

# Fault Injection Attacks and Countermeasures in Embedded Processors

Arnaud Tisserand

CNRS, Lab-STICC laboratory

ARCHI'17, Nancy



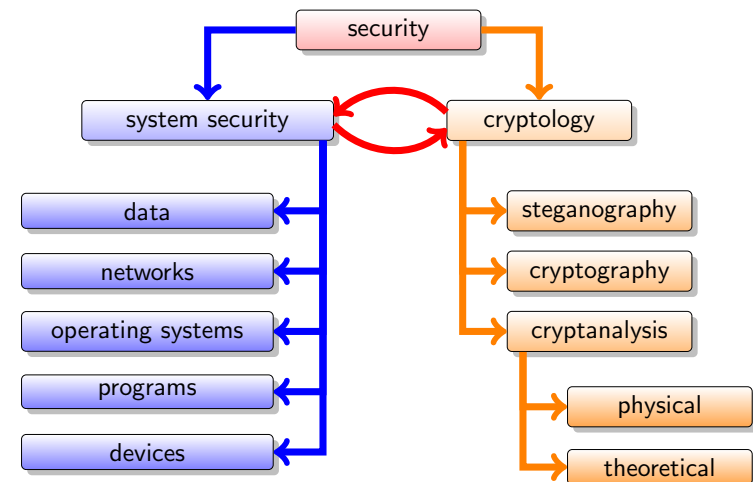
- Introduction
- Cryptographic Background  
See presentation from Jérémie Detrey for more details
- Side Channel Attacks
- Fault Injection Attacks
- Protections
- Conclusion and References

## Applications with Security Needs

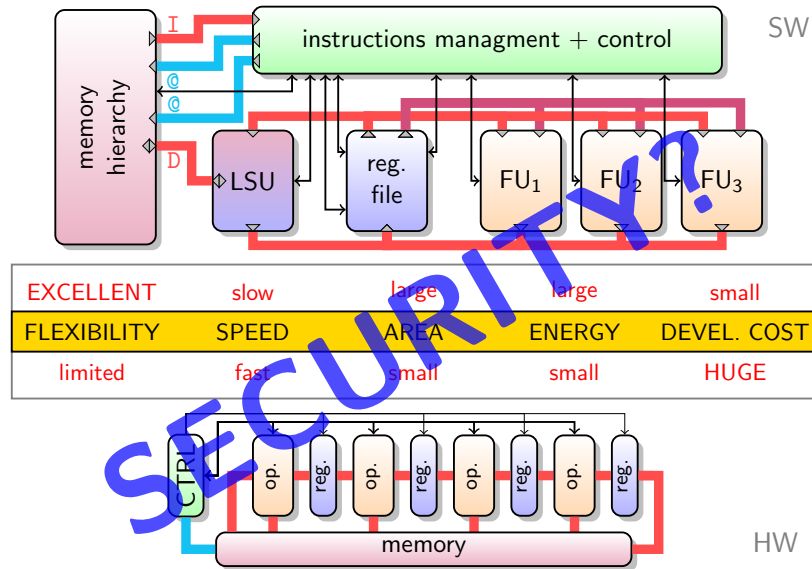


**Applications:** smart cards, computers, Internet, telecommunications, set-top boxes, data storage, RFID tags, WSN, smart grids...

## Security Aspects



## Software vs Hardware Support



## Cryptographic Features

### Objectives:

- Confidentiality
- Integrity
- Authenticity
- Non-repudiation
- ...

### Cryptographic primitives:

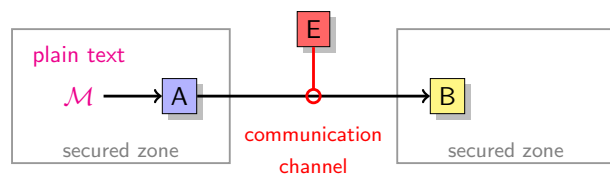
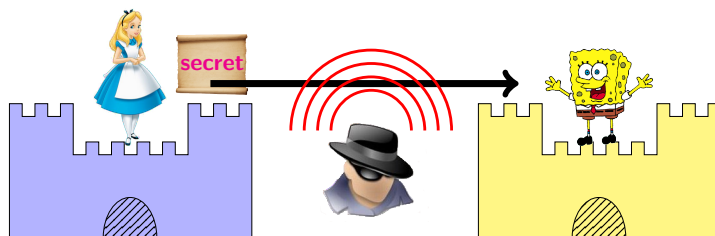
- Encryption
- Digital signature
- Hash function
- Random numbers generation
- ...

### Implementation issues:

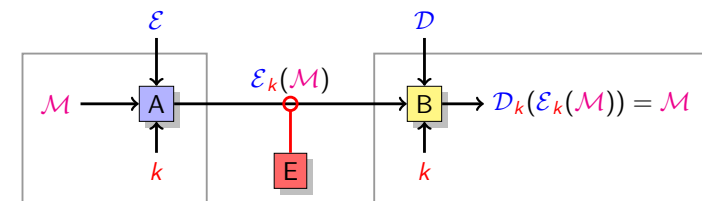
- **Performances:** speed, delay, throughput, latency
- **Cost:** device (memory, size, weight), low power/energy consumption, design
- **Security:** protection against attacks

## Basic Cyphering

Alice wants to **secretly** send a message to Bob in such a way **Eve** (eavesdropper/spy) should have **no** information

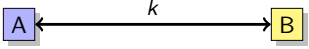
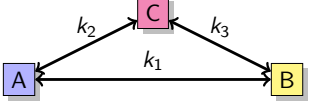
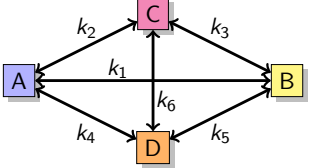


## Symmetric / Private-Key Cryptography

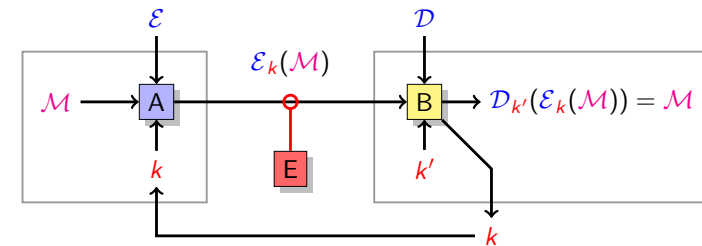


- **A**: Alice, **B**: Bob
- $\mathcal{M}$ : plain text/message
- $\mathcal{E}$ : encryption/ciphering algorithm,  $\mathcal{D}$ : decryption/deciphering algorithm
- $k$ : secret key to be shared by A and B
- $\mathcal{E}_k(\mathcal{M})$ : encrypted text
- $\mathcal{D}_k(\mathcal{E}_k(\mathcal{M}))$ : decrypted text
- **E**: eavesdropper/spy

## Symmetric Cryptography Limitation

people		required keys	
$n$	list	list	number
2	A, B		1
3	A, B, C		3
4	A, B, C, D		6
$n$	A, ...		$\frac{n \times (n-1)}{2}$

## Asymmetric / Public-Key Cryptography



- $k$ : B's **public key** (known to everyone including E)
- $\mathcal{E}_k(M)$ : ciphered text
- $k'$ : B's **private key** (must be kept secret)
- $\mathcal{D}_{k'}(\mathcal{E}_k(M))$ : deciphered text

## [Trapdoor] One Way Function

One way function:  $f : x \mapsto y = f(x)$

- given  $x$ , computing  $y$  is **easy**
- given  $y$ , computing  $x$  is **very hard**

Trapdoor one way function:  $f : x \mapsto y = f(x)$

- given  $x$ , computing  $y$  is **easy**
- given  $y$ , computing  $x$  is **very hard**
- given some (secret) information and  $y$ , computing  $x$  is **easy**

Example:  $p$  and  $q$  primes, computing  $n = pq$  is easy but finding  $(p, q)$  knowing just  $n$  is very hard

## Symmetric or Asymmetric Cryptography?

Private-key or **symmetric** cryptography:

- ☺ simple algorithms
  - ➔ fast computation
  - ➔ limited cost (silicon area, energy)
- ☹ requires a key exchange
- ☹ key distribution problem for  $n$  persons

Public-key or **asymmetric** cryptography:

- ☺ no key exchange required
- ☺ only 2 keys per person (1 private, 1 public)
- ☺ allows digital signature
- ☹ more complex algorithms
  - ➔ slower computation
  - ➔ higher cost

## Advanced Encryption Standard (AES)

Established by NIST  
in 2001

Symmetric encryption

Block size: 128 bits

key length	#round
128	10
192	12
256	14

Based on substitution-  
permutation  
network

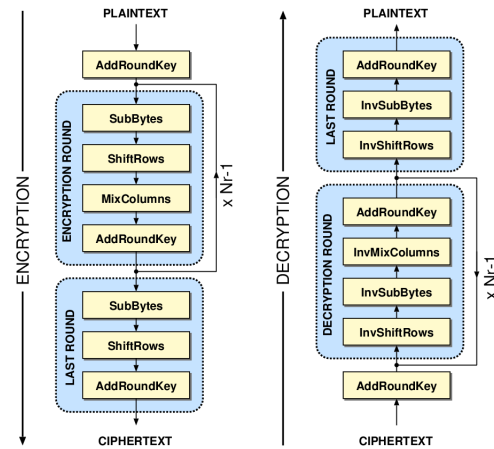
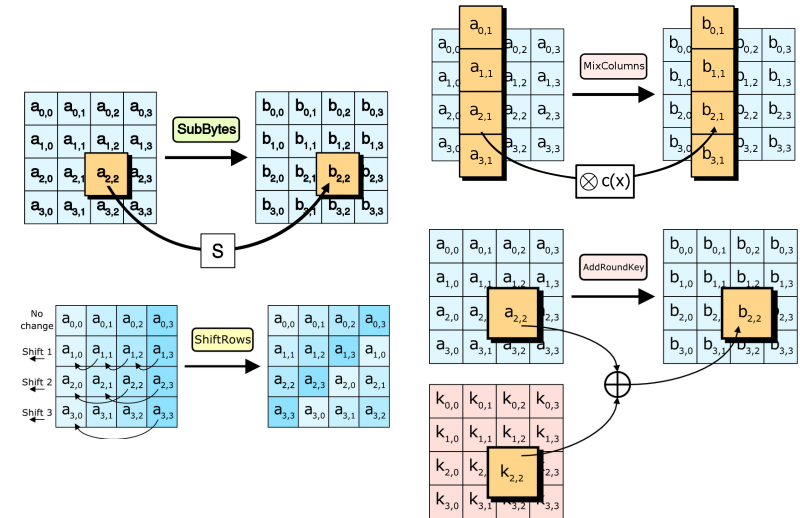


Image source: <http://fr.wikipedia.org/>

## AES Round Operations



Images source: <http://fr.wikipedia.org/>

NIST: National Institute of Standards and Technology

## RSA Asymmetric Cryptosystem (1/2)

Published in 1978 by Ron Rivest, Adi Shamir and Leonard Adleman [19]

**Key generation** (Alice side)

- Choose two **large prime integers**  $p$  and  $q$
- Compute the **modulus**  $n = pq$
- Compute  $\varphi(n) = (p - 1)(q - 1)$
- Choose an integer  $e$  such that  $1 < e < \varphi(n)$  and  $\text{gcd}(e, \varphi(n)) = 1$
- Compute  $d = e^{-1} \pmod{\varphi(n)}$
- **Private key** (kept secret by Alice):  $d$  and also  $p, q, \varphi(n)$
- **Public key** (published):  $(n, e)$

## RSA Asymmetric Cryptosystem (2/2)

Private key (Alice):  $d$

Public key (all):  $(n, e)$

**Encryption** (Bob side):

- convert the message  $M$  to an integer  $m$  ( $1 < m < n$  and  $\text{gcd}(m, n) = 1$ )
- compute the **cipher text**  $c = m^e \pmod{n}$

**Decryption** (Alice side):

- compute  $m = c^d \pmod{n}$
- convert the integer  $m$  to the message  $M$

**Theoretical security:** **integer factorization**, i.e. computing  $(p, q)$  knowing  $n$ , is not possible when  $n$  is large enough



## Modular Exponentiation

Computation of operations such as :  $a^b \bmod n$

$$a^b = \underbrace{a \times a \times a \times a \times \dots \times a \times a \times a}_a \text{ appears } b \text{ times}$$

Order of magnitude of exponents:  $2^{\text{size of exponent}} \rightsquigarrow 2^{1024} \dots 2^{2048} \dots 2^{4096}$

Fast exponentiation principle:

$$\begin{aligned} a^b &= (a^2)^{\frac{b}{2}} && \text{when } b \text{ is even} \\ &= a \times (a^2)^{\frac{b-1}{2}} && \text{when } b \text{ is odd} \end{aligned}$$

Least significant bit of the exponent: bit = 0  $\rightsquigarrow$  even and bit = 1  $\rightsquigarrow$  odd

## Square and Multiply Algorithm

**input** :  $a, b, n$  where  $b = (b_{t-1}b_{t-2} \dots b_1b_0)_2$

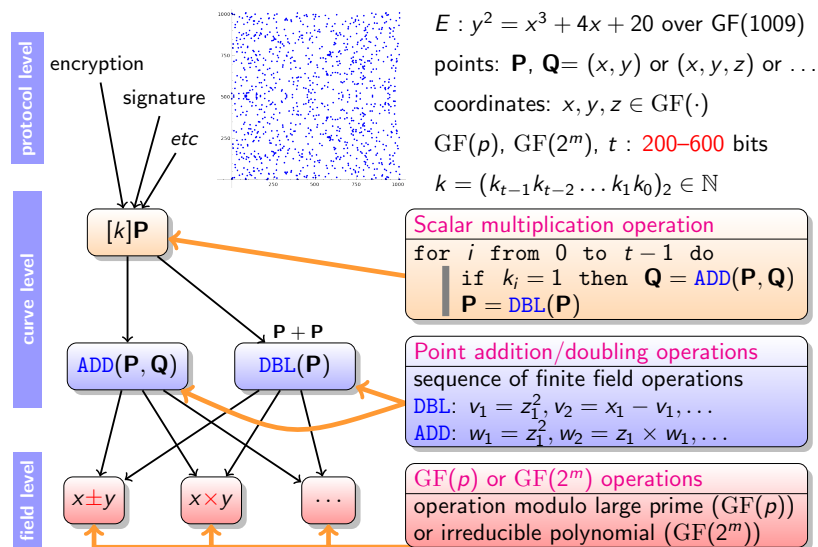
**output** :  $a^b \bmod n$

```

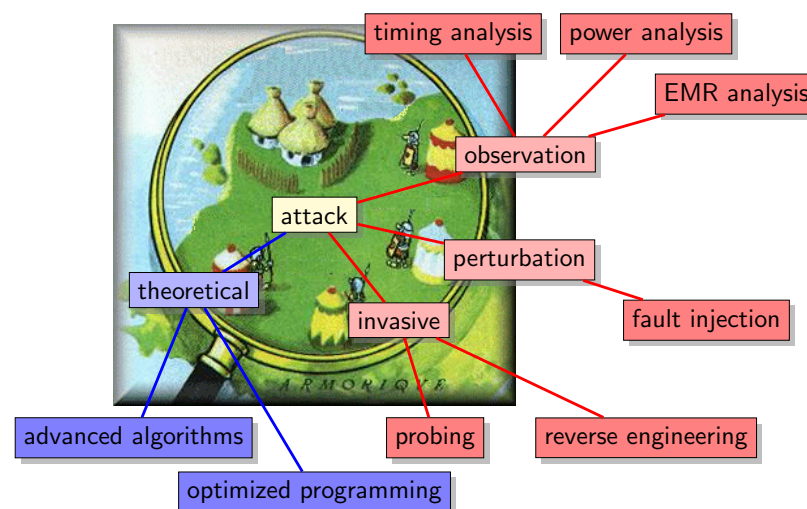
r = 1
for i from 0 to t-1 do
  if b_i = 1 then
    r = r · a mod n
  endif
  a = a^2 mod n
endfor
return r
    
```

This is the right to left version (there exists a left to right one)

## Hardware Accelerators for Elliptic Curve Crypto.



## Attacks



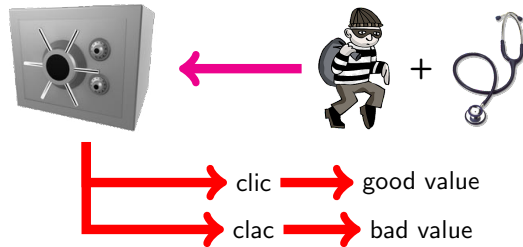
EMR = Electromagnetic radiation

## Side Channel Attacks (SCAs) (1/2)

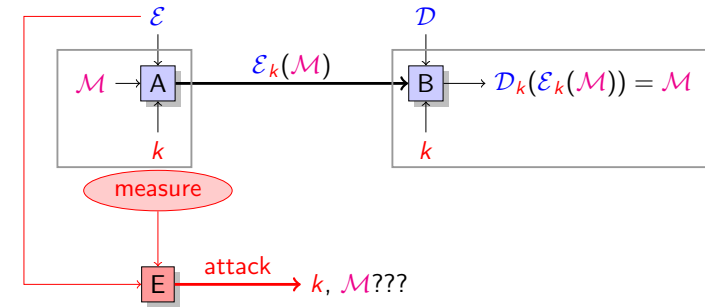
**Attack:** attempt to find, **without** any knowledge about the secret:

- the message (or parts of the message)
- informations on the message
- the secret (or parts of the secret)

“Old style” side channel attacks:



## Side Channel Attacks (SCAs) (2/2)



**General principle:** measure **external parameter(s)** on running device in order to deduce **internal informations**

## What Should be Measured?

**Answer:** **everything** that can “enter” and/or “get out” in/from the device

- power consumption
- electromagnetic radiation
- temperature
- sound
- computation time
- number of cache misses
- number and type of error messages
- ...

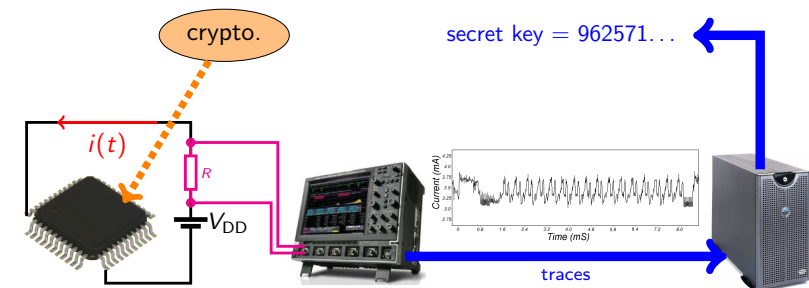
The measured parameters may provide informations on:

- **global** behavior (temperature, power, sound...)
- **local** behavior (EMR, # cache misses...)

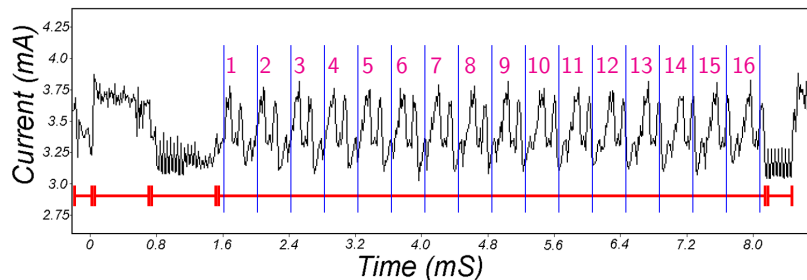
## Power Consumption Analysis

**General principle:**

1. measure the current  $i(t)$  in the cryptosystem
2. use those measurements to “deduce” secret informations



## “Read” the Traces



- algorithm → decomposition into steps
- detect loops
  - ▶ constant time for the loop iterations
  - ▶ non-constant time for the loop iterations

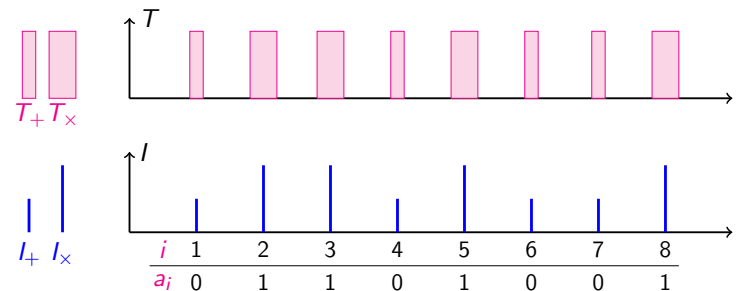
Source: [11] Kocher, Jaffe and Jun. **Differential Power Analysis**, Crypto99

## Differences & External Signature

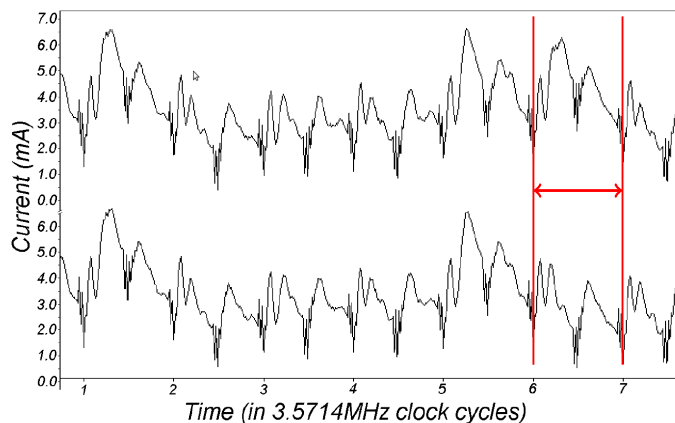
An algorithm has a **current signature** and a **time signature**:

```

r = c0
for i from 1 to n do
  if a_i = 0 then
    r = r + c1
  else
    r = r × c2
    
```



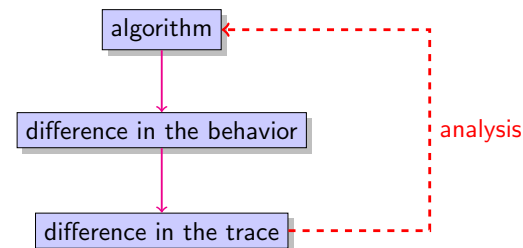
## Simple Power Analysis (SPA)



Source: [11]

## SPA in Practice

General principle:

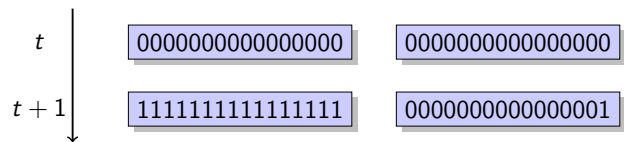


**Methods:** interpretation of the differences in

- control signals
- computation time
- operand values
- ...

## Limits of the SPA

Example of behavior difference: (activity into a register)

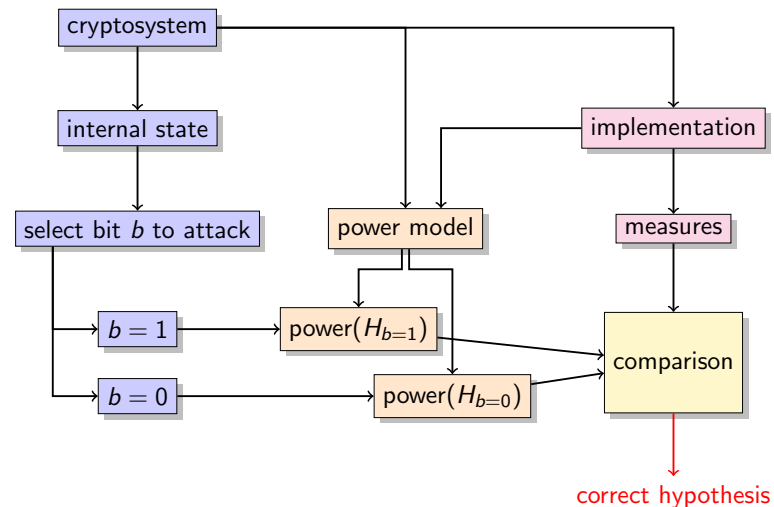


**Important:** a small difference may be evaluated as a **noise** during the measurement → traces cannot be distinguished

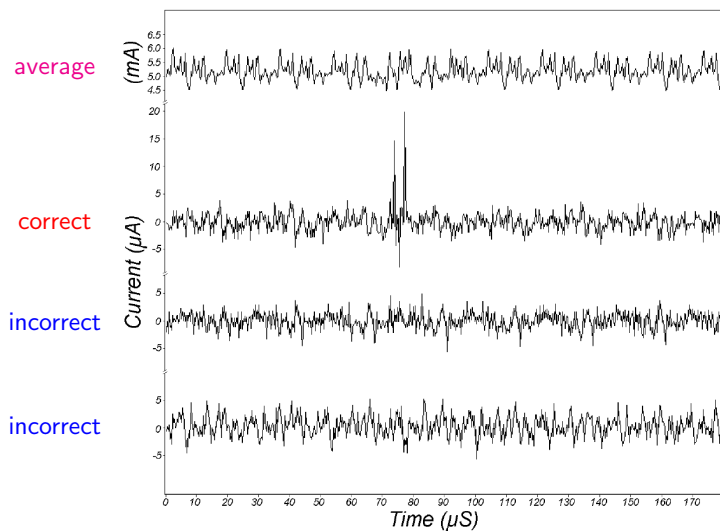
**Question:** what can be done when differences are too small?

**Answer:** use **statistics** over **several** traces

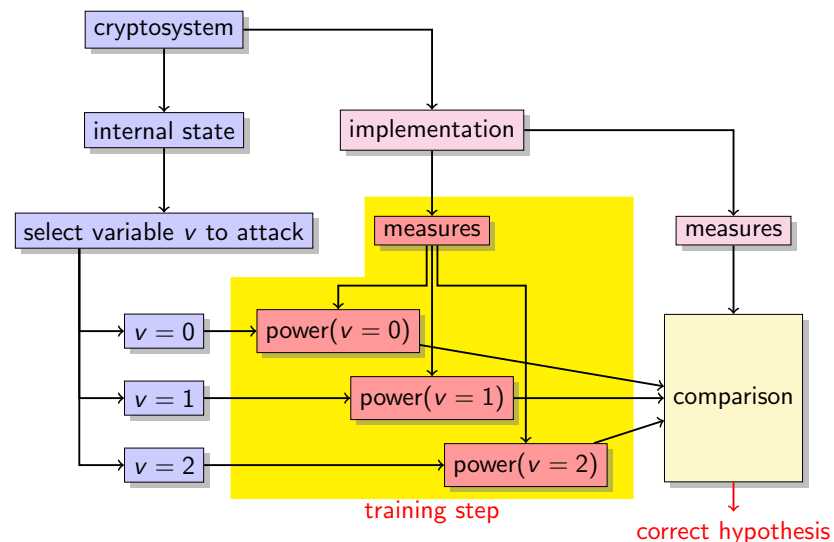
## Differential Power Analysis (DPA)



## Differential Power Analysis (DPA) Example

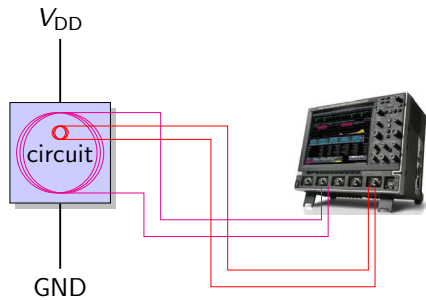


## Template Attack



# Electromagnetic Radiation Analysis (1/2)

**General principle:** use a **probe** to measure the EMR



**EMR measurement:**

- global EMR with a large probe
- local EMR with a micro-probe

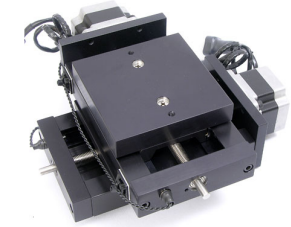
# Electromagnetic Radiation Analysis (2/2)

EMR analysis methods:

- simple electromagnetic analysis: SEMA
- differential electromagnetic analysis: DEMA

Local EMR analysis may be used to determine internal architecture details, and then select weak parts of the circuit for the attack

→ X-Y table



## Side Channel Attack on ECC

**protocol level**  
encryption, signature, etc

**curve level**  
[k]P

**field level**  
x ± y, x × y, ...

**Scalar multiplication operation**

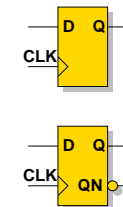
```

for i from 0 to t-1 do
  if ki = 1 then Q = ADD(P, Q)
  P = DBL(P)
  
```

• simple power analysis (& variants)  
• differential power analysis (& variants)  
• horizontal/vertical/templates/... attacks

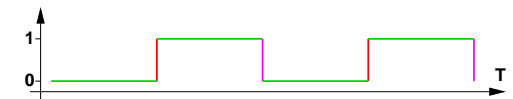
## Flip-Flops

There are many types of flip-flops, we will only focus on standard ones

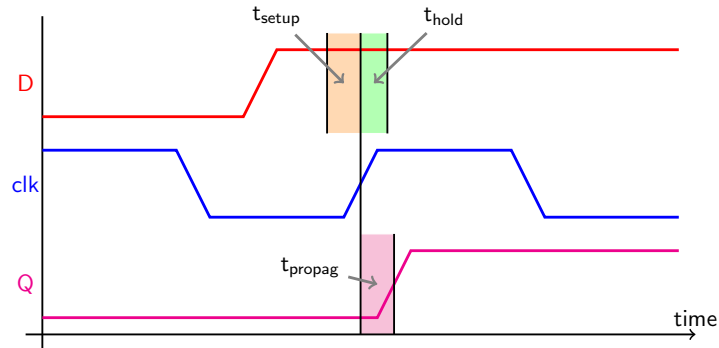


CLK	D	Q(t+1)	QN(t+1)
1	X	Q(t)	QN(t)
0	X	Q(t)	QN(t)
↑	0	0	1
↑	1	1	0

Remark:  
↑ is the rising clock edge



## Setup, Hold and Propagation Delays



- **setup** delay ( $t_{\text{setup}}$ ): data should be held steady *before* clock edge
- **hold** delay ( $t_{\text{hold}}$ ): data should be held steady *after* clock edge
- **propagation** delay ( $t_{\text{propag}}$ ): propagation time from D to Q

## Fault Injection Attacks

**Objective:** alter the correct functioning of a system “from outside”

**Fault effects examples:**

- modify a value in a register
- modify a value in the memory hierarchy
- modify an address (data location or code location)
- modify a control signal (e.g. status flag, branch direction)
- skip/modify the instruction decoding
- delay/advance propagation of internal control signals
- etc.

Also called **perturbation attacks**

## Presentation Scope

In this presentation we will only deal with **basic fault injection attacks at hardware level** (their principles and some state-of-the-art examples)

Not covered topics (even if they are very interesting):

- Denial of Service (DoS)
- (Pure) Software attacks (cache hierarchy, branch prediction, TLB, etc.)
- Fault attacks using “strong” invasive methods (e.g. probing, FIB, very accurate lasers)
- Advanced combinations of faults, observation and mathematical attacks

## Fault Targets in a Toy Code

```
100 integer length = 64
101 huser = hash(read_keyboard(), length)
102 href = get_hash_reference_password()
103 if equal(href, huser) then
104     secure_code()
105 else
106     error("unauthorized access")
107     exit()
108 other_secure_code()
```

## Fault Injection Techniques

### Typical techniques:

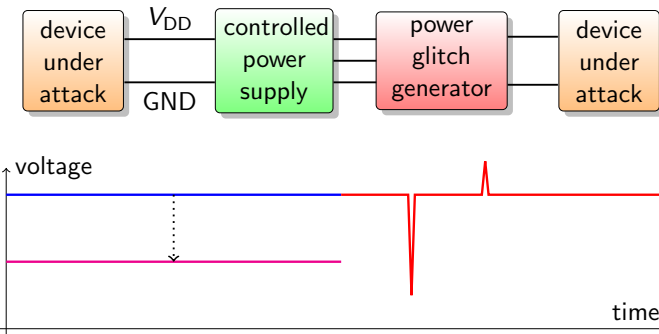
- perturbation in the power supply voltage
- perturbation of the clock signal
- temperature (over/under-heating the chip)
- radiation or electromagnetic (EM) disturbances
- exposing the chip to intense lights or beams
- etc

### Accuracy:

- **time**: part of clock cycle, clock cycle, code block (instruction sequence)
- **space**: gate, block, unit, core, chip, package
- **value**: set to a specific value, bit flip, stuck-at 0 or 1, random modification

## Perturbation on the Power Supply

### Principle:



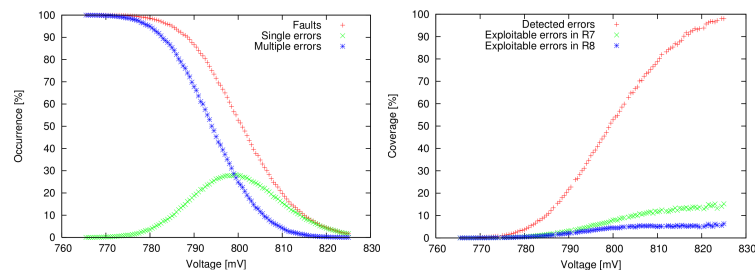
- **Nominal** power supply (e.g.  $\approx [0.7, 1.2]$  V for current technologies)
- **Non-nominal** constant power supply (e.g. 0.7V instead of 1.2V)
- **Glitches (dips, spikes)** in the power supply at some selected moments

## Under Powering Example

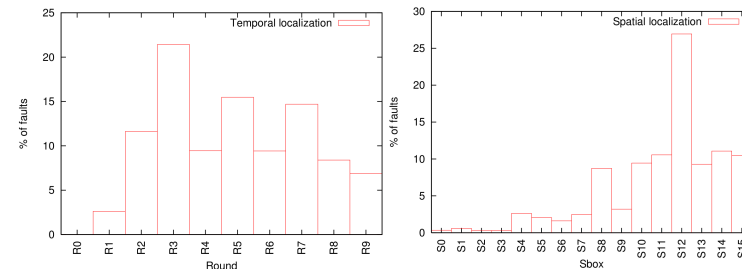
**Source:** paper [22] presented at EDCC 2008 conference

**Setup:** 130 nm smart card (1.2 V nominal  $V_{DD}$ ) with AES crypto-processor

**Measurement campaign:** triples (msg, key, cypher) recorded for 100  $V_{DD}$  in [775, 825] mV over 20,000 encryptions with comparison to a (RTL) simulation for one byte corruption in the state matrix at various rounds



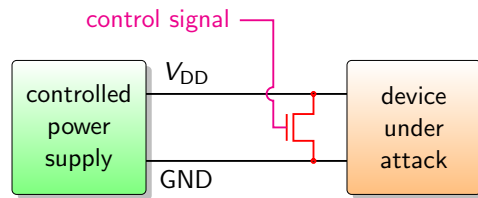
Observed behavior is compatible with setup violation model on a critical path (bell shape due to only one or multiple paths)



More details in 2010 PhD thesis [21]

## Simple Power Glitch Generator

Dips in the power supply can be “easily” generated by a short circuit between the power lines  $V_{DD}$  and GND using a transistor (e.g. MOSFET)

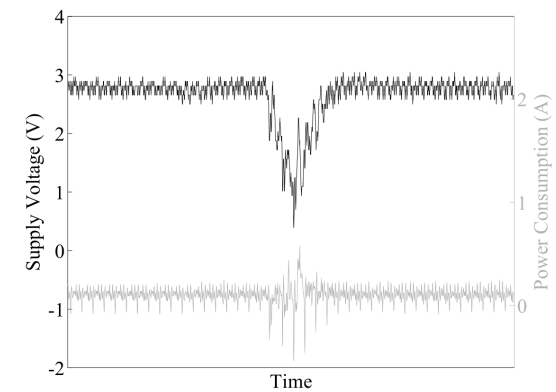


See example in IACR Eprint article [13] (attack on 8-bit AVR microcontroller)

## Power Glitching Example

Source: FDTC 2008 conference paper [20]

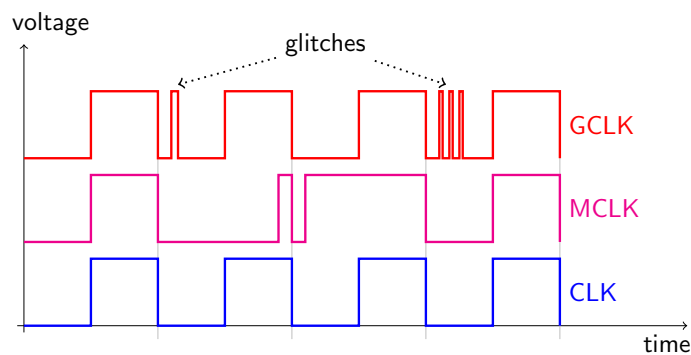
Setup: AVR microcontroller with RSA implementation



Attack result: a power glitch causes to skip some instruction

## Perturbation on the External Clock

Principle:



- Normal clock (at a given frequency, duty cycle  $\approx 50\%$ )
- Clock with a modified duty cycle
- Glitched clock
- Etc.

## Glitchy Clock Generation Example

Source: paper [8] published in J. Crypto. Eng. 2011

Setup: Virtex-II Pro FPGA (on SASEBO card) used to generate a “glitchy” clock for several programmable time parameters

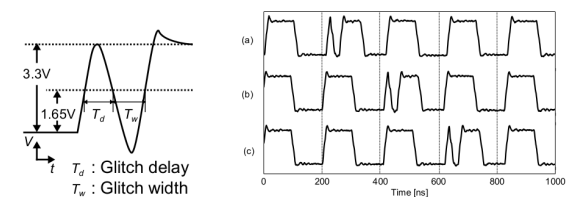


Fig. 3. Image of glitchy-clock cycle.

Fig. 4. Examples of glitchy-clock signals.

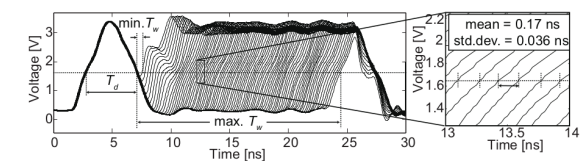


Fig. 5. Waveforms of glitchy-clock cycles for different glitch widths.



## Clock Glitch Attack Example

**Source:** paper [1] presented at FDTC 2011 conference

**Setup:** AVR ATmega 163 microcontroller @ 1MHz

mode	glitch period	cycle	instruction	opcode (bin)
normal	-	$i$	NOP	0000 0000 0000 0000
normal	-	$i + 1$	EOR R15,R5	0010 0100 1111 0101
glitch	59 ns	$i + 1$	NOP	0000 0000 0000 0000

mode	glitch period	cycle	instruction	opcode (bin)
normal	-	$i$	NOP	0000 0000 0000 0000
normal	-	$i + 1$	SER R18	1110 1111 0010 1111
glitch	61 ns	$i + 1$	LDI R18,0xEF	1110 1110 0010 1111
glitch	60 ns	$i + 1$	SBC R12,R15	0000 1000 0010 1111
glitch	59 ns	$i + 1$	NOP	0000 0000 0000 0000

mode	glitch period	cycle	instruction	opcode (bin)
normal	-	$i$	TST R12	0010 0000 1100 1100
normal	-	$i + 1$	BREQ PC+0x02	1111 0000 0000 1001
normal	-	$i + 2$	SER R26	1110 1111 1010 1111
glitch	57 ns	$i + 2$	LDI R26,0xEF	1110 1110 1010 1111
glitch	56 ns	$i + 2$	LDI R26,0xCF	1110 1100 1010 1111
glitch	52 ns	$i + 2$	LDI R26,0xF	1110 0000 1010 1111
glitch	45 ns	$i + 2$	LDI R16,0x09	1110 0000 0000 1001
glitch	32 ns	$i + 2$	LD R0,Y+0x01	1000 0000 0000 1001
glitch	28 ns	$i + 2$	LD R9,Y	1000 0000 0000 1000
glitch	27 ns	$i + 2$	LDI R16,0x09	1110 0000 0000 1001
glitch	15 ns	$i + 2$	BREQ PC+0x02	1111 0000 0000 1001

## Temperature

Temperature can be used for two types of attacks:

- as a **fault injection method**
  - ↪ temperature **impacts** current in the circuit (blocks)
- as a **side channel** for analysis
  - ↪ current in the circuit (blocks) **impacts** chip temperature

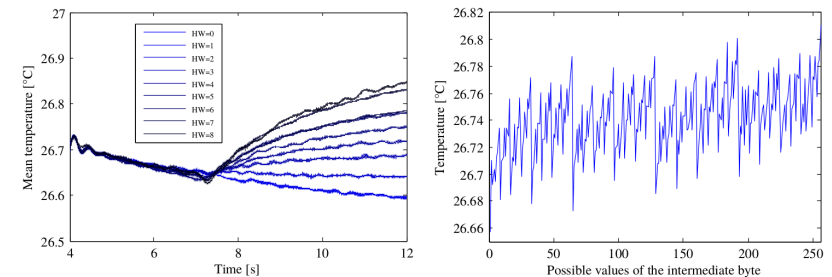
**Limits:**

- Very slow variations (e.g. leakage @ bit/minute)
- Very coarse space accuracy

## Temperature Attacks Examples

**Source:** article [9] presented at CARDIS 2013 conference

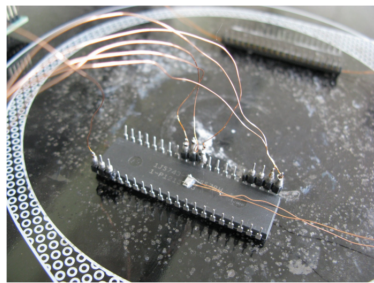
**Setup:** ATmega162 microcontroller, PT100 thermometer circuit (100 ms response time and 0.01 °C resolution), RSA implementation



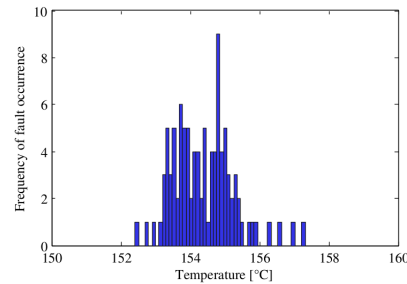
**Fig. 4:** Slow temperature increase of all the ATmega162.

**Fig. 5:** The ATmega162 leaks the Hamming weights that are processed by the ATmega162.

Heating the MCU around 150–160 °C  $\implies$  around 100 faults are injected during RSA decryptions (every 650 ms during 70 minutes), where about 31 can be exploited to guess secret bits of the exponent

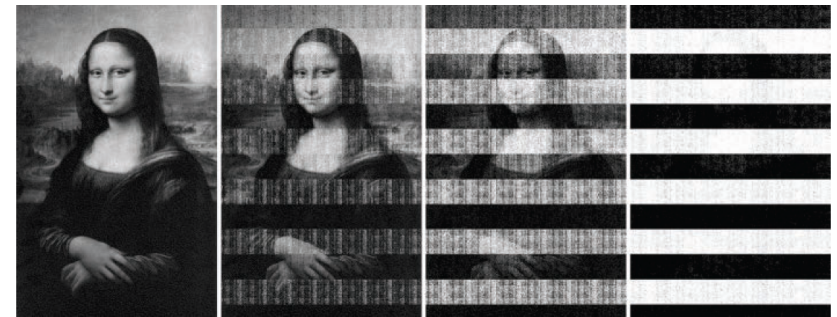


**Fig. 6:** Heating plate with two PT100 sensors measuring the rear-side and front-side temperature of an ATmega162.



**Fig. 7:** Distribution of fault occurrence between 150 and 160 °C. Mean fault-induction temperature is 154.4 °C.

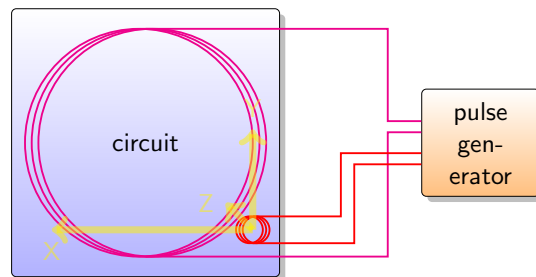
Figure 3: Visualizing memory decay. We loaded a bitmap image into memory on test machine A, then cut power for varying intervals. After 5 s (left), the image is nearly indistinguishable from the original; it gradually becomes more degraded, as shown after 30, 60 s, and 5 min. The chips remained close to room temperature. Even after this longest trial, traces of the original remain. The decay shows prominent patterns caused by regions with alternating ground states (horizontal bars) and by physical variations in the chip (fainter vertical bands).



By cooling (freezing) the memory, it can be read a “long” time after powering off the circuit

## Electromagnetic Perturbations

**Principle:**



- large antenna
- micro-antenna with motorized (X,Y,Z) stage/table

## Electromagnetic Attack Example

**Source:** article [12] presented at FDTC 2013 conference

**Setup:** 32-b Cortex-M3 ARM microprocessor (CMOS 130 nm SoC at 56 MHz), magnetic antenna with pulses in [-200, 200] V and [10, 200] ns

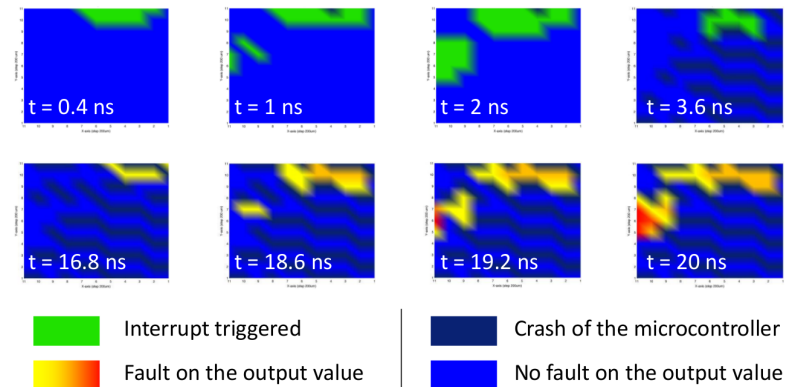


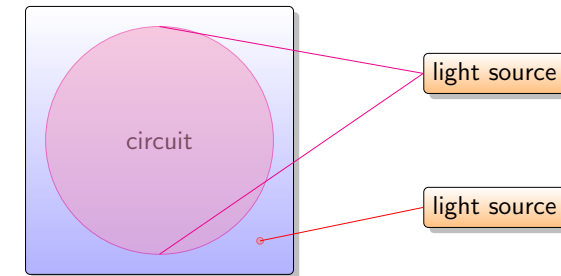
Figure 3: Impact of the probe's position

## Lights / Lasers

Loaded value: 12345678

Pulse voltage [V]	Loaded value	Occurrence rate [%]
170	1234 5678	100
172	1234 5678	100
174	9234 5678	73
176	FE34 5678	30
178	FFF4 5678	53
180	FFFD 5678	50
182	FFFF 7F78	46
184	FFFF FFFB	40
186	FFFF FFFF	100
188	FFFF FFFF	100
190	FFFF FFFF	100

**Principle:**



- large illuminated area (flash light with microscope)
- small "spot" (laser with variable locations)

## Differential Fault Analysis

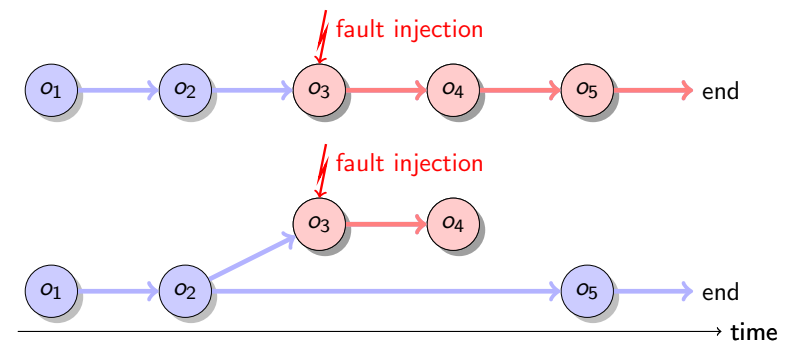
Most of time, exploiting only one fault does not provide enough information

- Accurately injecting fault is difficult
- The fault causes a few perturbations

Then, use statistical correlation(s)

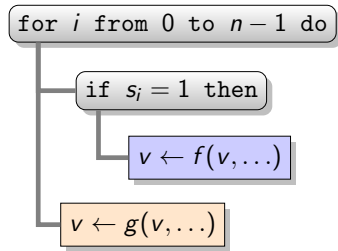
## Safe Error Attack

**Principle:** exploit the link (or the lack of link) between injected fault(s) during "useful" (or "useless") operations and the final result

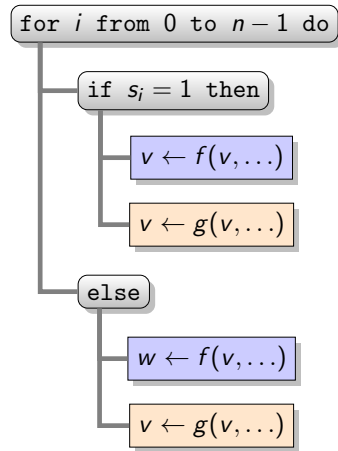


## Safe Error Attack Example in Asymmetric Crypto

WEAK against SPA

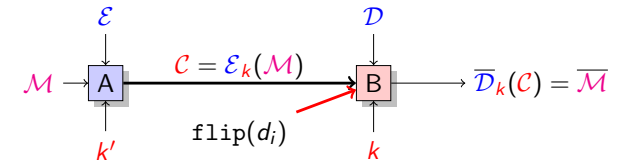


WEAK against SEA



Useless or dummy operations are a bad idea

## Fault Attack Example: Bit Flip on RSA Decryption



- choose a plaintext message  $\mathcal{M}$
- encrypt  $\mathcal{M}$  into  $C = \mathcal{E}_k(\mathcal{M})$
- inject a fault by flipping  $d_i$  for a random  $i$  ( $d$  is the secret key)
- compute  $\overline{\mathcal{M}} = \frac{c^{2^i d_i}}{c^{2^i}}$
- test:
  - ▶  $\frac{\overline{\mathcal{M}}}{\mathcal{M}} = \frac{1}{c^{2^i}} \pmod N \implies d_i = 1$
  - ▶  $\frac{\overline{\mathcal{M}}}{\mathcal{M}} = c^{2^i} \pmod N \implies d_i = 0$
- retry for several  $i$  ( $\implies$  get small parts of  $d$ , then mathematical attacks)

## Countermeasures

### Principles for preventing attacks:

- embed additional protection blocks
- modify the original circuit into a secured version
- application levels: circuit, architecture, algorithm, protocol...

### Countermeasures:

- electrical shielding
- detectors, estimators, decoupling
- use uniform computation durations and power consumption
- use detection/correction codes (for fault injection attacks)
- provide a random behavior (algorithms, representation, operations...)
- add noise (e.g. masking, useless instructions/computations)
- circuit reconfiguration (algorithms, block location, representation of values...)

Many other fault attacks...

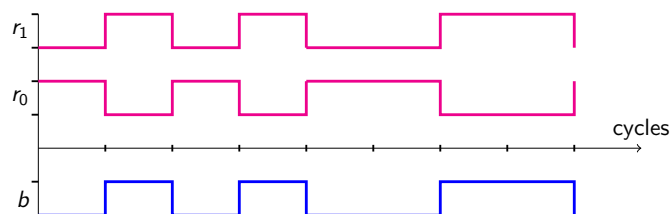
## Low-Level Coding and Circuit Activity

### Assumptions:

- $b$  is a bit (i.e.  $b \in \{0, 1\}$ , logical or mathematical value)
- electrical states for a wire  $\blacksquare$  :  $V_{DD}$  (logical 1) or GND (logical 0)

### Low-level codings of a bit:

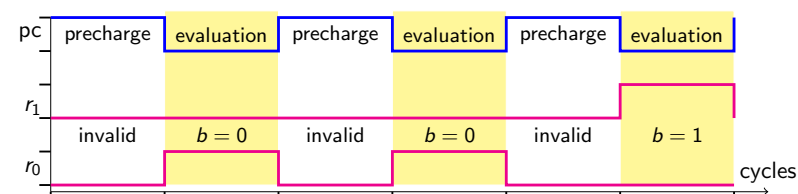
	$b = 0$	$b = 1$
standard	$\blacksquare$ GND	$\blacksquare$ $V_{DD}$
dual rail	$\blacksquare$ $r_0 = V_{DD}$ $\blacksquare$ $r_1 = GND$ ] $(1, 0)_{DR}$	$\blacksquare$ $r_0 = GND$ $\blacksquare$ $r_1 = V_{DD}$ ] $(0, 1)_{DR}$



## Circuit Logic Styles

**Countermeasure principles:** **uniformize** circuit activity and **exclusive** coding

### Solution based on precharge logic and dual-rail coding:



### Solution based on validity line and dual-rail coding:

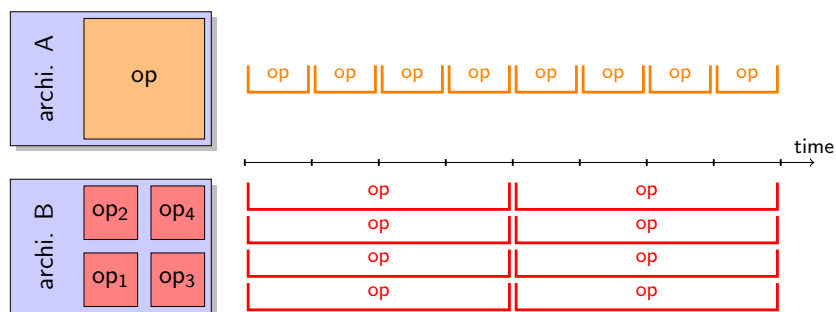


**Important overhead:** silicon area and local storage (registers)

## Countermeasure: Architecture

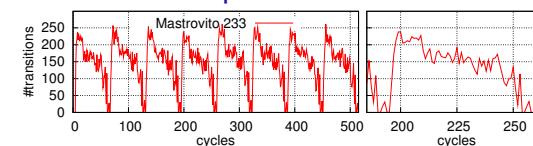
### Increase internal parallelism:

- replace one fast but big operator
- by several instances of a small but slow one

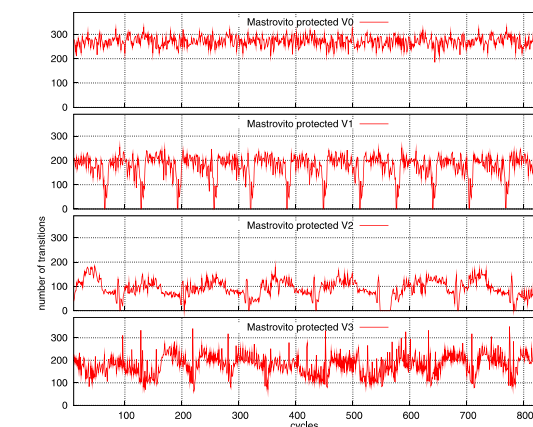


## Protected Multipliers

Unprotected



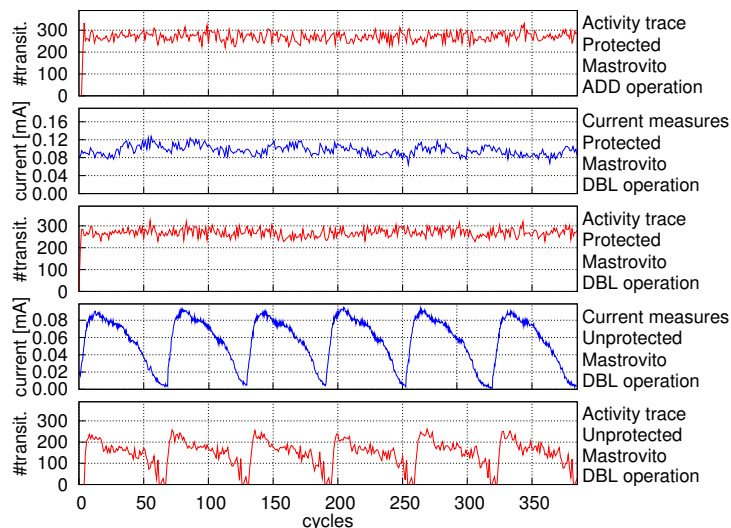
Protected



Overhead:  
Area/time < 10 %

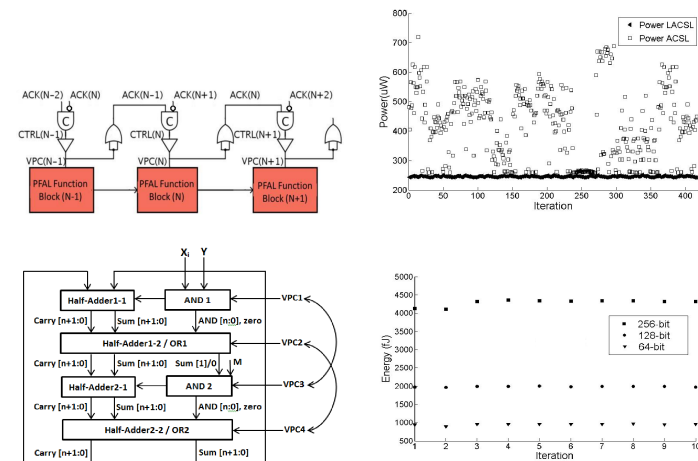
References:  
PhD D. Pamula [14]  
Articles: [17], [16],  
[15]

## Protected (Old) Accelerator

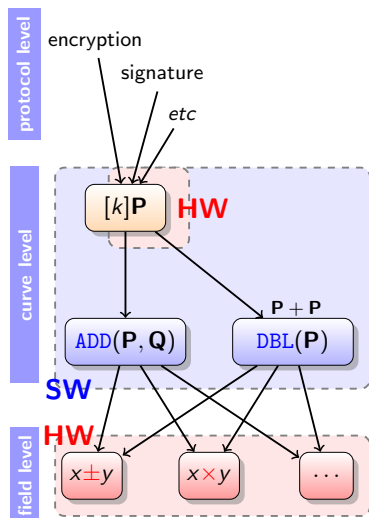


Warning: old dedicated accelerator (similar behavior is expected for our new one)

## Circuit-Level Protections for Arithmetic Operators

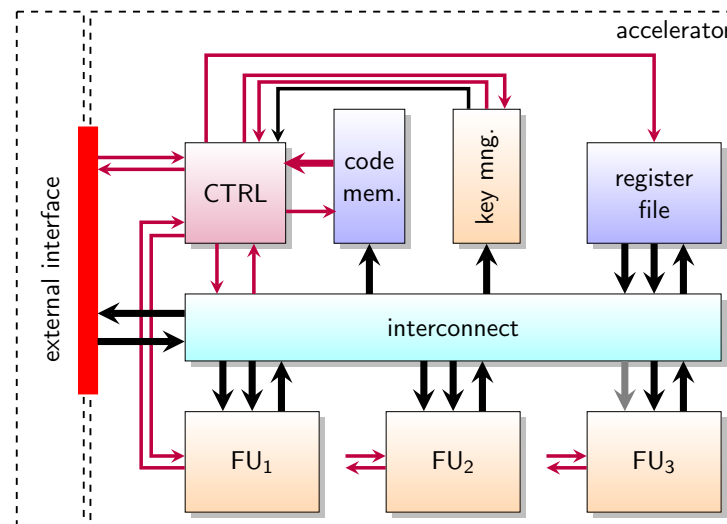


## Accelerator Specifications



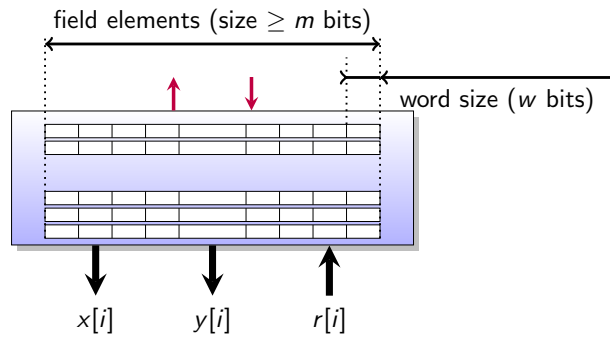
- Performances  $\Rightarrow$  hardware (HW)
  - ▶ dedicated functional units
  - ▶ internal parallelism
- Limited cost (embedded systems)
  - ▶ reduced silicon area
  - ▶ low energy (& power consumption)
  - ▶ large area used at each clock cycle
- Flexibility  $\Rightarrow$  software (SW)
  - ▶ curves, algorithms, representations (points/elements),  $k$  recoding, ...
  - ▶ at design time / at run time
- Security against SCAs  $\Rightarrow$  HW
  - ▶ secure units ( $GF(2^m)$ ,  $GF(p)$ )
  - ▶ secure key storage/management
  - ▶ secure control

## Accelerator Architecture



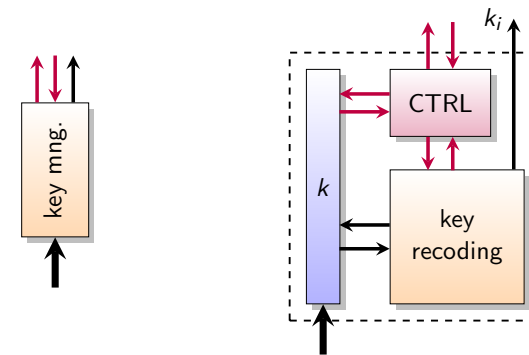
Data:  $w$ -bit (32, ..., 128) except for  $k$  digits, control: a few bits per unit

## Register File ( $\approx$ Dual Port Memory)



- Control signals: addresses (port A, port B), read/write, write enable
- Specific addressing model for  $GF(q)$  elements (through an intermediate address table with hardware loop)
- linear addresses, SW:  $LOAD @x \implies$  HW: loop  $x[0], x[1], \dots, x[\ell - 1]$
  - randomized addresses

## Key Management Unit

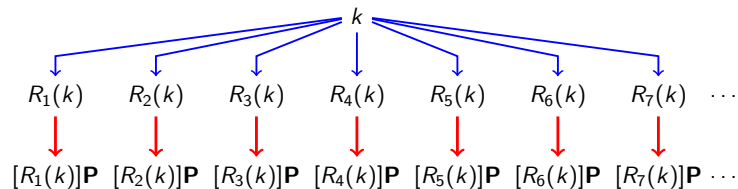


- **On-the-fly recoding** of  $k$ : binary,  $\lambda$ -NAF ( $\lambda \in \{2, 3, 4, 5\}$ ), variants (fixed/sliding), double-base [4] and multiple-base [5] number systems (w/wo randomization), addition chains [18], other ?
- Specific private path in the interconnect (no key leaks in RF or FUs)

## Arithmetic Level Countermeasures

Redundant number system =

- a way to improve the performance of some operations
- a way to **represent a value with different representations**



**Important property:**  $\forall i \quad [R_i(k)]P = [k]P$

**Proposed solution:** use **random redundant representations** of  $k$

## Double-Base Number System

Standard radix-2 representation:

$$k = \sum_{i=0}^{t-1} k_i 2^i = \begin{matrix} 2^{t-1} & 2^{t-2} & \dots & 2^2 & 2^1 & 2^0 & \text{implicit weights} \\ \boxed{k_{t-1}} & \boxed{k_{t-2}} & \dots & \boxed{k_2} & \boxed{k_1} & \boxed{k_0} & t \text{ explicit digits} \end{matrix}$$

Digits:  $k_i \in \{0, 1\}$ , typical size:  $t \in \{160, \dots, 600\}$

Double-Base Number System (DBNS):

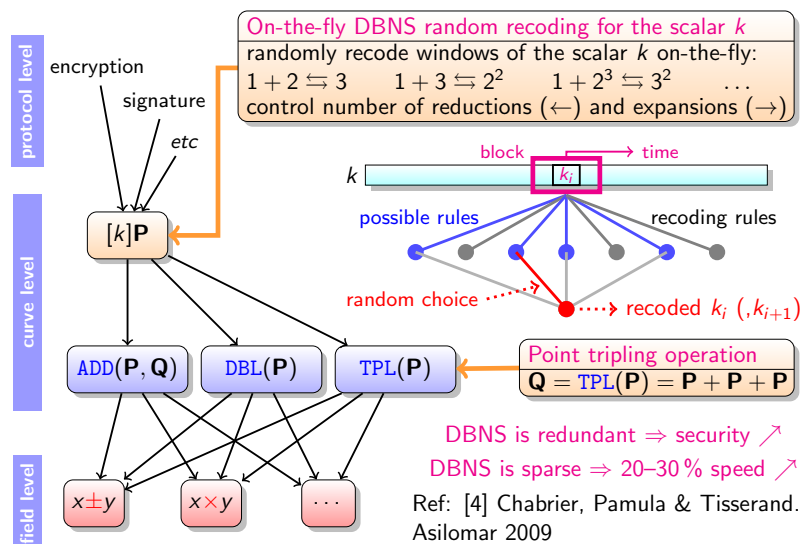
$$k = \sum_{j=0}^{n-1} k_j 2^{a_j} 3^{b_j} = \begin{matrix} k_{n-1} & \dots & k_1 & k_0 & n \text{ (2,3)-terms} \\ \boxed{a_{n-1}} & \dots & \boxed{a_1} & \boxed{a_0} & \text{explicit "digits"} \\ \boxed{b_{n-1}} & \dots & \boxed{b_1} & \boxed{b_0} & \text{explicit ranks} \end{matrix}$$

$a_j, b_j \in \mathbb{N}$ ,  $k_j \in \{1\}$  or  $k_j \in \{-1, 1\}$ , size  $n \approx \log t$

DBNS is a very **redundant** and **sparse** representation:  $1701 = (11010100101)_2$

$$\begin{aligned} 1701 &= 243 + 1458 = 2^0 3^5 + 2^1 3^6 = (1, 0, 5), (1, 1, 6) \\ &= 1728 - 27 = 2^6 3^3 - 2^0 3^3 = (1, 6, 3), (-1, 0, 3) \\ &= 729 + 972 = 2^0 3^6 + 2^2 3^5 = (1, 0, 6), (1, 2, 5) \\ &\dots \end{aligned}$$

## Randomized DBNS Recoding of the Scalar $k$



## Conclusion

- Side channel and fault attacks are **serious threats**
- **Attacks** are more and more **efficient** (many variants)
- Security analysis is mandatory at **all levels** (specification, algorithm, operation, implementation)
- Security = **trade-off** between performances, robustness and cost
- Security = *func*( secret value, attacker capabilities )
- **security** = **computer science + microelectronics + mathematics**

### Current works examples:

- Methods/tools for automating security analysis
- Circuit reconfiguration (representations, algorithms)
- Circuits with reduced activity variations
- Representation of numbers with error detection/correction “codes”
- Design space exploration
- CAD tools with security improvement capabilities

## Resources: Conferences, Workshops, Journals, etc

- International Association for Cryptologic Research (IACR) Eprint Archives
- ACM Special Interest Group on Security, Audit and Control (SIGSAC)
- IEEE Computer Society's Technical Committee on Security and Privacy (TCSP)
- French national working group on Code & Crypto (C2) of the GDR IM
- French national working group on Security of Embedded Systems of the GDR SoC-SiP
- Conferences, workshops: CHES, FDTTC, COSADE, CARDIS, CryptArchi ...
- Journals: Journal of Cryptographic Engineering, IEEE Trans. on Computers, Circuits and Systems, VLSI Systems, ...

## References I

- Surveys: Proc. IEEE 2006 [2], Proc. IEEE 2012 [3], IEEE TVLSI 2013 [10]
- [1] J. Balasch, B. Gierlichs, and I. Verbauwhede. An in-depth and black-box characterization of the effects of clock glitches on 8-bit MCUs. In *Proc. 8th International Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, pages 105–114, Nara, Japan, September 2011. IEEE.
  - [2] H. Bar-EI, H. Choukri, D. Naccache, M. Tunstall, and C. Whelan. The sorcerer's apprentice guide to fault attacks. *Proceedings of the IEEE*, 94(2):370–382, February 2006.
  - [3] A. Barengi, L. Breveglieri, I. Koren, and D. Naccache. Fault injection attacks on cryptographic devices: Theory, practice, and countermeasures. *Proceedings of the IEEE*, 100(11):3056–3076, November 2012.
  - [4] T. Chabrier, D. Pamula, and A. Tisserand. Hardware implementation of DBNS recoding for ECC processor. In *Proc. 44th Asilomar Conference on Signals, Systems and Computers*, pages 1129–1133, Pacific Grove, California, U.S.A., November 2010. IEEE.
  - [5] T. Chabrier and A. Tisserand. On-the-fly multi-base recoding for ECC scalar multiplication without pre-computations. In A. Nannarelli, P.-M. Seidel, and P. T. P. Tang, editors, *Proc. 21st Symposium on Computer Arithmetic (ARITH)*, pages 219–228, Austin, TX, U.S.A, April 2013. IEEE Computer Society.
  - [6] J. Chen, A. Tisserand, E. M. Popovici, and S. Cotozana. Robust sub-powered asynchronous logic. In J. Becker and M. R. Adrover, editors, *Proc. 24th International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, pages 1–7, Palma de Mallorca, Spain, September 2014. IEEE.
  - [7] J. Chen, A. Tisserand, E. M. Popovici, and S. Cotozana. Asynchronous charge sharing power consistent Montgomery multiplier. In J. Spaso and E. Yahya, editors, *Proc. 21st IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*, pages 132–138, Mountain View, California, USA, May 2015.



## References II

- [8] S. Endo, T. Sugawara, N. Homma, T. Aoki, and A. Satoh.  
An on-chip glitchy-clock generator for testing fault injection attacks.  
*Journal of Cryptographic Engineering*, 1(4):265–270, December 2011.
- [9] M. Hutter and J.-M. Schmidt.  
The temperature side channel and heating fault attacks.  
In A. Francillon and P. Rohatgi, editors, *Proc. 12th International Conference on Smart Card Research and Advanced Applications (CARDIS)*, volume 8419 of *LNCS*, pages 219–235, Berlin, Germany, November 2013.
- [10] D. Karaklajic, J.-M. Schmidt, and I. Verbauwhede.  
Hardware designer's guide to fault attacks.  
*IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 21(12):2295–2306, December 2013.
- [11] P. C. Kocher, J. Jaffe, and B. Jun.  
Differential power analysis.  
In *Proc. Advances in Cryptology (CRYPTO)*, volume 1666 of *LNCS*, pages 388–397. Springer, August 1999.
- [12] N. Moro, A. Dehbaoui, K. Heydemann, B. Robisson, and E. Encrenaz.  
Electromagnetic fault injection: Towards a fault model on a 32-bit microcontroller.  
In *Proc. 10th International Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, pages 77–88, Santa Barbara, CA, USA, August 2013. IEEE.
- [13] C. O'Flynn.  
Fault injection using crowbars on embedded systems.  
Technical Report 810, IACR Cryptology ePrint Archive, August 2016.
- [14] D. Pamula.  
*Arithmetic Operators on  $GF(2^m)$  for Cryptographic Applications: Performance - Power Consumption - Security Tradeoffs*.  
Phd thesis, University of Rennes 1 and Silesian University of Technology, December 2012.
- [15] D. Pamula, E. Hryniewicz, and A. Tisserand.  
Analysis of  $GF(2^{23})$  multipliers regarding elliptic curve cryptosystem applications.  
In *11th IFAC/IEEE International Conference on Programmable Devices and Embedded Systems (PDeS)*, pages 271–276, Brno, Czech Republic, May 2012.

## References III

- [16] D. Pamula and A. Tisserand.  
 $GF(2^m)$  finite-field multipliers with reduced activity variations.  
In *4th International Workshop on the Arithmetic of Finite Fields*, volume 7369 of *LNCS*, pages 152–167, Bochum, Germany, July 2012. Springer.
- [17] D. Pamula and A. Tisserand.  
Fast and secure finite field multipliers.  
In *Proc. 18th Euromicro Conference on Digital System Design (DSD)*, pages 653–660, Madeira, Portugal, August 2015.
- [18] J. Prouy, N. Veyrat-Charvillon, A. Tisserand, and N. Meloni.  
Full hardware implementation of short addition chains recoding for ECC scalar multiplication.  
In *Actes Conférence d'informatique en Parallélisme, Architecture et Système (ComPAS)*, Lille, France, June 2015.
- [19] R. L. Rivest, A. Shamir, and L. Adleman.  
A method for obtaining digital signatures and public-key cryptosystems.  
*Communications of the ACM*, 21(2):120–126, February 1978.
- [20] J. Schmidt and C. Herbst.  
A practical fault attack on square and multiply.  
In *Proc. 5th International Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, pages 53–58, Washington, DC, USA, August 2008. IEEE.
- [21] N. Selmane.  
*Attaques en fautes globales et locales sur les cryptoprocresseurs AES : mise en œuvre et contremesures*.  
Phd thesis, Télécom ParisTech, December 2010.
- [22] N. Selmane, S. Guilley, and J.-L. Danger.  
Practical setup time violation attacks on AES.  
In *Proc. 7th European Dependable Computing Conference (EDCC)*, Kaunas, Lithuania, 2008.

## Good Books (in French)

### Histoire des codes secrets

Simon Singh

1999

Livre de poche



## Good Books (in French)

### Cryptographie appliquée

Bruce Schneier

1997, 2ème édition

Wiley

ISBN: 2–84180–036–9

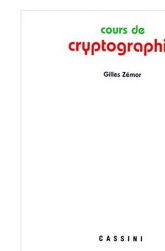


### Mathématiques, espionnage et piratage informatique

Joan Gomez

2010

Le monde est mathématique, RBA



### Cours de cryptographie

Gilles Zémor

2000

Cassini

ISBN: 2–84225–020–6

## Good Books (in French)

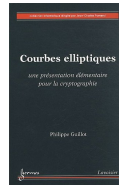
### Courbes elliptiques

Philippe Guillot

2010

Hermes

ISBN: 978-2-7462-2392-9



### Micro et nano-électronique

*Bases, Composants, Circuits*

Hervé Fanet

2006

Dunod

ISBN: 2-10-049141-5



## Good Books (in English)

### CMOS VLSI Design

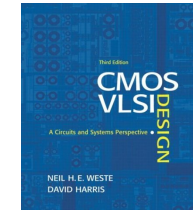
*A Circuits and Systems Perspective*

Neil Weste and David Harris

3rd edition, 2004

Addison Wesley

ISBN: 0-321-14901-7



### Power Analysis Attacks

*Revealing the Secrets of Smart Cards*

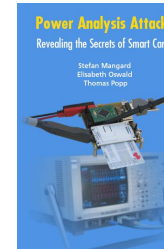
Stefan Mangard, Elisabeth Oswald and

Thomas Popp

2007

Springer

ISBN:978-0-387-30857-9



## Good Books (in English)

### Handbook of Applied Cryptography

Alfred J. Menezes, Paul C. van Oorschot and

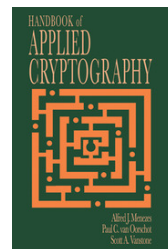
Scott A. Vanstone

2001

CRC Press

ISBN:0-8493-8523-7

Web: <http://cacr.uwaterloo.ca/hac/>



## The end, questions ?

Contact:

- <mailto:arnaud.tisserand@univ-ubs.fr>
- <http://www-labsticc.univ-ubs.fr/~tisseran>
- CNRS, Lab-STICC Laboratory  
University South Brittany (UBS),  
Centre de recherche C. Huygens, rue St Maudé, BP 92116,  
56321 Lorient cedex, France

Thank you