Fault Injection Attacks and Countermeasures in Embedded Processors

Arnaud Tisserand
CNRS, Lab-STICC laboratory
ARCHI’17, Nancy

Applications with Security Needs

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

Summary

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

Security Aspects

- security
- system security
- cryptology
  - steganography
  - cryptography
  - cryptanalysis
  - physical
  - theoretical
- data
- networks
- operating systems
- programs
- devices
Software vs Hardware Support

![Diagram showing memory hierarchy and instruction management]

<table>
<thead>
<tr>
<th>EXCELLENT</th>
<th>FLEXIBILITY</th>
<th>SPEED</th>
<th>AREA</th>
<th>ENERGY</th>
<th>DEVEL. COST</th>
</tr>
</thead>
<tbody>
<tr>
<td>slow</td>
<td>limited</td>
<td>fast</td>
<td>small</td>
<td>small</td>
<td>huge</td>
</tr>
</tbody>
</table>

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
- M: plain text/message
- E: encryption/ciphering algorithm, D: decryption/deciphering algorithm
- k: secret key to be shared by A and B
- $E_k(M)$: encrypted text
- $D_k(E_k(M))$: decrypted text
- Eve: eavesdropper/spy
## Symmetric Cryptography Limitation

<table>
<thead>
<tr>
<th>$n$</th>
<th>List</th>
<th>Required Keys</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>A, B</td>
<td>$k$</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>A, B, C</td>
<td>$k_1, k_2, k_3$</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>A, B, C, D</td>
<td>$k_1, k_2, k_3, k_4, k_5$</td>
<td>6</td>
</tr>
</tbody>
</table>

$n$ A, ... $\frac{n(n-1)}{2}$

## Asymmetric / Public-Key Cryptography

![Diagram]

- $k$: B's public key (known to everyone including E)
- $E_k(M)$: ciphered text
- $k'$: B's private key (must be kept secret)
- $D_{k'}(E_k(M))$: 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

<table>
<thead>
<tr>
<th>key length</th>
<th>#round</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>10</td>
</tr>
<tr>
<td>192</td>
<td>12</td>
</tr>
<tr>
<td>256</td>
<td>14</td>
</tr>
</tbody>
</table>

Based on substitution-permutation network


### NIST: National Institute of Standards and Technology

Arnaud Tisserand. CNRS – Lab-STICC. Fault Injection Attacks and Countermeasures in Embedded Processors, ARCHI’17 13/88

## 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 \( \gcd(e, \varphi(n)) = 1 \)
- Compute \( d = e - 1 \mod \varphi(n) \)
- **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 \( m \) \( (1 < m < n \text{ and } \gcd(m, n) = 1) \)
- compute the cipher text \( c = m^e \mod n \)

**Decryption** (Alice side):
- compute \( m = c^d \mod n \)
- 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

Arnaud Tisserand. CNRS–Lab-STICC. Fault Injection Attacks and Countermeasures in Embedded Processors, ARCHI’17 16/88
Modular Exponentiation

Computation of operations such as: \( a^b \mod n \)

\[
a^b = a \times a \times a \times a \times \ldots \times a \times a
\]

\( a \) appears \( b \) times

Order of magnitude of exponents: \( 2^{\text{size of exponent}} \approx 2^{1024} \ldots 2^{2048} \ldots 2^{4096} \)

Fast exponentiation principle:

\[
a^b = \begin{cases} 
(a^2)^{b/2} & \text{when } b \text{ is even} \\
(a^{2^{b-1}})^{2^{t-1-b}} & \text{when } b \text{ is odd}
\end{cases}
\]

Least significant bit of the exponent: \( \text{bit} = 0 \Rightarrow \text{even} \) and \( \text{bit} = 1 \Rightarrow \text{odd} \)

Square and Multiply Algorithm

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

**output:** \( a^b \mod n \)

\[
r = 1
\]

for \( i \) from 0 to \( t-1 \) do

if \( b_i = 1 \) then

\[
r = r \cdot a \mod n
\]

\[
a = a^2 \mod n
\]

end if

end for

return \( r \)

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

Hardware Accelerators for Elliptic Curve Crypto.

\[ E : y^2 = x^3 + 4x + 20 \text{ over } \text{GF}(1009) \]

points: \( P, Q = (x, y) \) or \( (x, y, z) \) or \ldots

coordinates: \( x, y, z \in \text{GF}(\cdot) \)

\( \text{GF}(p), \text{GF}(2^m), t : 200–600 \text{ bits} \)

\( k = (k_{t-1}k_{t-2} \ldots k_1k_0)_2 \in \mathbb{N} \)

Scalar multiplication operation

\[
\text{for } i \text{ from } 0 \text{ to } t-1 \text{ do} \\
\text{if } k_i = 1 \text{ then } Q = \text{ADD}(P, Q)
\]

\[
P = \text{DBL}(P)
\]

Point addition/doubling operations

sequence of finite field operations

\[
\text{DBL: } v_1 = z_1^2, v_2 = x_1 - v_1, \ldots
\]

\[
\text{ADD: } w_1 = z_1^2, w_2 = x_1 \times w_1, \ldots
\]

\( \text{GF}(p) \text{ or } \text{GF}(2^m) \) operations

operation modulo large prime \((\text{GF}(p))\)

or irreducible polynomial \((\text{GF}(2^m))\)

Attacks

- **EMR** = Electromagnetic radiation

- **timing analysis**, **power analysis**, **EMR analysis**

- **attack**, **perturbation**, **fault injection**

- **theoretical**, **invasive**, **optimized programming**

- **advanced algorithms**, **probing**, **reverse engineering**

- **Arnaud Tisserand, CNRS – Lab-STICC. Fault Injection Attacks and Countermeasures in Embedded Processors, ARCHI’17** 17/88

- **Arnaud Tisserand, CNRS – Lab-STICC. Fault Injection Attacks and Countermeasures in Embedded Processors, ARCHI’17** 18/88

- **Arnaud Tisserand, CNRS – Lab-STICC. Fault Injection Attacks and Countermeasures in Embedded Processors, ARCHI’17** 19/88

- **Arnaud Tisserand, CNRS – Lab-STICC. Fault Injection Attacks and Countermeasures in Embedded Processors, ARCHI’17** 20/88
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:

![Diagram of an attack](image)

- clic: good value
- clac: bad value

Side Channel Attacks (SCAs) (2/2)

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

```
\begin{align*}
E \rightarrow M & \quad \text{(plaintext)} \\
E_k(M) & \quad \text{(ciphertext)} \\
D_k(E_k(M)) & = M \\
\end{align*}
```

**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

```
\begin{align*}
\text{crypto.} & \quad \text{secret key} = 962571... \\
\end{align*}
```
“Read” the Traces

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


Differences & External Signature

An algorithm has a current signature and a time signature:

\[ r = c_0 \]

for \( i \) from 1 to \( n \) do
  if \( a_i = 0 \) then
    \( r = r + c_1 \)
  else
    \( r = r \times c_2 \)

Simple Power Analysis (SPA)

Source: [11]

SPA in Practice

General principle:

Methods: interpretation of the differences in

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

Arnaud Tisserand. CNRS – Lab-STICC. Fault Injection Attacks and Countermeasures in Embedded Processors, ARCHI’17 27/88

Arnaud Tisserand. CNRS – Lab-STICC. Fault Injection Attacks and Countermeasures in Embedded Processors, ARCHI’17 26/88
Limits of the SPA

Example of behavior difference: (activity into a register)

\[
\begin{array}{c|c}
  t & 0000000000000000 \\
  t+1 & 1111111111111111 \\
\end{array}
\]

**Important**: a small difference may be evaluated has 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)

**Example**

- **average**
- **correct**
- **incorrect**

**Template Attack**

- **correct**
- **incorrect**

**Correct hypothesis**
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

Side Channel Attack on ECC

**Scalar multiplication operation**

\[
\text{for } i \text{ from } 0 \text{ to } t - 1 \text{ do }
\]

\[
\begin{align*}
\text{if } k_i = 1 \text{ then } Q &= \text{ADD}(P, Q) \\
P &= \text{DBL}(P)
\end{align*}
\]

- 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

**Remark:**
\[\uparrow\] 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 else
```
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

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)

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. 0.7 V instead of 1.2 V)
• Glitches (dips, spikes) in the power supply at some selected moments

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).

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.

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

Glitchy Clock Generation Example


**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

<table>
<thead>
<tr>
<th>mode</th>
<th>glitch period</th>
<th>cycle</th>
<th>instruction</th>
<th>opcode (bin)</th>
</tr>
</thead>
<tbody>
<tr>
<td>normal</td>
<td>-</td>
<td>i</td>
<td>NOP</td>
<td>0000 0000 0000 0000</td>
</tr>
<tr>
<td>normal</td>
<td>-</td>
<td>i+1</td>
<td>EOR R15,R5</td>
<td>0010 0100 1111 0101</td>
</tr>
<tr>
<td>glitch</td>
<td>59 ns</td>
<td>i+1</td>
<td>NOP</td>
<td>0000 0000 0000 0000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>mode</th>
<th>glitch period</th>
<th>cycle</th>
<th>instruction</th>
<th>opcode (bin)</th>
</tr>
</thead>
<tbody>
<tr>
<td>normal</td>
<td>-</td>
<td>i</td>
<td>NOP</td>
<td>0000 0000 0000 0000</td>
</tr>
<tr>
<td>normal</td>
<td>-</td>
<td>i+1</td>
<td>SER R18</td>
<td>1110 1111 0010 1111</td>
</tr>
<tr>
<td>glitch</td>
<td>61 ns</td>
<td>i+1</td>
<td>LD R18,0xEF</td>
<td>1110 1110 0010 1111</td>
</tr>
<tr>
<td>glitch</td>
<td>60 ns</td>
<td>i+1</td>
<td>SBC R12,R15</td>
<td>0000 1000 0010 1111</td>
</tr>
<tr>
<td>glitch</td>
<td>59 ns</td>
<td>i+1</td>
<td>NOP</td>
<td>0000 0000 0000 0000</td>
</tr>
</tbody>
</table>

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 Hamming weights that are processed by the ATMega162.](image1)

![Fig. 5: The ATMega162 leaks the Hamming weight of all 256 possible intermediate values through the temperature.](image2)
Heating the MCU around 150–160°C → 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.

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

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

```
\begin{array}{c}
O_1 \rightarrow O_2 \rightarrow O_3 \rightarrow O_4 \rightarrow O_5 \rightarrow \text{end}
\end{array}
```

```
\begin{array}{c}
O_1 \rightarrow O_2 \rightarrow O_3 \rightarrow O_4 \rightarrow O_5 \rightarrow \text{end}
\end{array}
```
Safe Error Attack Example in Asymmetric Crypto

WEAK against SPA

\[
\text{for } i \text{ from } 0 \text{ to } n-1 \text{ do }
\]
\[
\text{if } s_i = 1 \text{ then }
\]
\[
v \leftarrow f(v, \ldots)
\]
\[
v \leftarrow g(v, \ldots)
\]
\[
\text{else }
\]
\[
w \leftarrow f(v, \ldots)
\]
\[
v \leftarrow g(v, \ldots)
\]

WEAK against SEA

\[
\text{for } i \text{ from } 0 \text{ to } n-1 \text{ do }
\]
\[
\text{if } s_i = 1 \text{ then }
\]
\[
v \leftarrow f(v, \ldots)
\]
\[
v \leftarrow g(v, \ldots)
\]
\[
\text{else }
\]
\[
w \leftarrow f(v, \ldots)
\]
\[
v \leftarrow g(v, \ldots)
\]

Useless or dummy operations are a bad idea

Fault Attack Example: Bit Flip on RSA Decryption

\[
\begin{align*}
M & \quad \cdot \quad \text{choose a plaintext message } M \\
C & = \mathcal{E}_k(M) \\
\mathcal{D} & (\mathcal{E}(C)) = M \\
D & \quad \cdot \quad \text{encrypt } M \text{ into } C = \mathcal{E}_k(M) \quad \cdot \quad \text{inject a fault by fliping } d_i \text{ for a random } i \left( d \text{ is the secret key} \right) \\
{\mathcal{E}}(C) & = C^{d_i} \\
\mathcal{D}(C) & = M^{d_i} \\
\mathcal{D}(\mathcal{E}(C)) & = M^{d_i} \\
\mathcal{D}(\mathcal{E}(C)) & = M \\
\text{test:} & \quad \cdot \quad \text{get small parts of } d, \text{ then mathematical attacks} \\
M & = \frac{1}{C^{d_i}} \mod N \implies d_i = 1 \\
M & = C^{d_i} \mod N \implies d_i = 0 \\
\text{retry for several } i & \quad \cdot \quad \text{get small parts of } d, \text{ then mathematical attacks}
\end{align*}
\]

Many other fault 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…)
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: $V_{DD}$ (logical 1) or GND (logical 0)

Low-level codings of a bit:

<table>
<thead>
<tr>
<th>$b$</th>
<th>Standard</th>
<th>Dual Rail</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>GND</td>
<td>$r_0 = V_{DD}$, $r_1 = \text{GND}$ (1, 0)$_{DR}$</td>
</tr>
<tr>
<td>1</td>
<td>$V_{DD}$</td>
<td>$r_0 = \text{GND}$, $r_1 = V_{DD}$ (0, 1)$_{DR}$</td>
</tr>
</tbody>
</table>

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)

Protected Multipliers

Unprotected

Protected

Overhead: Area/time $< 10\%$

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

### Circuit-Level Protections for Arithmetic Operators

- References: [6] and [7]

### Accelerator Architecture

- Data: \( w \)-bit (32, \ldots, 128) except for $k$ digits, **control**: a few bits per unit
Register File (≈ Dual Port Memory)

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

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: \( \forall i \ [R_i(k)]P = [k]P \)

Proposed solution: use random redundant representations of \( k \)

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)

Double-Base Number System

Standard radix-2 representation:
\[
k = \sum_{i=0}^{t-1} k_i 2^i = k_{t-1} 2^{t-1} + \cdots + k_2 2^2 + k_1 2 + k_0 \quad \text{implicit weights}
\]

Digits: \( k_i \in \{0, 1\} \), typical size: \( t \in \{160, \ldots, 600\} \)

Double-Base Number System (DBNS):
\[
k = \sum_{j=0}^{n-1} k_j 2^a 3^b = k_{n-1} 2^n 3^0 + \cdots + k_1 2^1 3^0 + k_0 2^0 3^0 \quad \text{explicit weights}
\]

\( a_j, b_j \in \mathbb{N}, \quad k_j \in \{1\} \text{ or } k_j \in \{-1, 1\}, \quad \text{size } n \approx \log t \)

DBNS is a very redundant and sparse representation: \( 1701 = (11010100101)_2 \)
\[
1701 = 243 + 1458 = 2^8 3^0 + 2^4 3^2 = (1, 0, 5), (1, 1, 6) \\
= 1728 - 27 = 2^9 3^1 - 2^0 3^3 = (1, 6, 3), (-1, 0, 3) \\
= 729 + 972 = 2^6 3^2 + 2^3 3^5 = (1, 0, 6), (1, 2, 5) \\
\ldots
\]
Resources: Conferences, Workshops, Journals, etc

- International Association for Cryptologic Research (IACR)
- 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 SoC-SiP
- French national working group on Security of Embedded Systems of the GDR SoC-SiP
- Conferences, workshops: CHES, FDTC, COSADE, CARDIS, CryptArchi...
- Journals: Journal of Cryptographic Engineering, IEEE Trans. on Computers, Circuits and Systems, VLSI Systems, ...
References II

An on-chip glitchy-clock generator for testing fault injection attacks.

The temperature side channel and heating fault attacks.

Hardware designer’s guide to fault attacks.

Differential power analysis.

In Proc. 10th International Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC), pages 77–88, Santa Barbara, CA, USA, August 2013. IEEE.

Fault injection using crowbars on embedded systems.

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.

Analysis of GF($2^{233}$) multipliers regarding elliptic curve cryptosystem applications.

References III

GF($2^n$) finite-field multipliers with reduced activity variations.
In 5th International Workshop on the Arithmetic of Finite Fields, volume 7389 of LNCS, pages 152–167, Bochum, Germany, July 2012. Springer.

Fast and secure finite field multipliers.
In Proc. 18th Euromicro Conference on Digital System Design (DSD), pages 653–660, Madeira, Portugal, August 2015.

Full hardware implementation of short addition chains recoding for ECC scalar multiplication.
In Actes Conference d’informatique en Parallélisme, Architecture et Système (ComPAS), Lille, France, June 2015.

A method for obtaining digital signatures and public-key cryptosystems.

A practical fault attack on square and multiply.
In Proc. 5th International Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC), pages 55–59, Washington, DC, USA, August 2008. IEEE.

Attaques en fautes globales et locales sur les cryptoprocesseurs AES : mise en œuvre et contremesures.

Practical setup time violation attacks on AES.
Good Books (in French)

Courbes elliptiques
Philippe Guillot
2010
Hermes

Micro et nano-électronique
Bases, Composants, Circuits
Hervé Fanet
2006
Dunod

Good Books (in English)

CMOS VLSI Design
A Circuits and Systems Perspective
Neil Weste and David Harris
3rd edition, 2004
Addison Wesley

Power Analysis Attacks
Revealing the Secrets of Smart Cards
Stefan Mangard, Elisabeth Oswald and Thomas Popp
2007
Springer

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