Statistical Ineffective Fault Attacks on Masked AES with Fault Countermeasures

Christoph Dobraunig, Maria Eichlseder, Hannes Gross, Stefan Mangard, Florian Mendel, Robert Primas

ASIACRYPT 2018

IAIK - Graz University of Technology

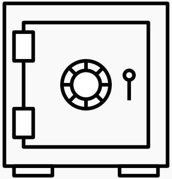
Motivation

Building cryptographic implementations is challenging:

Motivation

Building cryptographic implementations is challenging:

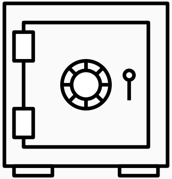
- Requires usage of proper cryptographic primitives



Motivation

Building cryptographic implementations is challenging:

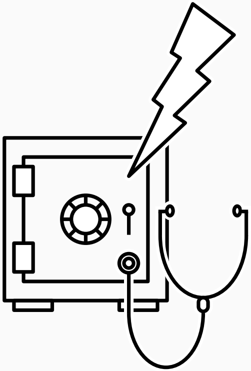
- Requires usage of proper cryptographic primitives
- But often also the usage of additional defenses ...
  - Microcontroller
  - FPGAs
  - ASICs



Motivation

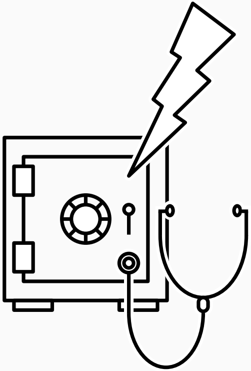
Building cryptographic implementations is challenging:

- Requires usage of proper cryptographic primitives
- But often also the usage of additional defenses ...
  - Microcontroller
  - FPGAs
  - ASICs
- ... because of implementation attacks



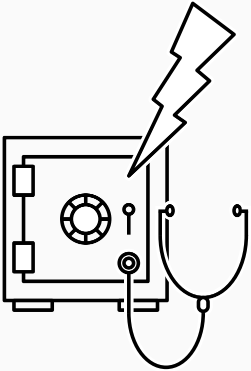
Motivation

- Proper cryptography does not mean practical security



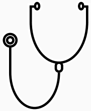
Motivation

- Proper cryptography does not mean practical security
- Every cryptographic implementation stores a secret

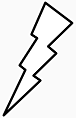


Motivation

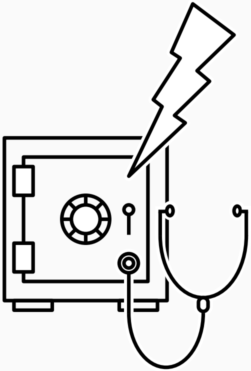
- Proper cryptography does not mean practical security
- Every cryptographic implementation stores a secret
- Secrets can be extracted by:



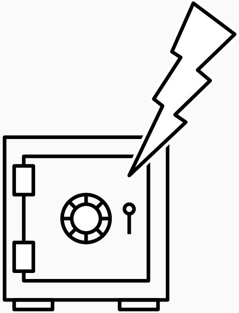
Power Analysis



Fault Attacks

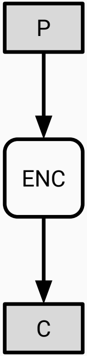


Fault Attacks



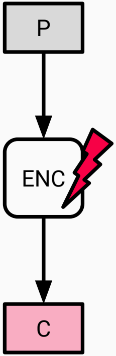
Fault Attacks

- Get physical access to target device:
  - Set plaintexts
  - Observe ciphertexts



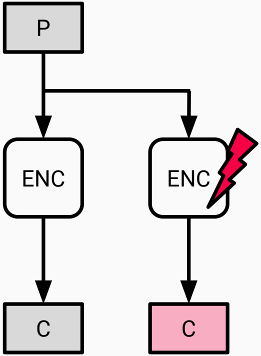
Fault Attacks

- Get physical access to target device:
  - Set plaintexts
  - Observe ciphertexts
- Cause erroneous computations via:
  - Clock glitches
  - Voltage glitches
  - Lasers



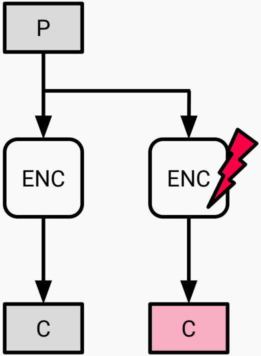
Fault Attacks

- Get physical access to target device:
  - Set plaintexts
  - Observe ciphertexts
- Cause erroneous computations via:
  - Clock glitches
  - Voltage glitches
  - Lasers
- Observe faulty and correct ciphertext



Fault Attacks

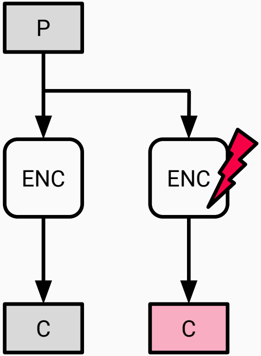
- Get physical access to target device:
  - Set plaintexts
  - Observe ciphertexts
- Cause erroneous computations via:
  - Clock glitches
  - Voltage glitches
  - Lasers
- Observe faulty and correct ciphertext
- Recover key



Fault Attacks

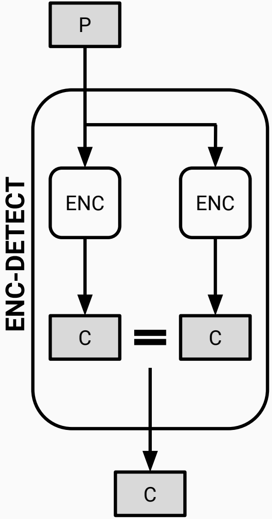
- Get physical access to target device:
  - Set plaintexts
  - Observe ciphertexts
- Cause erroneous computations via:
  - Clock glitches
  - Voltage glitches
  - Lasers
- Observe faulty and correct ciphertext
- Recover key

⇒ Differential Fault Attack (DFA)



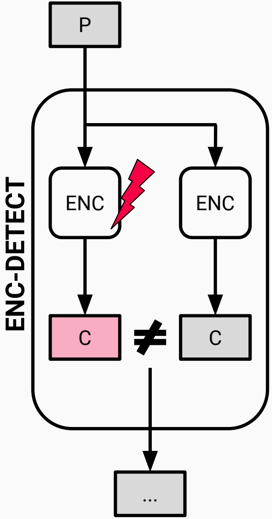
Fault Countermeasures - Detection

- Use redundancy to detect faults



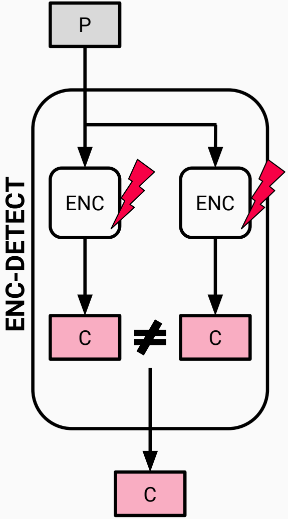
Fault Countermeasures - Detection

- Use redundancy to detect faults
- Fault detected → No ciphertext



Fault Countermeasures - Detection

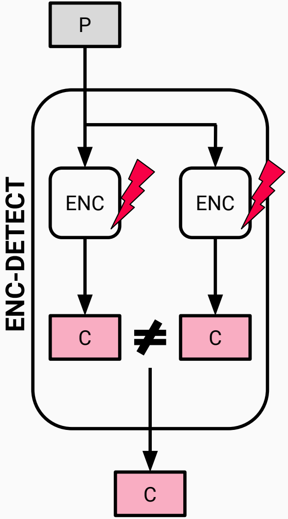
- Use redundancy to detect faults
- Fault detected → No ciphertext
- 2 identical faults necessary for attack



Fault Countermeasures - Detection

- Use redundancy to detect faults
- Fault detected → No ciphertext
- 2 identical faults necessary for attack

→ More redundancy, Enc-Dec, etc...



Statistical Ineffective Fault Attacks (SIFA)

- We presented SIFA at CHES 2018:
    - Breaks detection countermeasures (any degree of redundancy)
    - Breaks infection countermeasures
    - Requires just a single fault injection per encryption
    - Require no precise knowledge about location and effect of the fault

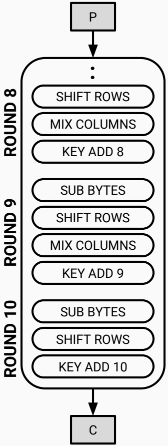
Statistical Ineffective Fault Attacks (SIFA)

- We presented SIFA at CHES 2018:
    - Breaks detection countermeasures (any degree of redundancy)
    - Breaks infection countermeasures
    - Requires just a single fault injection per encryption
    - Require no precise knowledge about location and effect of the fault
- We demonstrated applicability to AE schemes at SAC 2018

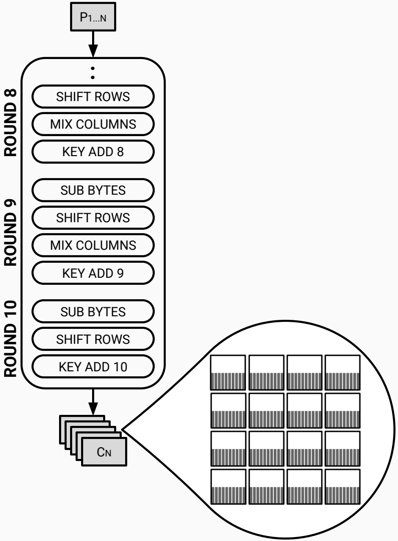
Statistical Ineffective Fault Attacks (SIFA)

- We presented SIFA at CHES 2018:
    - Breaks detection countermeasures (any degree of redundancy)
    - Breaks infection countermeasures
    - Requires just a single fault injection per encryption
    - Require no precise knowledge about location and effect of the fault
- We demonstrated applicability to AE schemes at SAC 2018
- What about power analysis countermeasures?

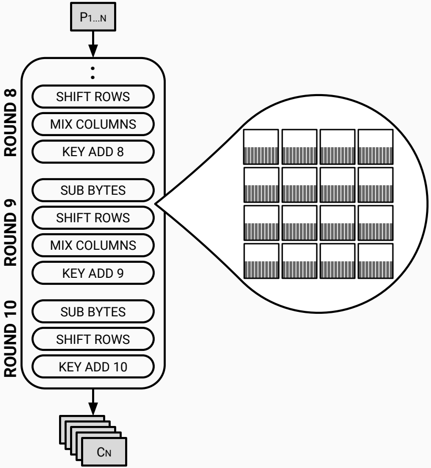
SIFA on AES in Pictures



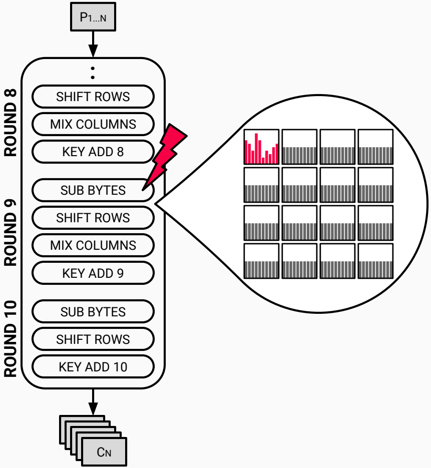
SIFA on AES in Pictures



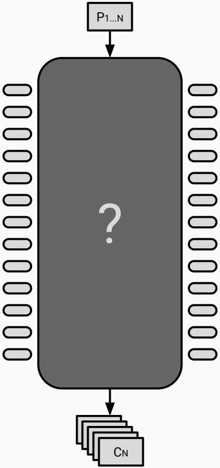
SIFA on AES in Pictures



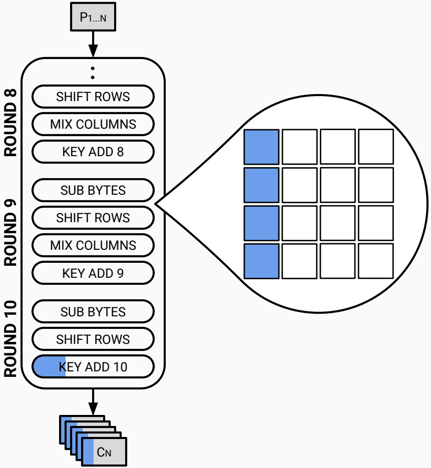
SIFA on AES in Pictures



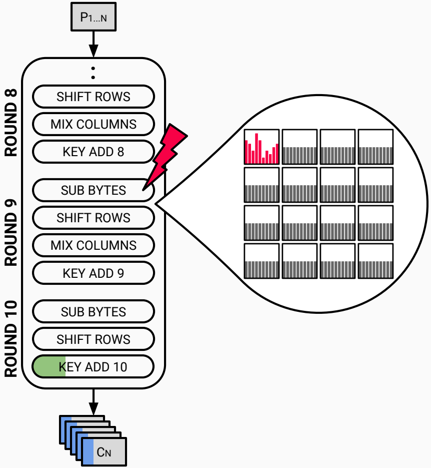
SIFA on AES in Pictures



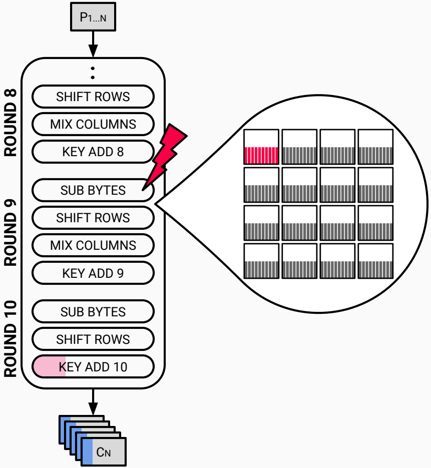
SIFA on AES in Pictures



SIFA on AES in Pictures

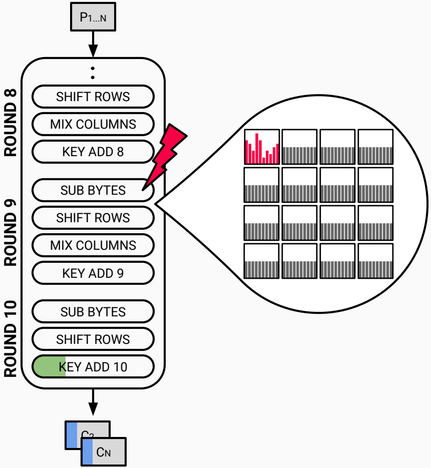


SIFA on AES in Pictures



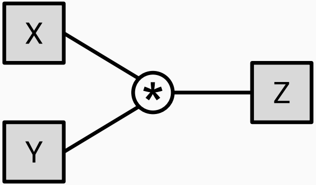
What about fault countermeasures?

SIFA in Pictures

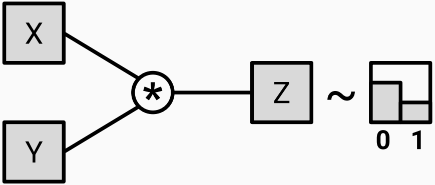


*only correct computations are considered

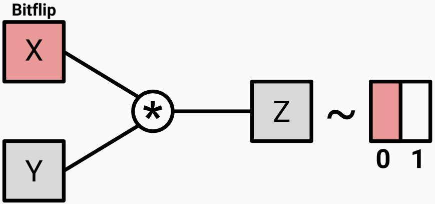
Ineffective Faults on AND



Ineffective Faults on AND



Ineffective Faults on AND

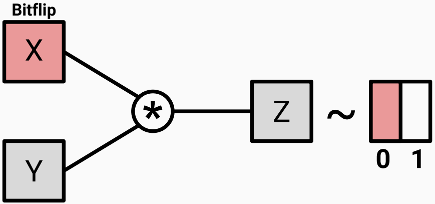


*only correct computations are considered

Ineffective Faults on AND

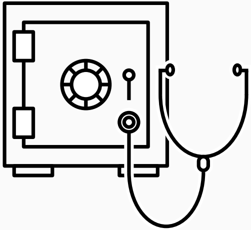
Also works with:

- Other instructions:
  LOAD, STORE, XOR
- Other fault types:
  Random, Stuck-at, Skip



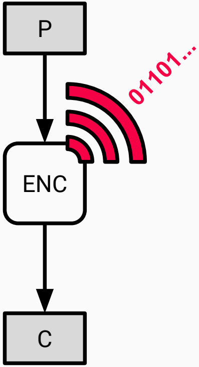
*only correct computations are considered

Power Analysis



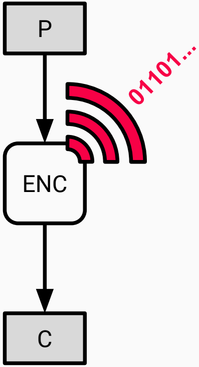
Power Analysis

- Circuits leak information via side-channels, e.g. power consumption



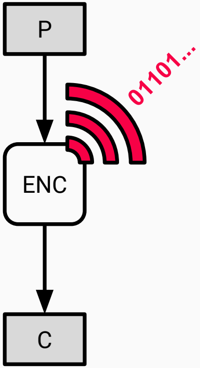
Power Analysis

- Circuits leak information via side-channels, e.g. power consumption
- CMOS circuits draw power almost only in case of “events”



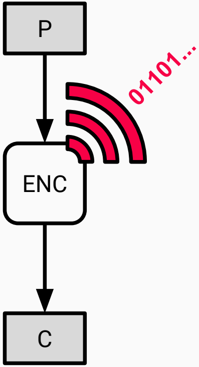
Power Analysis

- Circuits leak information via side-channels, e.g. power consumption
- CMOS circuits draw power almost only in case of “events”
- Correlation between processed data and power consumption



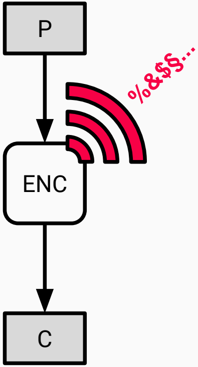
Power Analysis

- Circuits leak information via side-channels, e.g. power consumption
- CMOS circuits draw power almost only in case of “events”
- Correlation between processed data and power consumption
- Problematic if processed data contains secrets



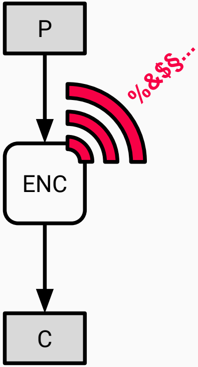
Power Analysis Countermeasures

- Make power consumption independent of processed data
  - Requires hardware support (filters, noise generators)



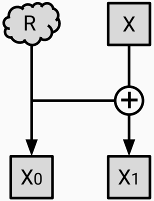
Power Analysis Countermeasures

- Make power consumption independent of processed data
  - – Requires hardware support (filters, noise generators)
- Make processed data independent of the actual data
  - + "Masking" can be done on algorithmic level



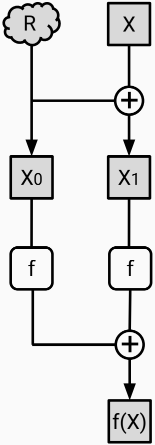
Masking Idea

- Split a value x into multiple “shares” s.t.:
  - The XOR-sum over all x_i equals x
  - The distribution of each x_i is independent from x



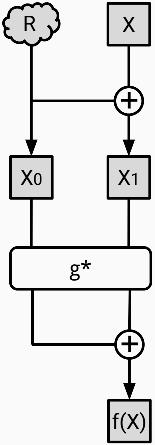
Masking Idea

- Split a value x into multiple "shares" s.t.:
  - The XOR-sum over all x_i equals x
  - The distribution of each x_i is independent from x
- Linear function f is performed separately



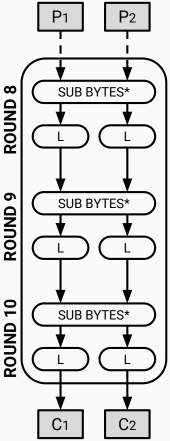
Masking Idea

- Split a value x into multiple “shares” s.t.:
  - The XOR-sum over all x_i equals x
  - The distribution of each x_i is independent from x
- Linear function f is performed separately
- Nonlinear functions g need more attention:
  - $g^\star$ works on all shares
  - $g^\star$ avoids direct combinations of shares



Masking Idea

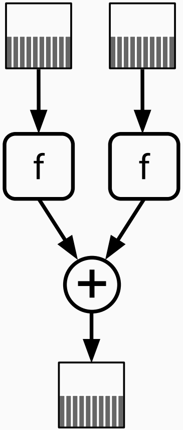
- Split a value x into multiple "shares" s.t.:
  - The XOR-sum over all x_i equals x
  - The distribution of each x_i is independent from x
- Linear function f is performed separately
- Nonlinear functions g need more attention:
  - $g^\star$ works on all shares
  - $g^\star$ avoids direct combinations of shares
- Applied to AES →



Does our attack still work?

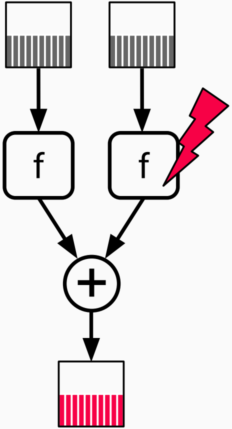
SIFA on Masked AES with Fault CM

- Faulting single shares in linear functions does not work...



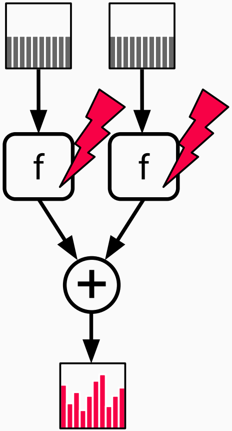
SIFA on Masked AES with Fault CM

- Faulting single shares in linear functions does not work...



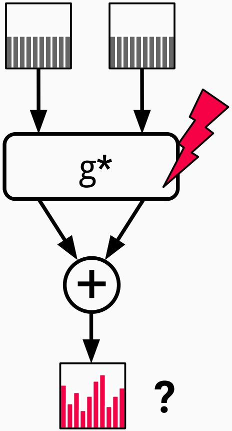
SIFA on Masked AES with Fault CM

- Faulting single shares in linear functions does not work...
- Faulting all shares would work but is boring...

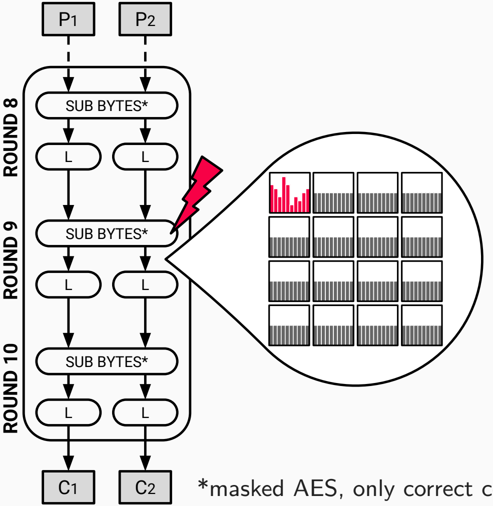


SIFA on Masked AES with Fault CM

- Faulting single shares in linear functions does not work...
- Faulting all shares would work but is boring...
- Can faulting single shares in non-linear functions lead to a bias in the unshared value?

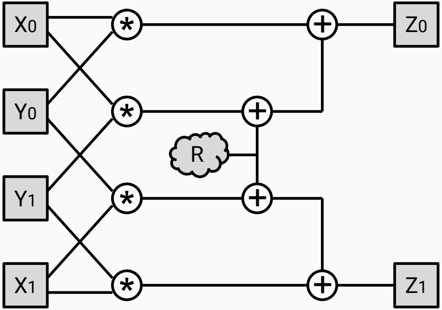


SIFA on Masked AES with Fault CM

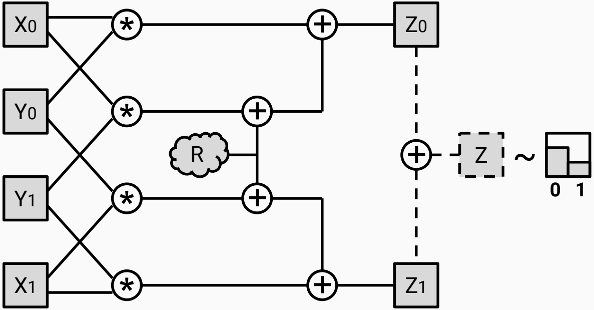


*masked AES, only correct computations are considered

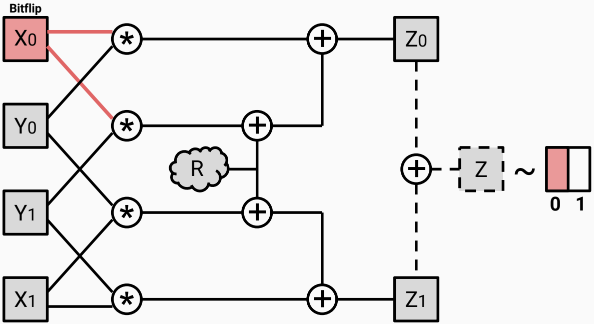
Faults on Masked AND



Faults on Masked AND

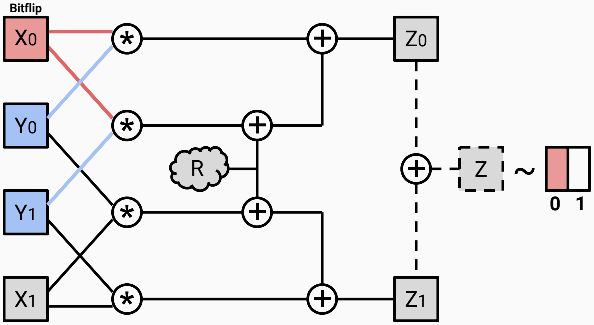


Faults on Masked AND



*only correct computations are considered

Faults on Masked AND

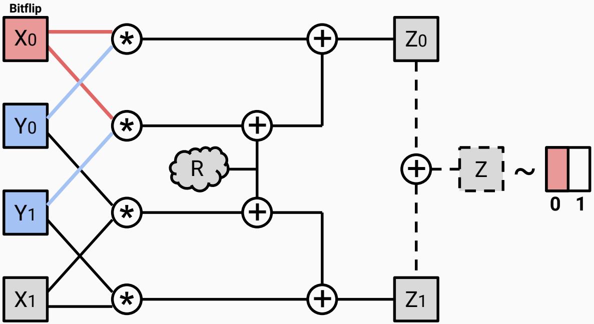


*only correct computations are considered

Faults on Masked AND

Also works with:

- Other types of faults
- Higher-order masking
- Threshold Implementations



*only correct computations are considered

Simulated Faults on Masked AES with Fault Detection

Target: First-order masked AES by Schwabe and Stoffelen et al.

- Publicly available

Simulated Faults on Masked AES with Fault Detection

Target: First-order masked AES by Schwabe and Stoffelen et al.

- Publicly available
- ARM Cortex M4, ASM optimized

Simulated Faults on Masked AES with Fault Detection

Target: First-order masked AES by Schwabe and Stoffelen et al.

- Publicly available
- ARM Cortex M4, ASM optimized

→ Originally CTR mode encryption, we only use it as block cipher

Simulated Faults on Masked AES with Fault Detection

Target: First-order masked AES by Schwabe and Stoffelen et al.

- Publicly available
- ARM Cortex M4, ASM optimized

→ Originally CTR mode encryption, we only use it as block cipher

→ Originally no fault countermeasures, we added "perfect" fault detection

Simulated Faults on Masked AES with Fault Detection

For each individual instruction in the masked Sbox:

- Simulated fault: Single bitflip in the result

Simulated Faults on Masked AES with Fault Detection

For each individual instruction in the masked Sbox:

- Simulated fault: Single bitflip in the result
- 2000 faulted Sbox computations, random inputs

Simulated Faults on Masked AES with Fault Detection

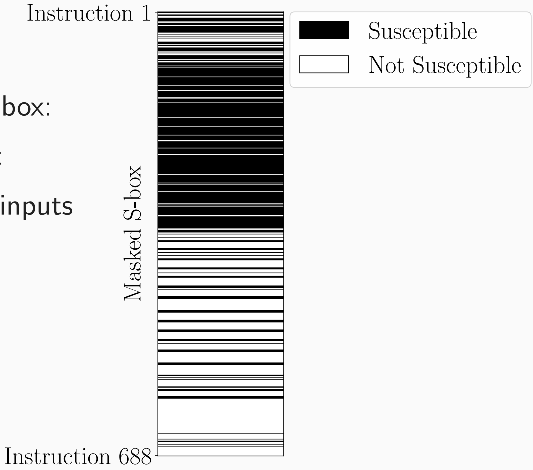
For each individual instruction in the masked Sbox:

- Simulated fault: Single bitflip in the result
- 2000 faulted Sbox computations, random inputs
- Check if correct outputs are non-uniform, i.e. if key recovery would work

Simulated Faults on Masked AES with Fault Detection

For each individual instruction in the masked Sbox:

- Simulated fault: Single bitflip in the result
- 2000 faulted Sbox computations, random inputs
- Check if correct outputs are non-uniform, i.e. if key recovery would work

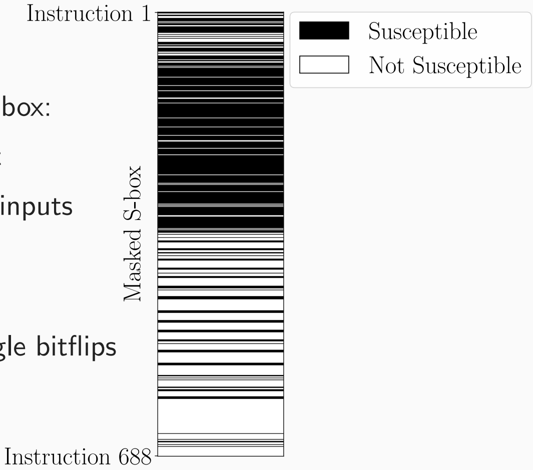


Simulated Faults on Masked AES with Fault Detection

For each individual instruction in the masked Sbox:

- Simulated fault: Single bitflip in the result
- 2000 faulted Sbox computations, random inputs
- Check if correct outputs are non-uniform, i.e. if key recovery would work

⇒ 52 % of instruction are "susceptible" to single bitflips

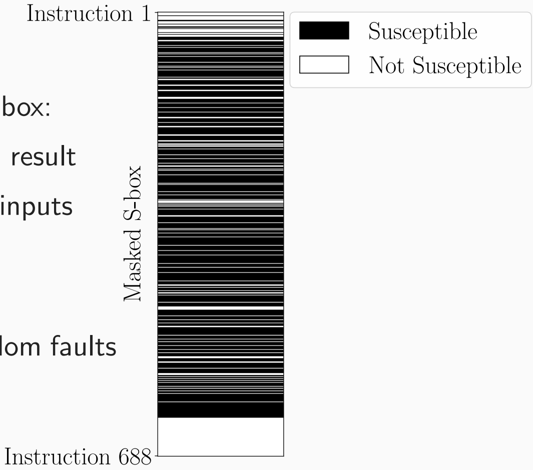


Simulated Faults on Masked AES with Fault Detection

For each individual instruction in the masked Sbox:

- Simulated fault: Randomized 8 bits of the result
- 2000 faulted Sbox computations, random inputs
- Check if correct outputs are non-uniform, i.e. if key recovery would work

⇒ 70 % of instruction are "susceptible" to random faults



Simulated Faults on Masked AES with Fault Detection

Exact numbers for one of the susceptible instructions

Fault Effect	# Ineffective Faults	# Faulted Encryptions	# Recoverable Key Bits
Flip one bit	194	386	32
Set one bit to zero	214	428	32
Randomize one bit	574	763	32
Flip one byte	192	2 940	128
Set one byte to zero	192	3 129	128
Randomize one byte	602	1 808	128
Instruction skip	400	45 527	128

Practical Results

Target: POC higher-order masked AES by Rivain et al.

- Setup: Clock glitches on ATXmega 128D4

Practical Results

Target: POC higher-order masked AES by Rivain et al.

- Setup: Clock glitches on ATXmega 128D4

→ We set masking order to 10

Practical Results

Target: POC higher-order masked AES by Rivain et al.

- Setup: Clock glitches on ATXmega 128D4

→ We set masking order to 10

→ We added “perfect” fault detection

Practical Results

Target: POC higher-order masked AES by Rivain et al.

- Setup: Clock glitches on ATXmega 128D4

→ We set masking order to 10

→ We added “perfect” fault detection

⇒ About 1000 faulted encryptions required

⇒ Thousands of possible fault locations

A word on other countermeasures

- Self Destruction
- Frequent Re-keying
- Multi Party Computation

Summing up

SIFA is quite powerful...

- Works for many ciphers and AE schemes

Summing up

SIFA is quite powerful...

- Works for many ciphers and AE schemes
- Breaks both fault and power analysis countermeasures

Summing up

SIFA is quite powerful...

- Works for many ciphers and AE schemes
- Breaks both fault and power analysis countermeasures
- Attacker does not need to hit specific bits/bytes

Summing up

SIFA is quite powerful...

- Works for many ciphers and AE schemes
- Breaks both fault and power analysis countermeasures
- Attacker does not need to hit specific bits/bytes
- Attacker does not need know how the faults influence the computation

Q?