

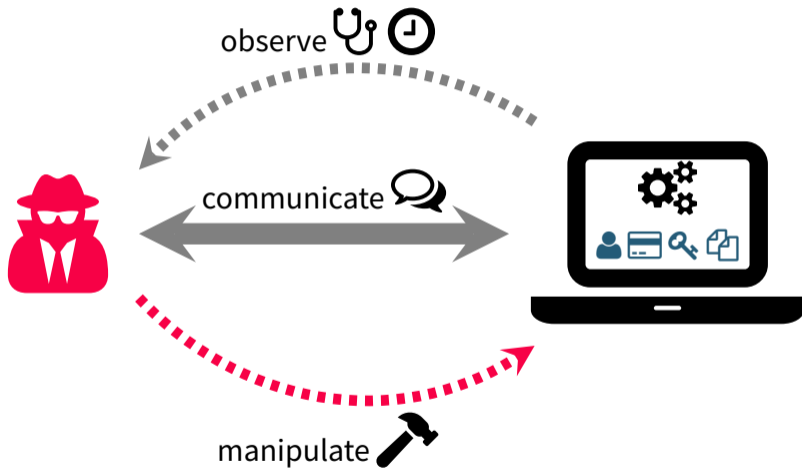
Effective Attacks from Ineffective Faults

Maria Eichlseder

Includes results of joint works with Joan Daemen, Christoph Dobraunig, Hannes Groß,
Thomas Korak, Stefan Mangard, Florian Mendel, Robert Primas

Graz Security Week, 20 September 2019

Motivation



Outline

Introduction to Fault Attacks

- ## Differential Fault Attacks
- Flipping Bits in Symmetric Crypto
 - A Detour to Differential Cryptanalysis
 - Application to AES

- ## Countermeasures
- Error Detection
 - Ineffective Fault Attacks and Friends
 - Side-Channel Countermeasures
 - Statistical Fault Attacks








Statistical Ineffective Fault Attacks

Introduction to Fault Attacks



Causing Faulty Computations

Extreme environmental conditions or targeted manipulations can cause errors in a processor's operation due to physical corruption. Examples:

-  Very high temperature
-  Unsupported supply voltage or current, voltage glitches
-  Overclocking, clock glitches
-  Excessive memory accesses
-  Strong electric or magnetic fields
-  Ionizing radiation
-  Laser

Possible Fault Effects

Fault effects in electronic devices have been studied at least since the 1950s, for example for radiation from nuclear testing:

- ⌚ Long-term effects, e.g., cumulative effect of “Total Ionization Dose (TID)”
- ⚡ Sudden effects, e.g., charged particle hits the circuit: “Single-Event Effects (SEE)”
 - Causing permanent damage (hard error)
e.g., shorts between ground and power: “Single-Event Latch-ups (SEL)”
 - Causing temporary damage (soft error)
e.g., transient pulse flips a bit in memory cell: “Single-Event Upsets (SEU)”

Some possible effects in processors:

- Flip a data bit
- Reset a data bit to 0
- Skip an instruction

Applications

Some examples:

Skipping or changing vital security check instructions

- Password comparison result
- DRM checks in Games, TV, ...
- sudo or access rights checks
- ...

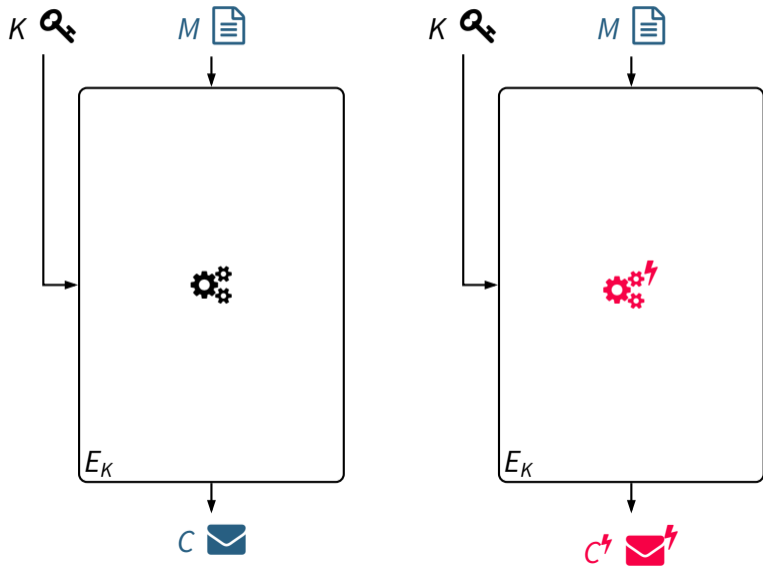
Recover a cryptographic key

- Public-key crypto, e.g., RSA [BDL97]
- Secret-key crypto, e.g., AES, DES [BS97]

Differential Fault Attacks

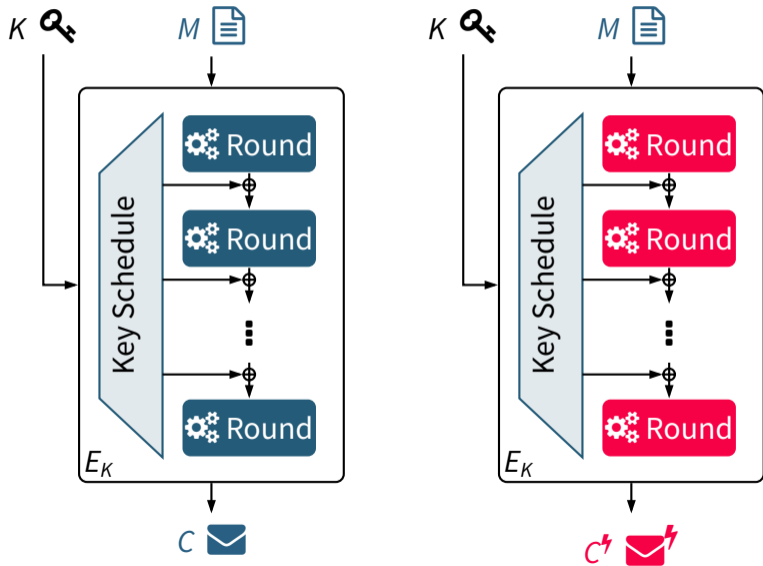


Scenario: Faulting a Block Cipher



- Multiple executions
- Get correct ciphertext C and faulty C^*


Scenario: Faulting a Block Cipher




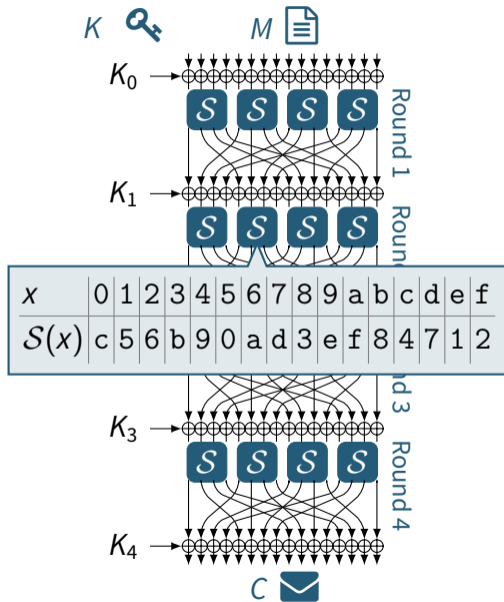
- Multiple executions
- Get correct ciphertext C and faulty C^*

Example: A 16-bit Toy Block Cipher

 Key Addition (XOR)

 S-box layer (lookup table)
for confusion

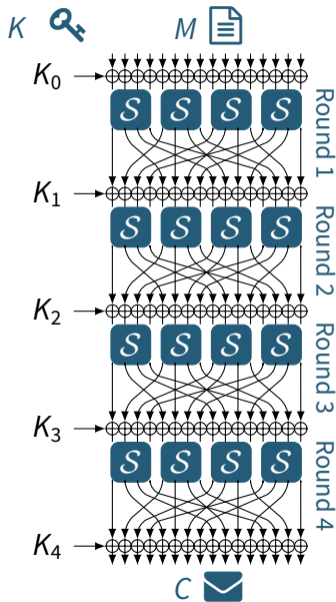
 Linear layer (rewiring, maybe XOR)
for diffusion



Example: Let's Flip a Bit

Where to put the bitflip fault?

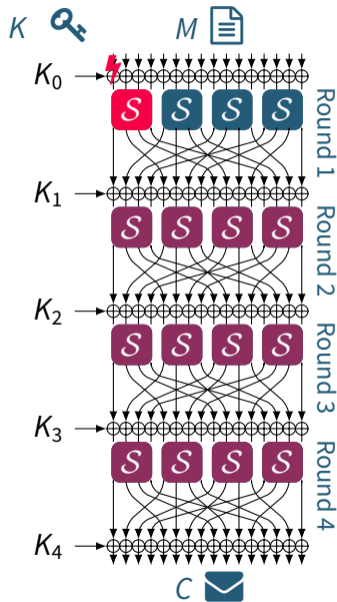
- At the beginning?
Bad idea, too much confusion!
- At the very end?
Bad idea, effect doesn't depend on the key!
- Before the last key addition?
Bad idea, effect doesn't depend on the key!
- Before the last S-box layer?
A-ha! What's going on?



Example: Let's Flip a Bit

Where to put the bitflip fault?

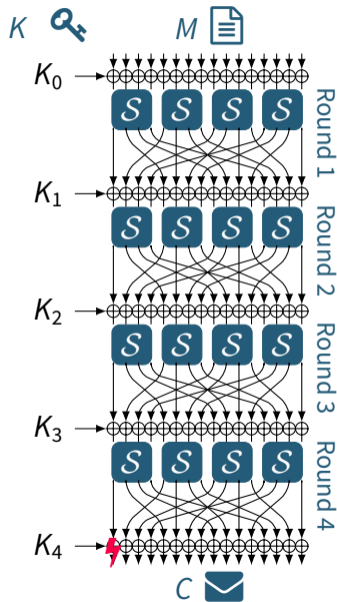
- At the beginning?
Bad idea, too much confusion!
- At the very end?
Bad idea, effect doesn't depend on the key!
- Before the last key addition?
Bad idea, effect doesn't depend on the key!
- Before the last S-box layer?
A-ha! What's going on?



Example: Let's Flip a Bit

Where to put the bitflip fault?

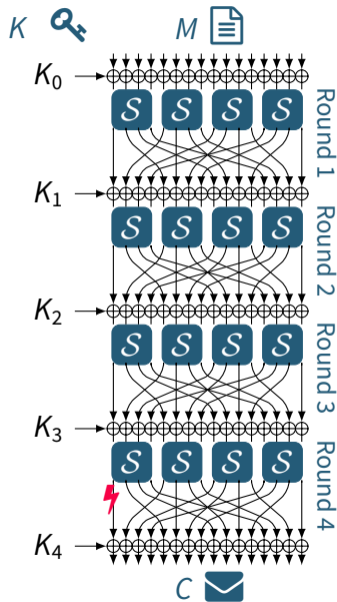
- At the beginning?
Bad idea, too much confusion!
- At the very end?
Bad idea, effect doesn't depend on the key!
- Before the last key addition?
Bad idea, effect doesn't depend on the key!
- Before the last S-box layer?
A-ha! What's going on?



Example: Let's Flip a Bit

Where to put the bitflip fault?

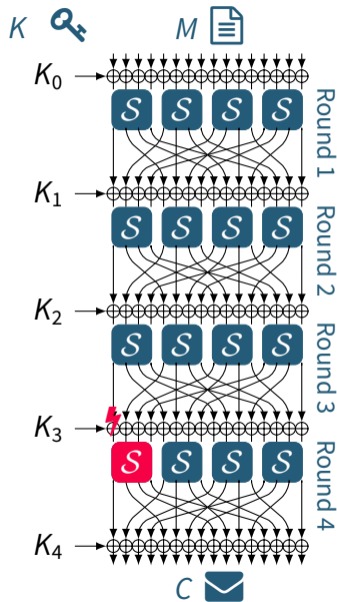
- At the beginning?
Bad idea, too much confusion!
- At the very end?
Bad idea, effect doesn't depend on the key!
- Before the last key addition?
Bad idea, effect doesn't depend on the key!
- Before the last S-box layer?
A-ha! What's going on?



Example: Let's Flip a Bit

Where to put the bitflip fault?

- At the beginning?
Bad idea, too much confusion!
- At the very end?
Bad idea, effect doesn't depend on the key!
- Before the last key addition?
Bad idea, effect doesn't depend on the key!
- Before the last S-box layer?
A-ha! What's going on?



A Detour to Differential Cryptanalysis

- One of the two most important cryptanalytic attacks for secret-key crypto
Biham and Shamir [BS90]
- Chosen-plaintext attack (no cheating with the implementation!)
- Main idea:
 1. Predict effect of plaintext difference $\Delta M = \text{📄 } M \oplus \text{📄 } M^*$ on ciphertext difference $\Delta C = \text{✉ } C \oplus \text{✉ } C^*$ without knowing $\text{🔑 } K$
 2. Use prediction as distinguisher to recover the key

Differential Properties of S-boxes

$$\Delta_{\text{in}} = 8 \rightarrow \Delta_{\text{out}} = ?$$

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$\mathcal{S}(x)$	2	0	4	3	9	5	6	7	1	d	e	f	a	8	c	b

Differential Properties of S-boxes

$$\Delta_{in} = 8 \rightarrow \Delta_{out} = ?$$

$\Delta_{in} = 8$

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$S(x)$	2	0	4	3	9	5	6	7	1	d	e	f	a	8	c	b

$\Delta_{out} = 3$

Differential Properties of S-boxes

$\Delta_{in} = 8 \rightarrow \Delta_{out} = ?$

$\Delta_{in} = 8$

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$S(x)$	2	0	4	3	9	5	6	7	1	d	e	f	a	8	c	b

$\Delta_{out} = d$

Differential Properties of S-boxes

$\Delta_{in} = 8 \rightarrow \Delta_{out} = ?$

$\Delta_{in} = 8$

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$S(x)$	2	0	4	3	9	5	6	7	1	d	e	f	a	8	c	b

$\Delta_{out} = a$

Differential Properties of S-boxes

$$\Delta_{in} = 8 \rightarrow \Delta_{out} \in \{3, a, c, d\}$$

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$\mathcal{S}(x)$	2	0	4	3	9	5	6	7	1	d	e	f	a	8	c	b



- Knowing the **value** tells us the **difference**
- Knowing the **difference** tells us (something about) the **value**:

$$\text{solutions}(\Delta_{in}, \Delta_{out}) := \{x : \mathcal{S}(x \oplus \Delta_{in}) \oplus \mathcal{S}(x) = \Delta_{out}\}$$

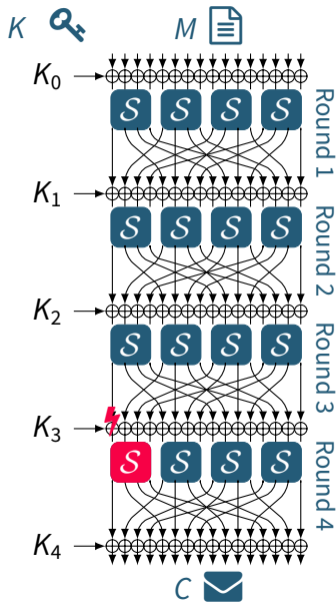
Differential Distribution Table (DDT)

I \ O	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	16	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
1	-	4	4	-	-	-	-	4	-	-	-	-	4	-	-	-
2	-	-	4	4	-	-	4	-	-	-	-	-	-	-	-	4
3	-	4	-	4	4	-	-	-	-	-	-	-	-	-	4	-
4	-	-	4	-	4	4	-	-	-	-	-	4	-	-	-	-
5	-	-	-	4	-	4	-	4	-	4	-	-	-	-	-	-
6	-	-	-	-	4	-	4	4	-	-	-	-	-	4	-	-
7	-	4	-	-	-	4	4	-	-	-	4	-	-	-	-	-
8	-	-	-	4	-	-	-	-	-	-	4	-	4	4	-	-
9	-	4	-	-	-	-	-	-	-	-	-	4	-	4	-	4
a	-	-	-	-	-	4	-	-	-	-	-	-	4	-	4	4
b	-	-	4	-	-	-	-	-	-	4	-	-	-	4	4	-
c	-	-	-	-	-	-	-	-	16	-	-	-	-	-	-	-
d	-	-	-	-	4	-	-	-	-	4	4	-	-	-	-	4
e	-	-	-	-	-	-	-	4	-	-	4	4	-	-	4	-

Example: Let's Flip a Bit – Key Recovery

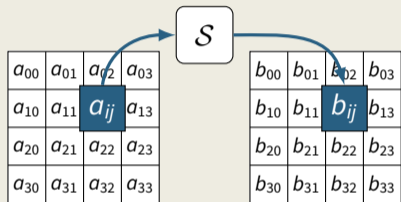
1. Obtain correct C  and faulty C^* 
2. Compute the difference $\Delta C = C \oplus C^*$ and derive the S-box output difference
3. For each possible guess of the partial key:
 - Partially decrypt C, C^* and check if the observed S-box input difference matches the fault model
 - If not, reject partial key candidate
4. Repeat to further narrow down the keys

This works for many ciphers in a similar way.



Design of AES [DR02] – Round Function (10 or 12 or 14 Rounds)

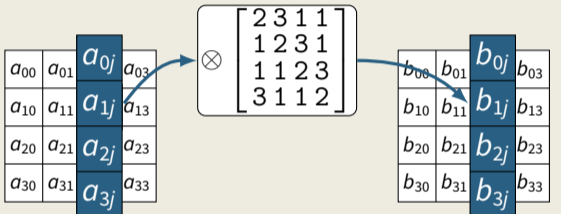
1 SubBytes (SB)



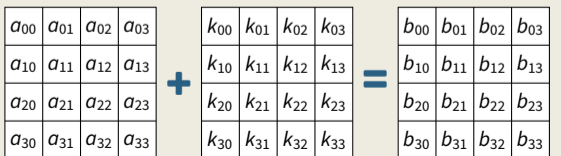
2 ShiftRows (SR)



3 MixColumns (MC)



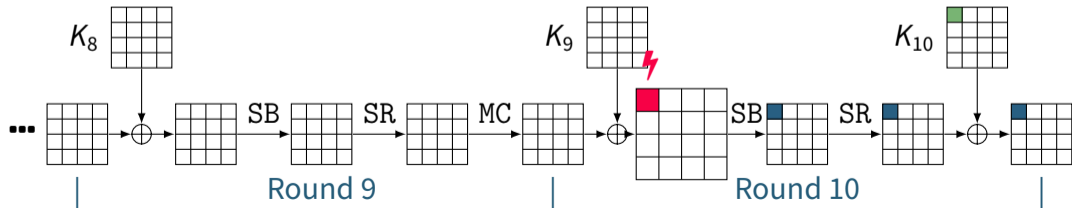
4 AddRoundKey (AK)



AES – Simple DFA

SB – SubBytes
SR – ShiftRows
MC – MixColumns

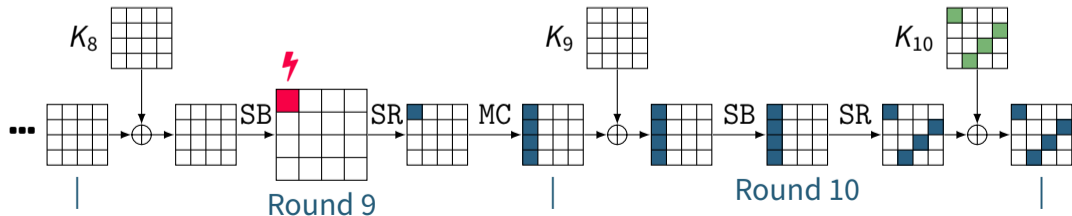
- Assume the attacker can cause precise **1-bit flips** in Round 9 of AES, before S-box
- For each of 2^8 **key guesses**,
Test if the **partial decryption** produces the expected 1-bit flip.



AES – Piret and Quisquater's DFA [PQ03]

SB – SubBytes
SR – ShiftRows
MC – MixColumns

- Assume the attacker can cause imprecise **1-byte errors**
- For each of 2^{32} **key guesses**,
Test if the **partial decryption** produces the expected 1-byte error.
(This can be optimized to require only 2 faulty encryptions to recover the full key)



Countermeasures



and Countermeasures against Countermeasures :-)

Types of Countermeasures

Physical level

- Shielding of the circuit so that it's harder to access
- Sensors that detect tampering

Implementation-level

- Detect or correct errors
- Randomize the execution details

Protocol-level

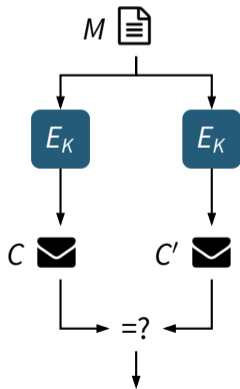
- Prevent an attacker from collecting useful data by limiting key usage, randomizing inputs, ...

Error Detection

- For DFA, the attacker requires the **faulty ciphertext** C'  and the **correct ciphertext** C  for the same plaintext M 

Countermeasure 1: Error Detection

- Check the correctness of each encryption
- For example by evaluating it twice
- Only return result if correct



Error Detection

- For DFA, the attacker requires the **faulty ciphertext** C^*  and the **correct ciphertext** C  for the same **plaintext** M 

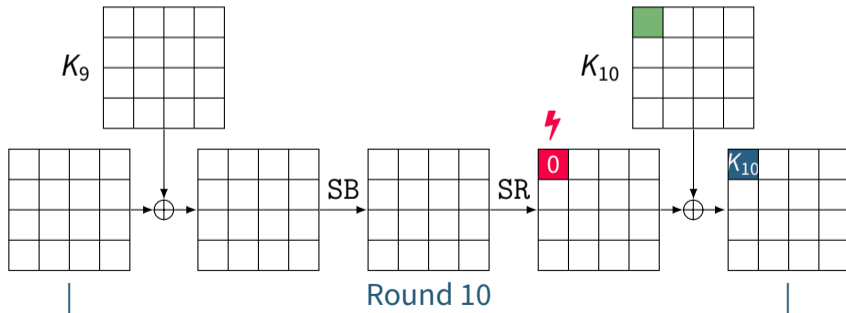
- 🛡 Countermeasure 2: **Authenticated Encryption (AEAD)**
AEAD typically prevents DFA by design:

E During **AEAD Encryption**, a random **nonce** is used to “randomize” the inputs $M \rightarrow$ cannot get C, C^* for the same M


D During **AEAD Decryption**, results are only returned if the authentication **tag** was verified correctly, so we don't get C^*

Ineffective Fault Attacks (IFA) [Cla07] and Friends

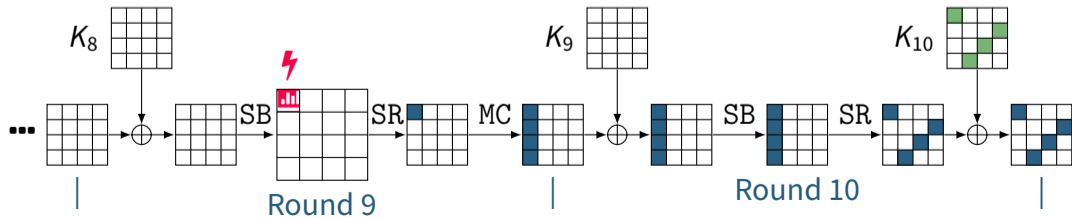
- Observation: In practice, it's often easier to cause **biased errors** than bitflips
- Example: Stuck-at-0 error sets bit (or byte) to 0
- If the attacker can reliably cause such errors, there are very simple attacks:



Statistical Fault Attacks (SFA) [FJLT13]

- Assume the attacker can cause a **biased error** (e.g., reset to 0 with prob. $\frac{1}{2}$).
- For each of 2^{32} **key guesses**,
Test if the **partial decryption** produces a non-uniform distribution  with a metric such as the Squared Euclidean Imbalance (SEI) or Pearson's χ^2 :


$$SEI(\hat{p}) = \sum_{x \in \mathcal{X}} \left| \hat{p}(x) - \frac{1}{\#\mathcal{X}} \right|^2$$



Side-Channel Countermeasures

IFA allows to “peek” at intermediate values, similar to side-channel attacks.

Many side-channel countermeasures help against IFA and friends:

 **Hiding:** Randomize the order of instructions, insert dummy instructions, etc., to make it harder for the attacker to hit the right bit

 **Masking:** Replace each data bit x by $d + 1$ random bits x_0, x_1, \dots, x_d with

$$x = x_0 \oplus x_1 \oplus \dots \oplus x_d$$

Then learning up to d bits x_i is useless for the attacker.

Statistical Ineffective Fault Attacks



Statistical Ineffective Fault Attacks (SIFA) [DEK+18; DEG+18]

So far, we inserted faults *right before / after* S-boxes.

When the attacker can only place 1 fault, error detection and/or masking prevent these attacks.

💡 **SIFA idea 1:** Use only faulty encryptions where **no fault was detected**:
This condition may lead to a **bias** in some intermediate variables!

💡 **SIFA idea 2:** Place **fault inside** the S-box circuit,
but **measure before/after** S-box with SFA methods!

This approach can attack implementations with **masking and error detection**.
It may, however, require more data (1000s of messages).

SIFA Example: Inside a Masked S-box Circuit

- Example S-box: A smaller version of SHA-3's S-box (χ)
- 3-bit input a, b, c , masked as
 - $a = a_0 \oplus a_1$
 - $b = b_0 \oplus b_1$
 - $c = c_0 \oplus c_1$
- 3-bit output r, s, t , masked as
 - $r = r_0 \oplus r_1$
 - $s = s_0 \oplus s_1$
 - $t = t_0 \oplus t_1$
- Implemented as circuit of instructions / gates XOR \oplus , AND \odot , NOT \ominus

SIFA Example: Inside a Masked S-box Circuit

Input: $\{a_0, a_1, b_0, b_1, c_0, c_1\}$

$$T_0 \leftarrow \overline{b_0} \odot c_1 ; \quad T_2 \leftarrow a_1 \odot b_1$$

$$T_1 \leftarrow \overline{b_0} \odot c_0 ; \quad T_3 \leftarrow a_1 \odot b_0$$

$$T_0 \leftarrow T_0 \oplus a_0 ; \quad T_2 \leftarrow T_2 \oplus c_1$$

$$r_0 \leftarrow T_0 \oplus T_1 ; \quad t_1 \leftarrow T_2 \oplus T_3$$

$$T_0 \leftarrow \overline{c_0} \odot a_1 ; \quad T_2 \leftarrow b_1 \odot c_1$$

$$T_1 \leftarrow \overline{c_0} \odot a_0 ; \quad T_3 \leftarrow b_1 \odot c_0$$

$$T_0 \leftarrow T_0 \oplus b_0 ; \quad T_2 \leftarrow T_2 \oplus a_1$$

$$s_0 \leftarrow T_0 \oplus T_1 ; \quad r_1 \leftarrow T_2 \oplus T_3$$

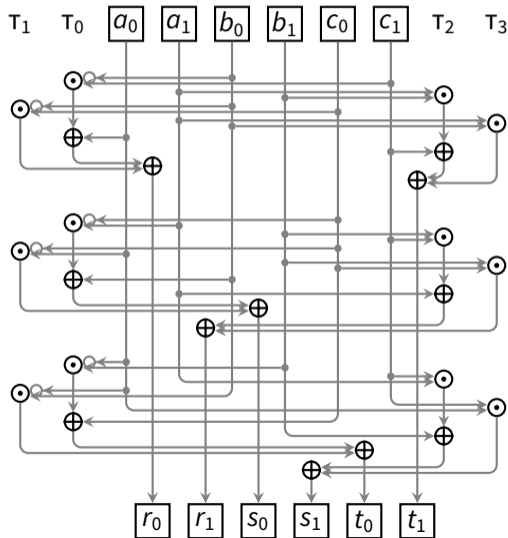
$$T_0 \leftarrow \overline{a_0} \odot b_1 ; \quad T_2 \leftarrow c_1 \odot a_1$$

$$T_1 \leftarrow \overline{a_0} \odot b_0 ; \quad T_3 \leftarrow c_1 \odot a_0$$

$$T_0 \leftarrow T_0 \oplus c_0 ; \quad T_2 \leftarrow T_2 \oplus b_1$$

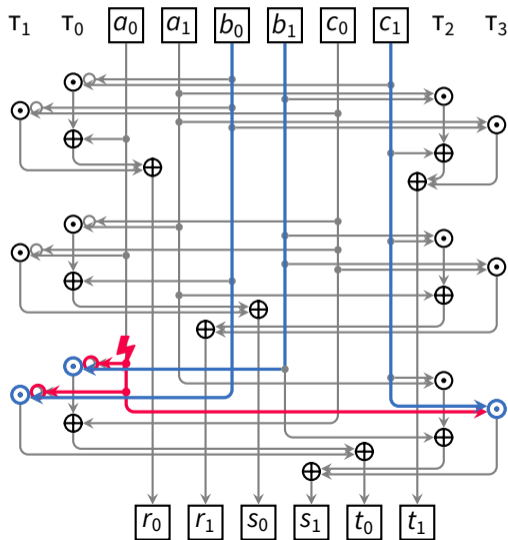
$$t_0 \leftarrow T_0 \oplus T_1 ; \quad s_1 \leftarrow T_2 \oplus T_3$$

Output: $\{r_0, r_1, s_0, s_1, t_0, t_1\}$



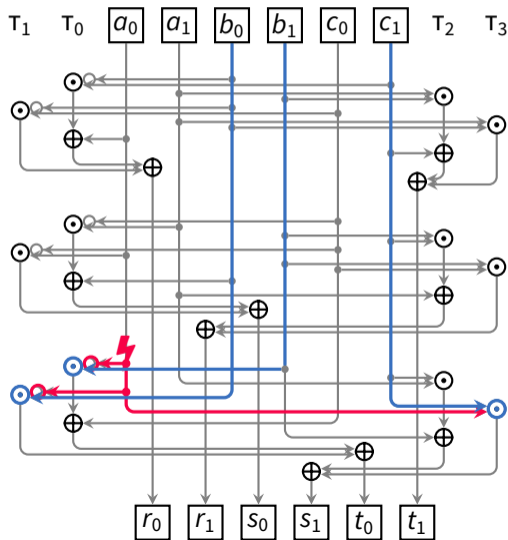
SIFA Example: Inside a Masked S-box Circuit

- Cause a bitflip fault in $\text{⚡}a_0$ at the indicated moment
- The faulty value goes into 3 \odot s
- Correctness of the \odot -output depends on the other input
 - if the other input is 0, the \odot -output is correct
 - if the other input is 1, the \odot -output is faulty



SIFA Example: Inside a Masked S-box Circuit

- The S-box output is correct if \odot with c_1 is correct and
 - both \odot s with b_0, b_1 are correct: $b_0 = b_1 = 0$, or
 - both \odot s with b_0, b_1 are faulty: $b_0 = b_1 = 1$
- Either way, $b = b_0 \oplus b_1 = 0$
- If the cipher output is correct, learn $b = 0$ (bias)
- Use as before to recover the key!



SIFA Example: Application to AES

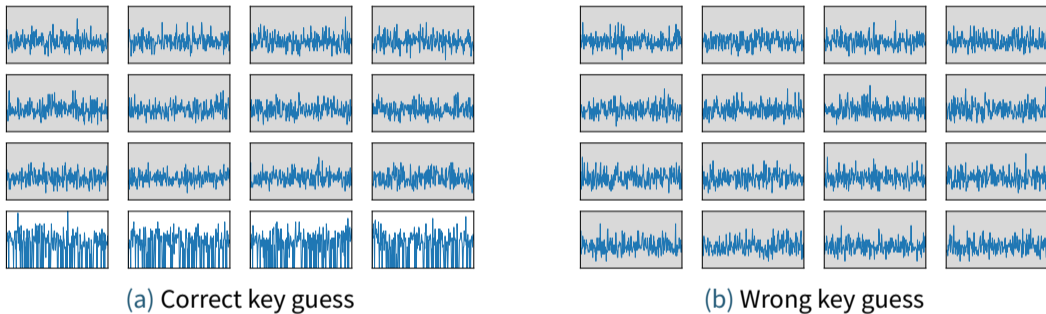
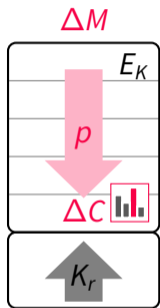
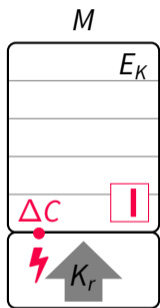


Figure: Results for **bitsliced AES** implementation on 32-bit platform (ARM Cortex M4) with **masking** (1st order) and **error detection** (temporal redundancy). Simulated byte-stuck-at-0 faults. Recovered distribution after S-box in round 9. [DEG+18]

Statistical (Ineffective) Fault Attacks



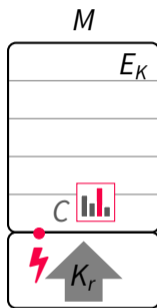
Diff.
cryptanalysis
DC [BS90]



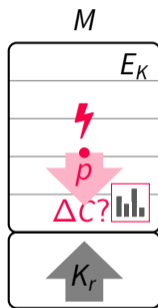
Diff. fault attack
DFA [BS97]



Stat. fault attack
SFA [FJLT13;
DEK+16]

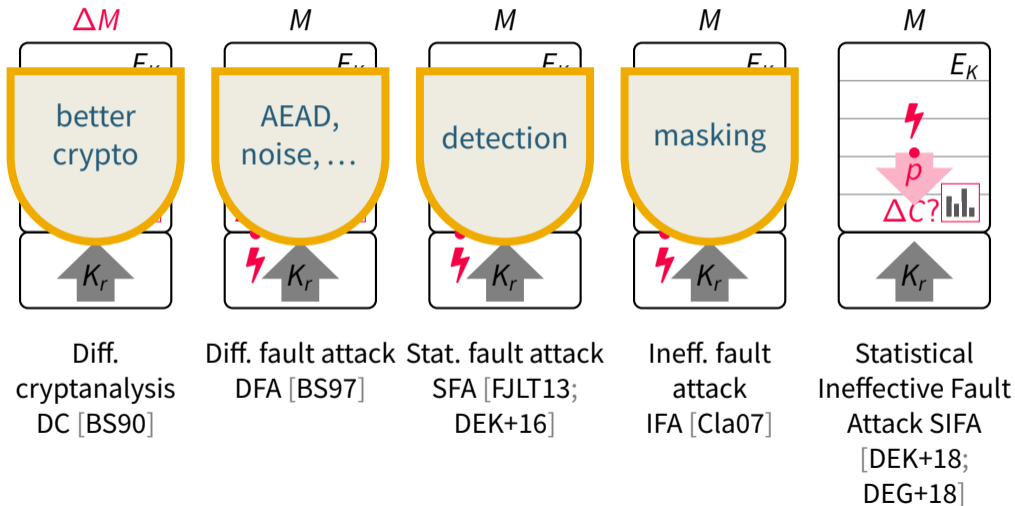


Ineff. fault
attack
IFA [Cla07]



Statistical
Ineffective Fault
Attack SIFA
[DEK+18;
DEG+18]

Statistical (Ineffective) Fault Attacks



Conclusion

- ⚡ Fault attacks are a very powerful type of implementation attacks
- 🛡 Countermeasures include error detection, side-channel countermeasures (hiding, masking), and physical protection
- 🔍 With enough effort (money, time, data), attackers may be able to defeat countermeasures – make sure this effort is higher than it's worth!

Questions



Bibliography I

- [BDL97] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. **On the Importance of Checking Cryptographic Protocols for Faults (Extended Abstract)**. Advances in Cryptology – EUROCRYPT '97. Vol. 1233. LNCS. Springer, 1997, pp. 37–51. doi: [10.1007/3-540-69053-0_4](https://doi.org/10.1007/3-540-69053-0_4).
- [BS90] Eli Biham and Adi Shamir. **Differential Cryptanalysis of DES-like Cryptosystems**. Advances in Cryptology – CRYPTO 1990. Vol. 537. LNCS. Springer, 1990, pp. 2–21. doi: [10.1007/3-540-38424-3_1](https://doi.org/10.1007/3-540-38424-3_1).
- [BS97] Eli Biham and Adi Shamir. **Differential Fault Analysis of Secret Key Cryptosystems**. Advances in Cryptology – CRYPTO '97. Vol. 1294. LNCS. Springer, 1997, pp. 513–525. doi: [10.1007/BFb0052259](https://doi.org/10.1007/BFb0052259).
- [Cla07] Christophe Clavier. **Secret External Encodings Do Not Prevent Transient Fault Analysis**. Cryptographic Hardware and Embedded Systems – CHES 2007. Vol. 4727. LNCS. Springer, 2007, pp. 181–194. doi: [10.1007/978-3-540-74735-2_13](https://doi.org/10.1007/978-3-540-74735-2_13).
- [DEG+18] Christoph Dobraunig, Maria Eichlseder, Hannes Groß, Stefan Mangard, Florian Mendel, and Robert Primas. **Statistical Ineffective Fault Attacks on Masked AES with Fault Countermeasures**. Advances in Cryptology – ASIACRYPT 2018. Vol. 11273. LNCS. Springer, 2018, pp. 315–342. doi: [10.1007/978-3-030-03329-3_11](https://doi.org/10.1007/978-3-030-03329-3_11).

Bibliography II

- [DEK+16] Christoph Dobraunig, Maria Eichlseder, Thomas Korak, Victor Lomné, and Florian Mendel. **Statistical Fault Attacks on Nonce-Based Authenticated Encryption Schemes**. *Advances in Cryptology – ASIACRYPT 2016*. Vol. 10031. LNCS. Springer, 2016, pp. 369–395. doi: [10.1007/978-3-662-53887-6_14](https://doi.org/10.1007/978-3-662-53887-6_14).
- [DEK+18] Christoph Dobraunig, Maria Eichlseder, Thomas Korak, Stefan Mangard, Florian Mendel, and Robert Primas. **SIFA: Exploiting Ineffective Fault Inductions on Symmetric Cryptography**. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2018.3 (2018), pp. 547–572. doi: [10.13154/tches.v2018.i3.547-572](https://doi.org/10.13154/tches.v2018.i3.547-572).
- [DR02] Joan Daemen and Vincent Rijmen. **The Design of Rijndael: AES – The Advanced Encryption Standard**. *Information Security and Cryptography*. Springer, 2002. ISBN: 3-540-42580-2. doi: [10.1007/978-3-662-04722-4](https://doi.org/10.1007/978-3-662-04722-4).
- [FJLT13] Thomas Fuhr, Éliane Jaulmes, Victor Lomné, and Adrian Thillard. **Fault Attacks on AES with Faulty Ciphertexts Only**. *Fault Diagnosis and Tolerance in Cryptography – FDTC 2013*. IEEE Computer Society, 2013, pp. 108–118. doi: [10.1109/FDTC.2013.18](https://doi.org/10.1109/FDTC.2013.18).
- [PQ03] Gilles Piret and Jean-Jacques Quisquater. **A Differential Fault Attack Technique against SPN Structures, with Application to the AES and KHAZAD**. *Cryptographic Hardware and Embedded Systems – CHES 2003*. Vol. 2779. LNCS. Springer, 2003, pp. 77–88. doi: [10.1007/978-3-540-45238-6_7](https://doi.org/10.1007/978-3-540-45238-6_7).