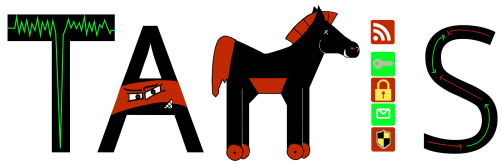
IoT and Physical Attacks

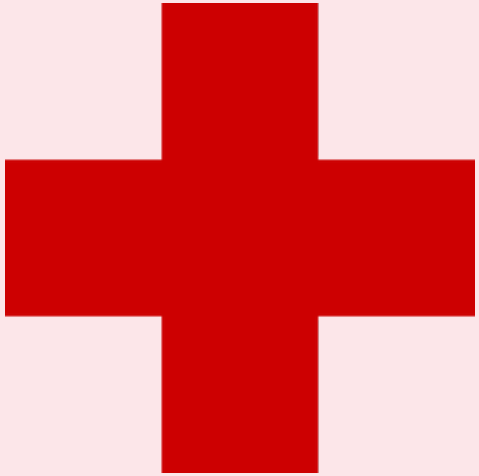
Hélène Le Bouder, Ronan Lashermes, Jean-Louis Lanet
Thierno Barry, Damien Courroussé.



PÔLE D'EXCELLENCE
CYBER

Security in IOT

Internet of things



Natural idea to improve security

1. Add cryptography for data.
2. Authentify user with a PIN code.

Physical Attacks

- A cipher or PIN code Verification algorithm is securely designed against theoretical attacks (ex : classic cryptanalysis).
- The implementation of an algorithm on a device, introduces new security vulnerabilities.
- Physical attacks $\neq$ classic cryptanalysis and logical attacks.
- Physical attacks can be divided in two families.

The side channel analysis consist in observing some physical characteristics which are modified during the computation performed on the circuit. Most classic leakages are: timing, power consumption, electromagnetic emissions (EM)

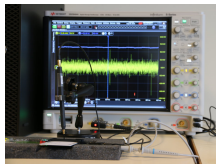
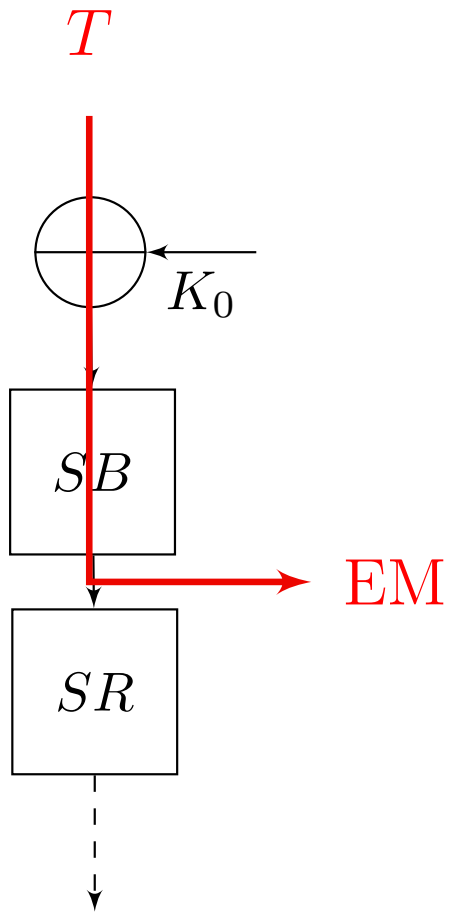


Figure: EMA, the side channel platform of the LHS

- Side Channel Attacks on block ciphers: physical values of a device leak information about intermediate state of the cipher.
- Typical SCA links texts and measurements.
- Restricted on the first or last round.



The fault injection attacks consist in disrupting the circuit behavior. Disruption means: timing, power consumption, electromagnetic pulses, laser.

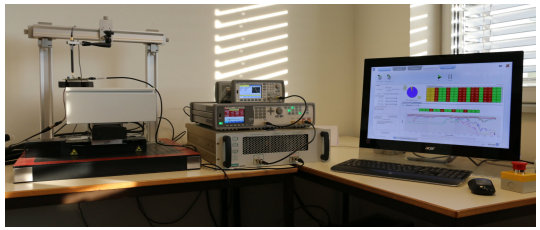


Figure: Faustine, the fault injection platform of the LHS

Personal Identification Number (PIN) codes.

- Used to authenticate the user,
- mostly in payment cards or SIM cards...
- Targets of choice for malicious adversaries.
- A limited number of trials.

Personal Identification Number (PIN) codes.

Verify PIN algorithm

1: **procedure** VERIFY PIN(candidate PIN V)
2: counter = counter − 1
3: **if** counter > 0 **then**
4: status = COMPARISON(U, V)
5: status$_2$ = COMPARISON(U, V)
6: **if** status ≠ status$_2$ **then**
7: ERROR, device is blocked
8: **else**
9: **if** status = TRUE **then**
10: counter initialized at original value.
11: **end if**
12: **end if**
13: **else**
14: device is blocked
15: **end if**
16: **return** status
17: **end procedure**

- PIN code is an array of m digits.
- **True PIN**: U,
- **Candidate PIN**: V,
- $U \in [\![0, 9]\!]^m$.
- 10^m different PIN codes.
- Countermeasure against fault attack: compare U and V twice.

Personal Identification Number (PIN) codes.

Comparison of two PIN codes

```
1:  procedure COMPARISON(candidate PIN V, true PIN U)
2:      status = FALSE
3:      diff = FALSE
4:      fake = FALSE
5:      for b = 0 to m do
6:          if U_b ≠ V_b then
7:              diff = TRUE
8:          else
9:              fake = TRUE
10:         end if
11:         if (b = m) and (diff = FALSE) then
12:             status = TRUE
13:         else
14:             fake = TRUE
15:         end if
16:     end for
17:     return status
18: end procedure
```

Countermeasure against timing attack:
comparison between U and V has to be in a constant time.

A template attack

The main difficulty of the attack is to succeed with very few traces.

A template attack

The main difficulty of the attack is to succeed with very few traces.

2 phases

1. profiling phase,
2. attack phase.

The attacker can :

- obtain one trace on the targeted device;
- change the True PIN in her profiling device;
- obtain many traces on her profiling device.

On the profiling device

Step 1: Campaign on the profiling device

- Campaign is for one given digit b.
- The digit U_b of the True PIN takes all values k in $[\![0, 9]\!]$ and the other digits stay to zero.
- Digits of Candidate PIN V are fixed to a chosen value v.
- For each (k, v) collect many traces: $M_{v,k} = \left\{xk_{(i,j)}\right\}$, i for trace, j for time.

On the profiling device

Step 1: Campaign on the profiling device

- Campaign is for one given digit b.
- The digit U_b of the True PIN takes all values k in $[\![0, 9]\!]$ and the other digits stay to zero.
- Digits of Candidate PIN V are fixed to a chosen value v.
- For each (k, v) collect many traces: $M_{v,k} = \left\{xk_{(i,j)}\right\}$, i for trace, j for time.

Step 2: Detection of points of interest.

Select the moment of computation of Comparison (relevant j).

On the profiling device

Step 1: Campaign on the profiling device

- Campaign is for one given digit b.
- The digit U_b of the True PIN takes all values k in $[\![0, 9]\!]$ and the other digits stay to zero.
- Digits of Candidate PIN V are fixed to a chosen value v.
- For each (k, v) collect many traces: $M_{v,k} = \left\{xk_{(i,j)}\right\}$, i for trace, j for time.

Step 2: Detection of points of interest.

Select the moment of computation of Comparison (relevant j).

Step 3: Build of templates.

- Compute the covariance matrix $S_{v,k} = \{sk_{(j,j')}\}$,

$$sk_{(j,j')} = \frac{1}{n-1} \cdot \left(xk_j - \overline{xk}_j\right)^t \left(xk_{j'} - \overline{xk}_{j'}\right) \quad .$$

On targeted device

Step 4: Campaign on the targeted device

- True PIN digit U_b is **unknown**, it is the target;
- Candidate PIN digit V_b is equal to v.
- Trace is a vector $T_v = \{x_j\}$.

On targeted device

Step 4: Campaign on the targeted device

- True PIN digit U_b is **unknown**, it is the target;
- Candidate PIN digit V_b is equal to v.
- Trace is a vector $T_v = \{x_j\}$.

Step 5: Confrontation between measurements

- Confront the trace T_v to the template matrix $S_{v,k}$.
- General formula in template attack:
$$F_v\left(T_v | S_{v,k}, \overline{xk}\right) = \frac{1}{\sqrt{2\pi^p \cdot |S_{v,k}|}} \cdot \exp\left(-\frac{1}{2} \cdot \left(T_v - \overline{xk}\right) \cdot S_{v,k}^{-1} \cdot \left(T_v - \overline{xk}\right)^t\right).$$

On targeted device

Step 4: Campaign on the targeted device

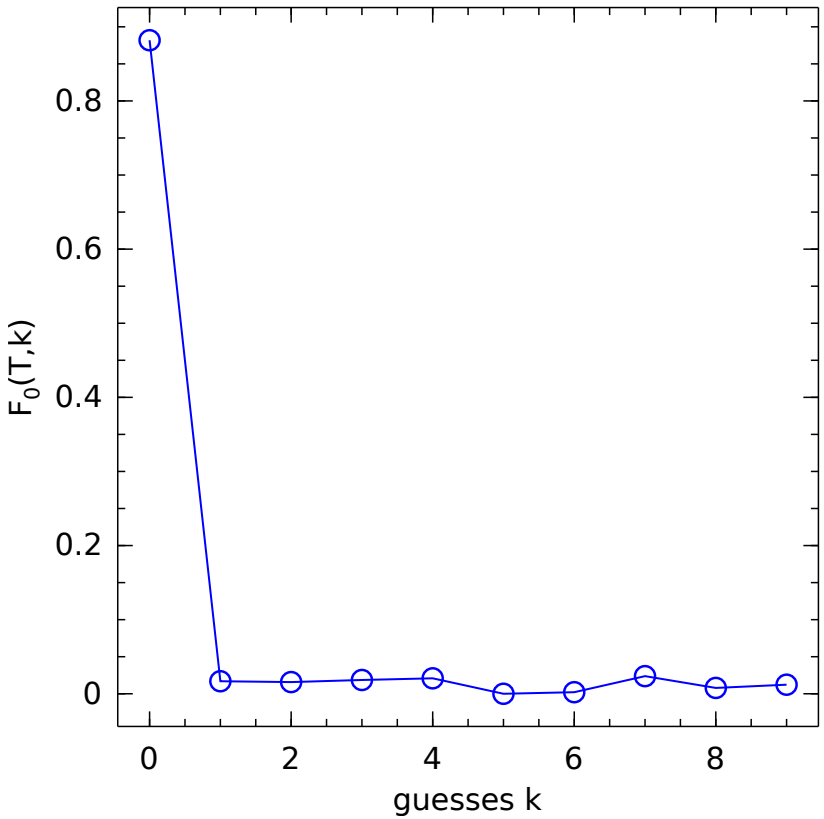
- True PIN digit U_b is **unknown**, it is the target;
- Candidate PIN digit V_b is equal to v.
- Trace is a vector $T_v = \{x_j\}$.

Step 5: Confrontation between measurements

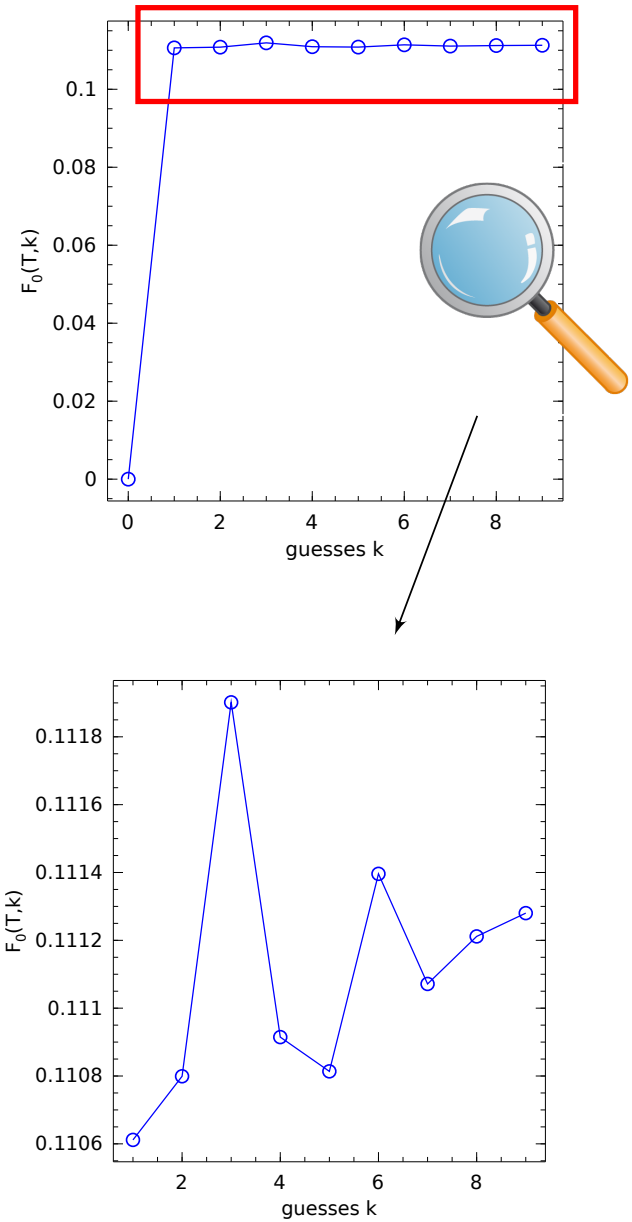
- Confront the trace T_v to the template matrix $S_{v,k}$.
- General formula in template attack:
$$F_v\left(T_v | S_{v,k}, \overline{xk}\right) = \frac{1}{\sqrt{2\pi^p \cdot |S_{v,k}|}} \cdot \exp\left(-\frac{1}{2} \cdot \left(T_v - \overline{xk}\right) \cdot S_{v,k}^{-1} \cdot \left(T_v - \overline{xk}\right)^t\right).$$

Step 6: Discriminating guesses

- Return the guess k_v for which F_v is maximal for a given T_v.
- Rank the guesses k according to the value of $F_v(T_v, k)$.



- The True digit PIN:
  $U_b = 0$
- The Candidate digit PIN:
  $V_b = 0$
- The returned guess is clearly:
  $k = 0$
- If $U_b = V_b$.
  The attack always succeeds.



- The True PIN digit: $U_b = 3$.
- The Candidate PIN digit: $V_b = 0$.
- The returned guess is $k = 3$.
- $U_b \neq V_b$: The attack succeeds, not so clearly.

Success rate

number of traces:		1	2	3	4	5	6	7	8
$n = 100000$	1 COMPARAISON	27.70	41.47	53.84	63.99	73.07	81.33	88.51	100
	2 COMPARAISON	31.71	46.56	57.82	67.76	76.63	84.36	90.68	100
$n = 200000$	1 COMPARAISON	29.28	44.27	56.79	67.41	76.66	83.91	90.68	100
	2 COMPARAISON	32.72	49.52	61.96	72.05	80.49	87.53	93.23	100
$n = 400000$	1 COMPARAISON	29.56	44.11	56.0	66.88	75.96	84.04	90.58	100
	2 COMPARAISON	32.91	48.38	60.88	71.68	80.07	86.91	92.94	100

- Success rate to retrieve a digit of a True PIN U_b according to the size n of the templates and the number and the choice of traces.
- Attack on an STM32 microcontroller with a low cost EM analysis platform.
- **Worst case**: in 8 trials, the PIN code is retrieved.

How to protect your design?

Generic countermeasures

- Masking against side channels,
- redundancy (ip replication, error detecting codes, ...) against fault attacks,
- ...

Specific countermeasures (for PIN codes)

- Device specific secret key K,
- store $S = HMAC_K(U)$ on chip instead of PIN code U.
- To test candidate V, compare S with $HMAC_K(V)$ on chip.

Security hygiene

Basic principles that designers must follow:

- No security through obscurity,
- if a device is compromised, the whole system must not be (no global secret on all devices),
- secrets should not be handled by the device if possible (reduce physical attacks exposure),
- impose first access security configuration (no default login/password).

Cogito

- Introduce randomization with runtime code generation to produce polymorphic application components.
- Use semantic equivalences at the instruction level to produce different instances of code sequences.
- Shuffle, at runtime, the machine instructions and randomize the mapping to physical registers.
- Combine with hardware protections.
- With limited memory consumption and fast code generation, so that it is applicable in very small computing units such as secure elements.

- Adding security is not a trivial thing.
- Cryptography and PIN code may be vulnerable to physical attacks.
- Trade-off must be made between resource use and security.
- Dedicated chips embedding security features is a must-have for IoT devices to be truly considered secure.
- Software solutions are also possible in some cases.

Thank you for your attention !

Do you have any questions?