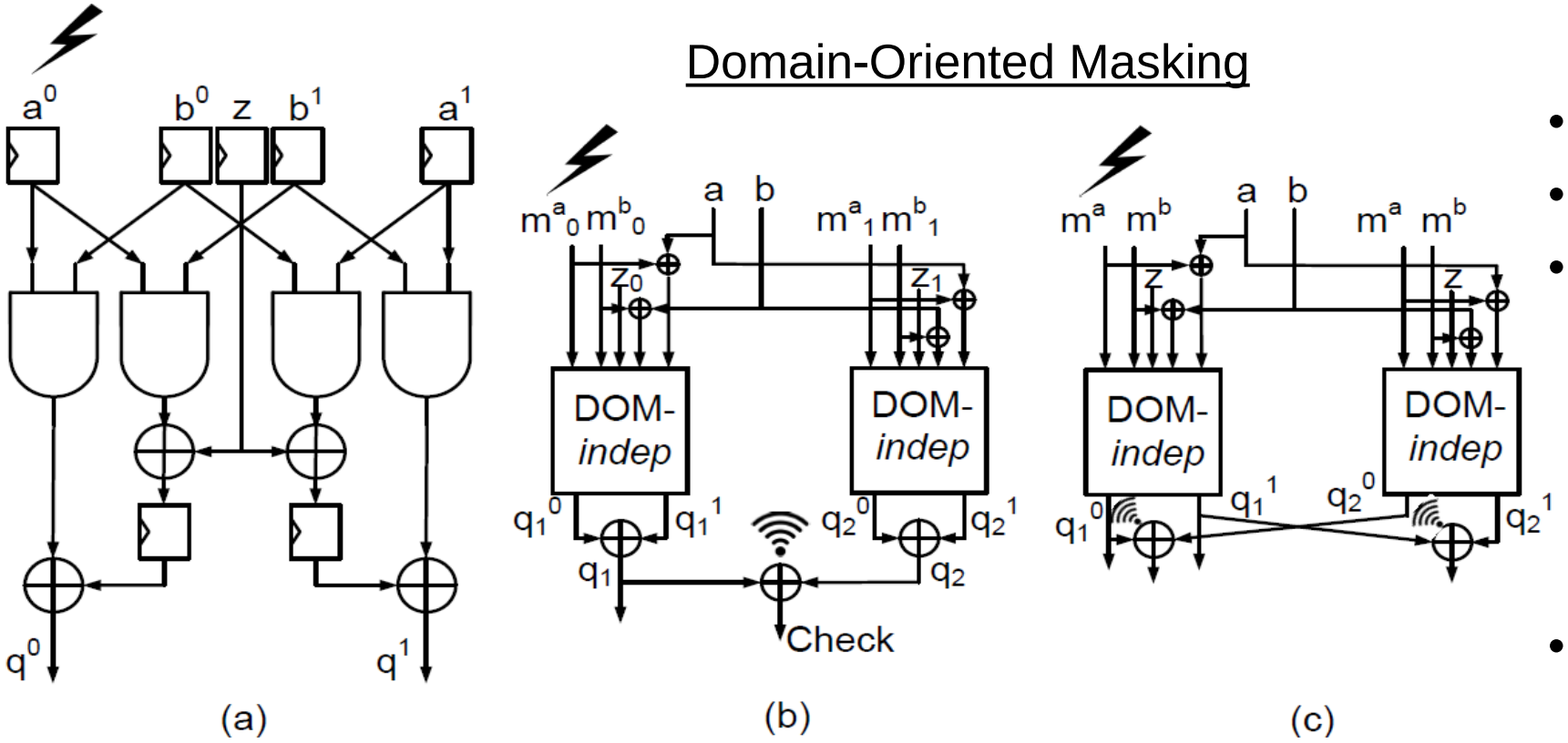
- Faults are injected at S-Box input
- Bit flip/bit stuck-at
- SCA leakage from XOR-based error detection (represented as HWs)
- One fault location and one injection per execution
- No ciphertext access, plaintext fixed
- Can be performed on middle rounds if there is error detection at each round
- Roughly 11,000 traces for state recovery in practical setups

HW (noise-free) vs state values for error-detection

fl_1	fl_2	fl_2	fl_3	State
2.0	3.0	2.0	3.0	(10)
4.0	2.0	2.0	2.0	(0, 15)
3.0	3.0	2.0	2.0	(12, 14)
4.0	2.0	3.0	3.0	(7)
2.0	2.0	3.0	3.0	(3)
3.0	2.0	2.0	2.0	(4, 13)
4.0	3.0	2.0	3.0	(8)
3.0	2.0	3.0	2.0	(1,5)
3.0	2.0	2.0	3.0	(6, 9)
2.0	2.0	2.0	3.0	(2, 11)

$$fl_1 = x_0 \quad fl_3 = x_2$$
$$fl_2 = x_1 \quad fl_4 = x_3$$

SCA-FTA on Masking and Error-Detection



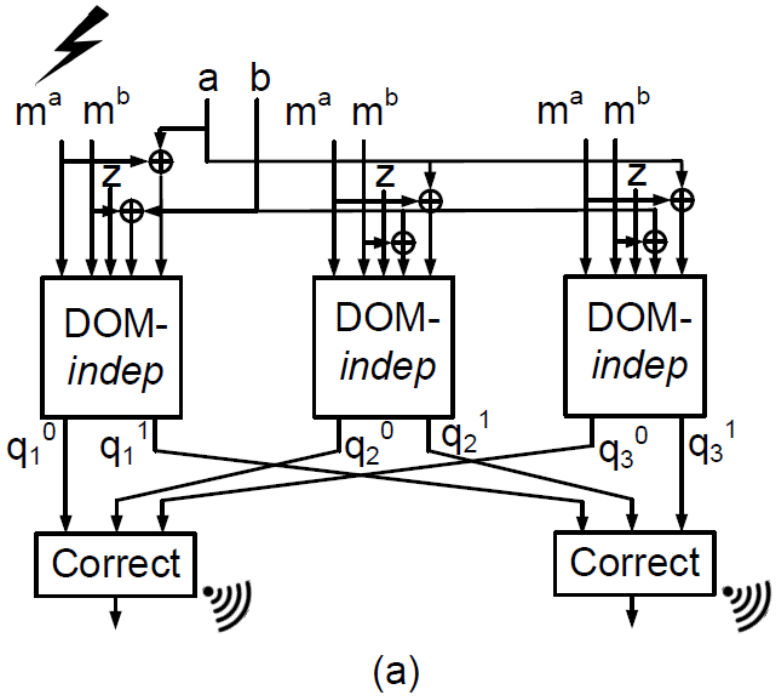
- Detection after unmasking
- Detection on shares
- Need to consider the order of SCA leakage
  - An attack is only *efficient* if leakage order is less than the masking order
- Schemes like PINI also shows similar leakage

$$q^0 = a^0 b^0 + (a^0 b^1 + z)$$

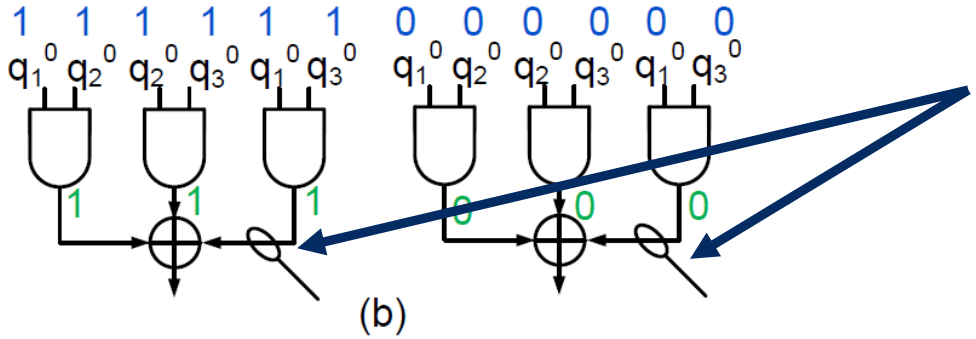
$$q^1 = a^1 b^1 + (a^1 b^0 + z)$$

- Error in DOM only propagates through one output share
  - Enables first-order SCA for DOM of any order
- The output share q^0 is faulty only if $b^0 + b^1 = b$ is 1

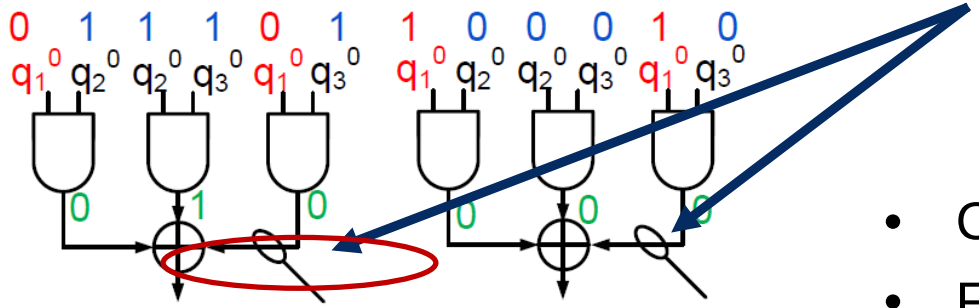
SCA-FTA on Masking and Error-Correction on Shares



(a)



(b)



(c)

Toggles randomly

Fixed at zero

- Correction on shares
- First-order SCA leakage for correction logic
- Applicable to other correction logics too if correction is detectable

SCA-FTA on SIFA Countermeasures

Toffoli-based Countermeasure

- Masked S-Boxes are implemented in a way so that every fault propagates to at least one output wire
- Error correction at the end of encryption
- Our observation: Although there is a mandatory fault propagation there also exists ineffective fault propagations
- Such ineffective propagations leak when detected
- No faulty ciphertext required
- First-order SCA leakage

Algorithm 1 *SIFA-PROTECTED* χ_3

Input: $(a^0, a^1, b^0, b^1, c^0, c^1)$
Output: $(r^0, r^1, s^0, s^1, t^0, t^1)$

1: $R_s \leftarrow R_r^{'} + R_t^{'}$
2: $T_0 \leftarrow \overline{b^{0'}} c^{1'}$; $T_2 \leftarrow a^{1'} b^{1'}$
3: $T_1 \leftarrow \overline{b^{0'}} c^{0'}$; $T_3 \leftarrow a^{1'} b^{0'}$
4: $r^0 \leftarrow T_0 + R_r^{'}$; $t^1 \leftarrow T_2 + R_t^{'}$
5: $r^0 \leftarrow r^0 + T_1$; $t^1 \leftarrow t^1 + T_3$
6: $T_0 \leftarrow \overline{c^{0'}} a^{1'}$; $T_2 \leftarrow b^{1'} c^{1'}$
7: $T_1 \leftarrow \overline{c^{0'}} a^{0'}$; $T_3 \leftarrow b^{1'} c^{0'}$
8: $s^0 \leftarrow T_0 + R_s^{'}$; $r^1 \leftarrow T_2 + R_r$
9: $s^0 \leftarrow s^0 + T_1$; $r^1 \leftarrow r^1 + T_3$
10: $T_0 \leftarrow \overline{a^{0'}} b^{1'}$; $T_2 \leftarrow c^{1'} a^{1'}$
11: $T_1 \leftarrow \overline{a^{0'}} b^{0'}$; $T_3 \leftarrow c^{1'} a^{0'}$
12: $t^0 \leftarrow T_0 + R_t$; $s^1 \leftarrow T_2 + R_s$
13: $t^0 \leftarrow t^0 + T_1$; $s^1 \leftarrow s^1 + T_3$
14: $r^0 \leftarrow r^0 + a^0$; $t^1 \leftarrow t^1 + c^1$
15: $s^0 \leftarrow s^0 + b^0$; $r^1 \leftarrow r^1 + a^1$
16: $t^0 \leftarrow t^0 + c^0$; $s^1 \leftarrow s^1 + b^1$
17: **Return** $(r^0, r^1, s^0, s^1, t^0, t^1)$

J. Daemen , C. Dobraunig. M. Eichlseder, H. Gross, F. Mendel, R. Primas “Protecting against Statistical Ineffective Fault Attacks.”. *IACR TCHES* 2020

SCA-FTA on SIFA Countermeasures

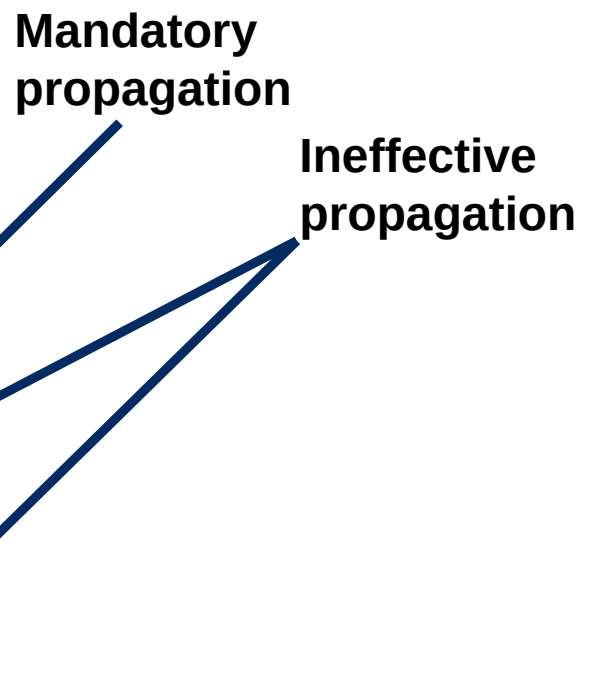
Toffoli-based Countermeasure

- Masked S-Boxes are implemented in a way so that every fault propagates to at least one output wire
- Error correction at the end of encryption
- Our observation: Although there is a mandatory fault propagation there also exists ineffective fault propagations
- Such ineffective propagations leak when detected
- No faulty ciphertext required
- First-order SCA leakage

Algorithm 1 *SIFA-PROTECTED* χ_3

Input: $(a^0, a^1, b^0, b^1, c^0, c^1)$
Output: $(r^0, r^1, s^0, s^1, t^0, t^1)$

1: $R_s \leftarrow R_r' + R_t'$
2: $T_0 \leftarrow \overline{b^{0'}} c^{1'}$; $T_2 \leftarrow a^{1'} b^{1'}$
3: $T_1 \leftarrow \overline{b^{0'}} c^{0'}$; $T_3 \leftarrow a^{1'} b^{0'}$
4: $r^0 \leftarrow T_0 + R_r'$; $t^1 \leftarrow T_2 + R_t'$
5: $r^0 \leftarrow r^0 + T_1$; $t^1 \leftarrow t^1 + T_3$
6: $T_0 \leftarrow \overline{c^{0'}} a^{1'}$; $T_2 \leftarrow b^{1'} c^{1'}$
7: $T_1 \leftarrow \overline{c^{0'}} a^{0'}$; $T_3 \leftarrow b^{1'} c^{0'}$
8: $s^0 \leftarrow T_0 + R_s'$; $r^1 \leftarrow T_2 + R_r$
9: $s^0 \leftarrow s^0 + T_1$; $r^1 \leftarrow r^1 + T_3$
10: $T_0 \leftarrow \overline{a^{0'}} b^{1'}$; $T_2 \leftarrow c^{1'} a^{1'}$
11: $T_1 \leftarrow \overline{a^{0'}} b^{0'}$; $T_3 \leftarrow c^{1'} a^{0'}$
12: $t^0 \leftarrow T_0 + R_t$; $s^1 \leftarrow T_2 + R_s$
13: $t^0 \leftarrow t^0 + T_1$; $s^1 \leftarrow s^1 + T_3$
14: $r^0 \leftarrow r^0 + a^0$; $t^1 \leftarrow t^1 + c^1$
15: $s^0 \leftarrow s^0 + b^0$; $r^1 \leftarrow r^1 + a^1$
16: $t^0 \leftarrow t^0 + c^0$; $s^1 \leftarrow s^1 + b^1$
17: **Return** $(r^0, r^1, s^0, s^1, t^0, t^1)$



J. Daemen , C. Dobraunig. M. Eichlseder, H. Gross, F. Mendel, R. Primas “Protecting against Statistical Ineffective Fault Attacks.”. *IACR TCHES* 2020

SCA-FTA on SIFA Countermeasures

Toffoli-based Countermeasure

- Masked S-Boxes are implemented in a way so that every fault propagates to at least one output wire
- Error correction at the end of encryption
- Our observation: Although there is a mandatory fault propagation there also exists ineffective fault propagations
- Such ineffective propagations leak when detected
- No faulty ciphertext required
- First-order SCA leakage

Algorithm 1 *SIFA-PROTECTED* χ_3

Input: $(a^0, a^1, b^0, b^1, c^0, c^1)$
Output: $(r^0, r^1, s^0, s^1, t^0, t^1)$

1: $R_s \leftarrow R_r' + R_t'$
2: $T_0 \leftarrow \overline{b^{0'}} c^{1'}$; $T_2 \leftarrow a^{1'} b^{1'}$
3: $T_1 \leftarrow \overline{b^{0'}} c^{0'}$; $T_3 \leftarrow a^{1'} b^{0'}$
4: $r^0 \leftarrow T_0 + R_r'$; $t^1 \leftarrow T_2 + R_t'$
5: $r^0 \leftarrow r^0 + T_1$; $t^1 \leftarrow t^1 + T_3$
6: $T_0 \leftarrow \overline{c^{0'}} a^{1'}$; $T_2 \leftarrow b^{1'} c^{1'}$
7: $T_1 \leftarrow \overline{c^{0'}} a^{0'}$; $T_3 \leftarrow b^{1'} c^{0'}$
8: $s^0 \leftarrow T_0 + R_s'$; $r^1 \leftarrow T_2 + R_r$
9: $s^0 \leftarrow s^0 + T_1$; $r^1 \leftarrow r^1 + T_3$
10: $T_0 \leftarrow a^{0'} b^{1'}$; $T_2 \leftarrow c^{1'} a^{1'}$
11: $T_1 \leftarrow \overline{a^{0'}} b^{0'}$; $T_3 \leftarrow c^{1'} a^{0'}$
12: $t^0 \leftarrow T_0 + R_t$; $s^1 \leftarrow T_2 + R_s$
13: $t^0 \leftarrow t^0 + T_1$; $s^1 \leftarrow s^1 + T_3$
14: $r^0 \leftarrow r^0 + a^0$; $t^1 \leftarrow t^1 + c^1$
15: $s^0 \leftarrow s^0 + b^0$; $r^1 \leftarrow r^1 + a^1$
16: $t^0 \leftarrow t^0 + c^0$; $s^1 \leftarrow s^1 + b^1$
17: **Return** $(r^0, r^1, s^0, s^1, t^0, t^1)$

Mandatory propagation

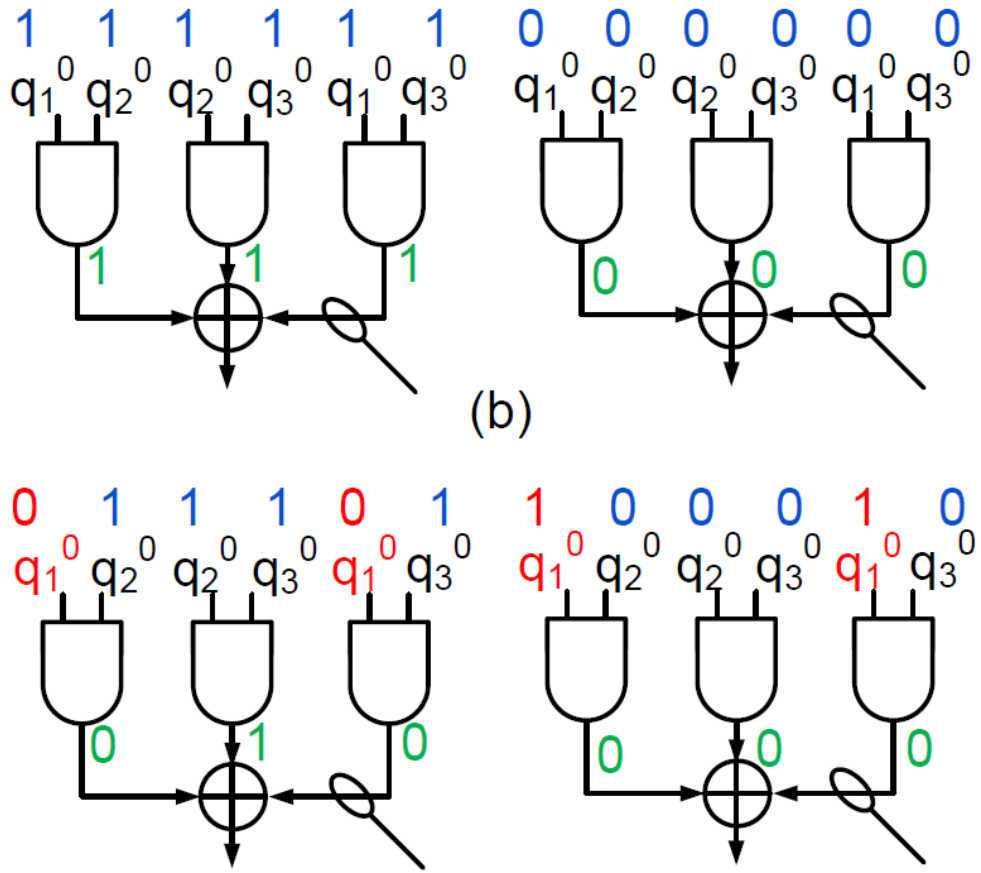
Ineffective propagation

J. Daemen , C. Dobraunig. M. Eichlseder, H. Gross, F. Mendel, R. Primas “Protecting against Statistical Ineffective Fault Attacks.”. *IACR TCHES* 2020

SCA-FTA on SIFA Countermeasures

Transform-and-Encode Framework

- Masking and error-correction on each share for SIFA protection
- Any masking scheme can be used
- Our observation: Vulnerable for DOM based masking schemes



S. Saha , D. Jap. D.B. Roy, S. Bhasin, D. Mukhopadhyay, "A Framework to Counter Statistical Ineffective Fault Analysis of Block Ciphers Using Domain Transformation and Error Correction,.". *IEEE TIFS* 2020

SCA-FTA on SIFA Countermeasures

Case of Higher-Order TI (HOTI)

- HOTI often involves *share compression* to manage the blowup of share counts
- Our observation: Share compression makes HOTI vulnerable

$$d^0 = c^1 + a^1b^1 + a^0b^1 + a^1b^0, \quad d^1 = c^2 + a^2b^2 + a^0b^2 + a^2b^0,$$
$$d^2 = c^3 + a^3b^3 + a^0b^3 + a^3b^0, \quad d^3 = c^0 + a^0b^0 + a^0b^4 + a^4b^0,$$
$$d^4 = c^4 + a^4b^4 + a^1b^4 + a^4b^1, \quad d^5 = a^1b^3 + a^3b^1,$$
$$d^6 = a^1b^2 + a^2b^1, \quad d^7 = a^2b^3 + a^2b^4 + a^3b^4,$$
$$d^8 = a^3b^2 + a^4b^2 + a^4b^3$$

Share Compression

$$e^0 = d^0 + d^5, \quad e^1 = d^1 + d^6, \quad e^2 = d^2 + d^7, \quad e^3 = d^3 + d^8, \quad e^4 = d^4$$

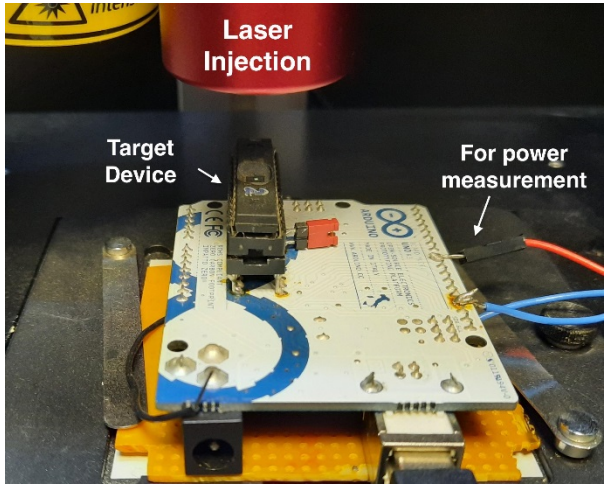
- Fault at a^4
- Probes at e^3 and e^4 (detection/correction operations)
- Leaks: $b^0 + b^1 + b^2 + b^3 + b^4 = b$
- Second-order attack on second-order secure TI

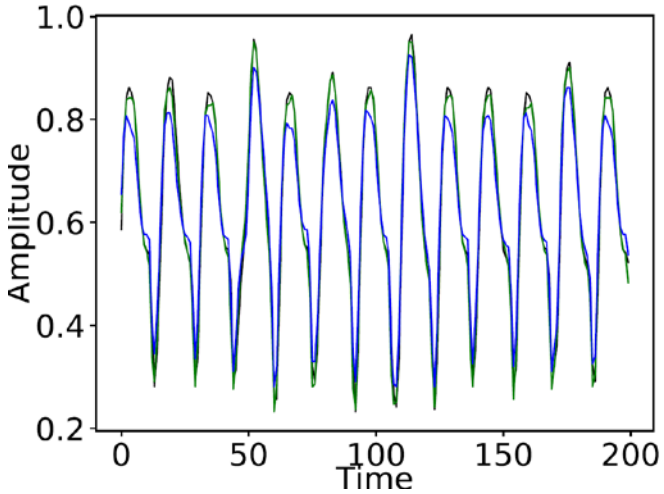
Practical Evaluation

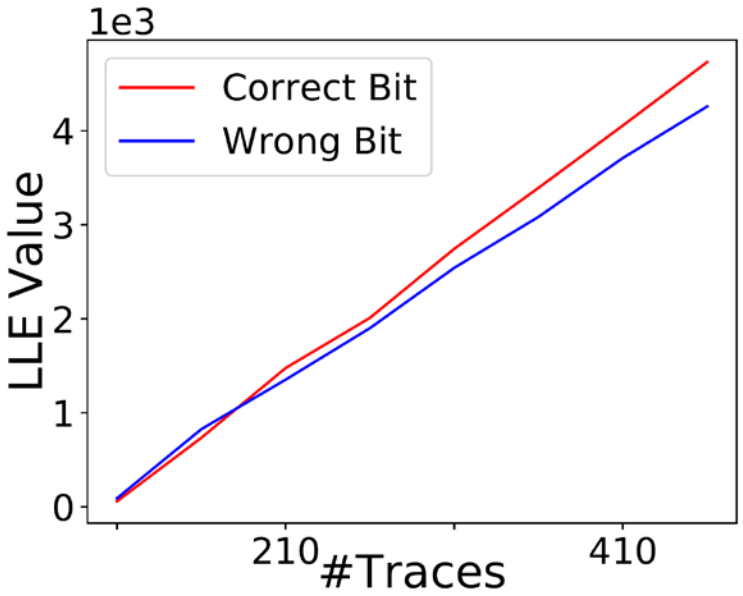
Experiment 1:

- Open-source SIFA-protected Keccak
- Target Platform: Atmega328P mounted on Arduino UNO
- Laser fault injection: 8W diode pulse laser
- Power measurement for SCA
- Detection operations
- SCA measurement clock cycle is 12 microsecond away from fault injection clock cycle
  - No noise due to injection
- 210-220 traces per fault location for unshared detection
- 320-350 for shared detection

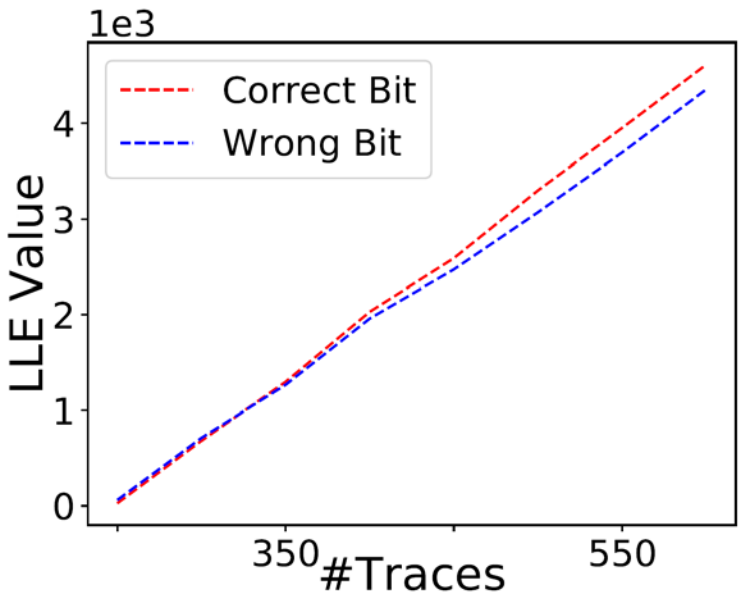
https://github.com/sifa-aux/countermeasures







Detection on Unshared Values

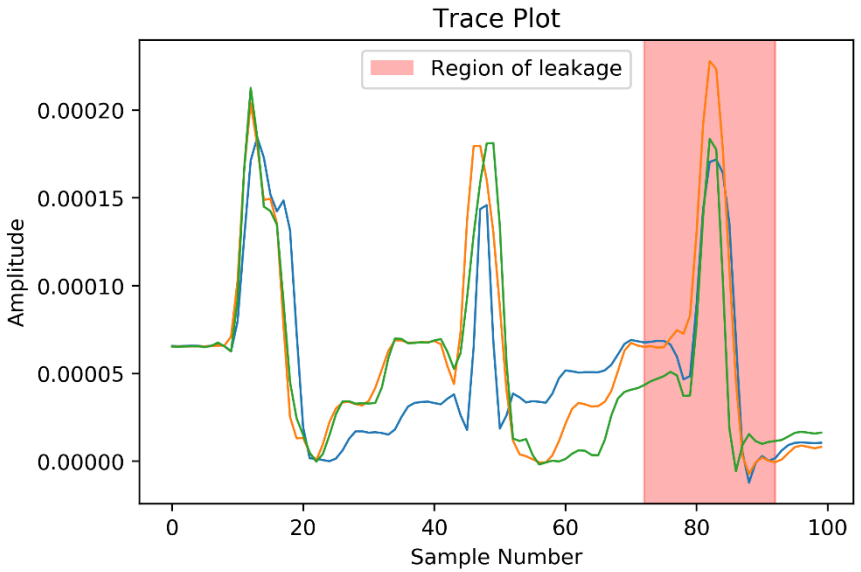


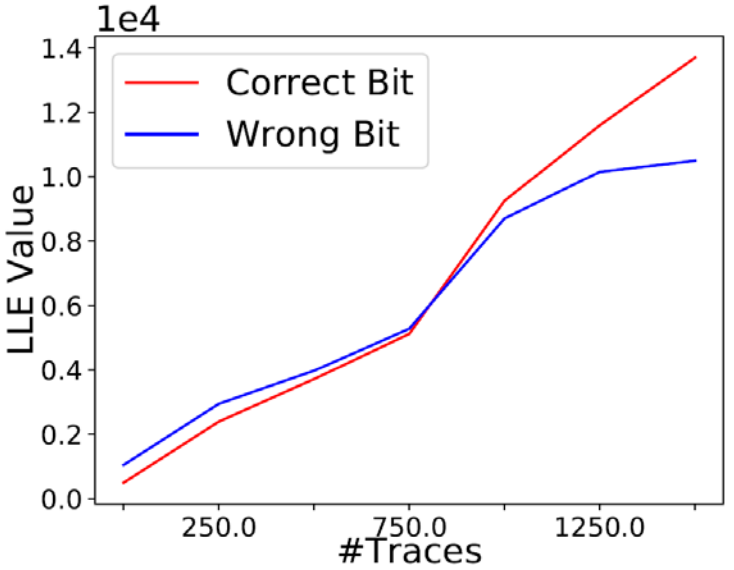
Detection on Shared Values

Practical Evaluation

Experiment 2:

- Hardware implementation of χ_3
- Simulated Leakage: Synopsys Design Compiler and PrimeTime
- Gate-level leakage simulation considering the noise due to adjacent gate switching
- Roughly 1000 required traces for each fault location





Correction on Shared Values

Potential Countermeasures

Observation for TI:

- Non-completeness plays a crucial role
- Error-detection/correction over non-complete paths maintains security
- The concrete implementation from TaE paper with first order TI on PRESENT is secure

Other Secure Primitives:

- CAPA
- NINA

$$f_{10} = x_1^2 + x_2^2 x_0^2 + x_2^2 x_0^3 + x_2^3 x_0^2$$
$$f_{20} = x_1^3 + x_2^3 x_0^3 + x_2^1 x_0^3 + x_2^3 x_0^1$$
$$f_{30} = x_1^1 + x_2^1 x_0^1 + x_2^1 x_0^2 + x_2^2 x_0^1$$

- A fault at x_0^2 leaks $x_2 = x_2^1 + x_2^2 + x_2^3$
- But the error propagation happens through f_{10} and f_{30}
- So we need to probe 2 locations for a first-order secure implementation, which is not an *efficient* attack

Conclusion

- Combined attacks are practical and should be considered for implementations having both FA and SCA countermeasures
- Having both countermeasures does not mean combined security
  - We combine FTA with SCA template attacks
  - Extremely powerful adversary
  - Breaks even SIFA countermeasures
  - Exposes certain critical properties of masking
- Potential future work:
  - Analyzing other countermeasures
    - Multi-party based
  - Finding lightweight countermeasures