ELECTROMAGNETIC FAULT INJECTION: TOWARDS A FAULT MODEL ON A 32-BIT MICROCONTROLLER

Nicolas Moro\textsuperscript{1,3}, Amine Dehbaoui\textsuperscript{2}, Karine Heydemann\textsuperscript{3}, Bruno Robisson\textsuperscript{1}, Emmanuelle Encrenaz\textsuperscript{3}

\begin{enumerate}
\item \textbf{CEA}
Commissariat à l’Energie Atomique et aux Energies Alternatives
\item \textbf{ENSM.SE}
Ecole Nationale Supérieure des Mines de Saint-Etienne
\item \textbf{LIP6 - UPMC}
Laboratoire d’Informatique de Paris 6
Université Pierre et Marie Curie
\end{enumerate}

FDTC 2013 – AUGUST 20, SANTA BARBARA, USA
MOTIVATIONS

- **Security** of microcontroller-based embedded systems against fault injection attacks

- **Target**: ARM Cortex-M3 microcontroller

- **Fault injection means**: Pulsed electromagnetic fault injection

- Theoretical attacks rely on an **attacker’s fault model**

- Electromagnetic fault injection is quite recent

- Very few **in-depths studies** of the effects on complex systems

 ➔ Better understanding of the effects of **EM fault injection**

 ➔ Detailed fault model at a **register-transfer level**
I. Experimental setup

II. General approach

III. Study of the injection parameters

IV. Register-transfer level fault model

V. Conclusion
Several **physical ways** to inject faults into a circuit’s computation

- Necessary for an attacker to **know the type of injected faults**

<table>
<thead>
<tr>
<th>Fault target</th>
<th>Data, instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fault type</td>
<td>Bit flip, reset at 0, set at 1, stuck</td>
</tr>
<tr>
<td>Granularity</td>
<td>Bit, byte, word</td>
</tr>
<tr>
<td>Determinism</td>
<td>Deterministic, metastable, random</td>
</tr>
<tr>
<td>Temporal aspect</td>
<td>Single piece of data/instruction, multiple</td>
</tr>
</tbody>
</table>
Pulsed electromagnetic fault injection

- **Transient and local** effect of the fault injection
- **Standard circuits not protected** against this technique
- **Solenoid** used as an injection antenna
- Up to **200V** sent on the injection antenna, pulses width longer than **10ns**

**Microcontroller based on an ARM Cortex-M3**

- Frequency 56 MHz
- 16/32 bits **Thumb2** RISC instruction set
- ARMv7-M modified Harvard architecture
- SWD link to **debug the microcontroller**
Experiment **driven by the computer**
Execution of a **computation on the target device**
Sending of a **voltage pulse**
**Stop** of the microcontroller
**Harvesting** of the microcontroller’s internal data
Analysis of the obtained results

**Main experimental parameters**

- **Position** of the injection antenna
- **Electric parameters** of the pulse
- **Injection time** of the pulse
- **Executed code** on the microcontroller
OUTLINE

I. Experimental setup

II. General approach

III. Study of the injection parameters

IV. Register-transfer level fault model

V. Conclusion
GENERAL APPROACH

Exhaustive instruction simulation
(finds instructions which could enable to reach B’ from A)

Fault injection

Experimental fault
(depends on the experimental parameters)

Initial state A

Instruction

Expected state B’

Output pieces of data | Detail
--- | ---
R0 to R12 | General-purpose registers
R13 (SP) | Stack pointer
R14 (LR) | Link register
R15 (PC) | Program counter
XPSR | Program Status Register
- Flags
- Details about the triggered interruptions
- Details about the execution mode
Result | Memory address that contains the calculation’s output

II – General approach
Instruction skip simulation

<table>
<thead>
<tr>
<th>IT</th>
<th>Ligne_assembleur</th>
<th>Detail_IT</th>
<th>Adresse</th>
<th>R0</th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
<th>R4</th>
<th>R5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thread_mode</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>0x20000040c</td>
<td>0x2</td>
<td>0x20000421</td>
<td>0x3</td>
<td>0x2</td>
<td>0x400</td>
</tr>
<tr>
<td>Thread_mode</td>
<td>0x080002245603 LDR</td>
<td>r3,[r0,#0x00]</td>
<td>None</td>
<td>0x20000040c</td>
<td>0x2</td>
<td>0x20000421</td>
<td>0x7[FAUTE]</td>
<td>0x2</td>
<td>0x400</td>
</tr>
<tr>
<td>Thread_mode</td>
<td>0x080002264423 ADD</td>
<td>r3,r3,r4</td>
<td>None</td>
<td>0x20000040c</td>
<td>0x2</td>
<td>0x20000421</td>
<td>0x2[FAUTE]</td>
<td>0x2</td>
<td>0x400</td>
</tr>
<tr>
<td>Thread_mode</td>
<td>0x080002286003 STR</td>
<td>r3,[r0,#0x00]</td>
<td>None</td>
<td>0x20000040c</td>
<td>0x2</td>
<td>0x20000421</td>
<td>0x2[FAUTE]</td>
<td>0x2</td>
<td>0x400</td>
</tr>
<tr>
<td>Thread_mode</td>
<td>0x0800022A1F010101 ADD</td>
<td>r1,r1,#0x01</td>
<td>None</td>
<td>0x20000040c</td>
<td>0x2</td>
<td>0x20000421</td>
<td>0x4[FAUTE]</td>
<td>0x2</td>
<td>0x400</td>
</tr>
<tr>
<td>Thread_mode</td>
<td>0x0800022E2902 CMP</td>
<td>r1,#0x02</td>
<td>None</td>
<td>0x20000040c</td>
<td>0x1[FAUTE]</td>
<td>0x20000421</td>
<td>0x1[FAUTE]</td>
<td>0x400</td>
<td></td>
</tr>
<tr>
<td>Thread_mode</td>
<td>0x08000230DBF6 BLT</td>
<td>None</td>
<td>None</td>
<td>0x20000040c</td>
<td>0x2</td>
<td>0x20000421</td>
<td>0x2[FAUTE]</td>
<td>0x1[FAUTE]</td>
<td>0x400</td>
</tr>
</tbody>
</table>

Experimental measurements

<table>
<thead>
<tr>
<th>Instant</th>
<th>Exec</th>
<th>IT</th>
<th>Ligne_assembleur</th>
<th>Detail_IT</th>
<th>Adresse</th>
<th>R0</th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
<th>R4</th>
<th>R5</th>
<th>R6</th>
<th>R7</th>
</tr>
</thead>
<tbody>
<tr>
<td>37600</td>
<td>1</td>
<td>Thread_mode</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>0x20000040c</td>
<td>0x2</td>
<td>0x20000421</td>
<td>0x3</td>
<td>0x2</td>
<td>0x4010c10</td>
<td>0x4010c14</td>
<td>0x100</td>
</tr>
<tr>
<td>37600</td>
<td>2</td>
<td>Thread_mode</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>0x20000040c</td>
<td>0x2</td>
<td>0x20000421</td>
<td>0x3</td>
<td>0x2</td>
<td>0x4010c10</td>
<td>0x4010c14</td>
<td>0x100</td>
</tr>
<tr>
<td>37600</td>
<td>3</td>
<td>Thread_mode</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>0x20000040c</td>
<td>0x2</td>
<td>0x20000421</td>
<td>0x3</td>
<td>0x2</td>
<td>0x4010c10</td>
<td>0x4010c14</td>
<td>0x100</td>
</tr>
<tr>
<td>37600</td>
<td>4</td>
<td>Thread_mode</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>0x20000040c</td>
<td>0x2</td>
<td>0x20000421</td>
<td>0x3</td>
<td>0x2</td>
<td>0x4010c10</td>
<td>0x4010c14</td>
<td>0x100</td>
</tr>
<tr>
<td>37600</td>
<td>5</td>
<td>Thread_mode</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>0x20000040c</td>
<td>0x2</td>
<td>0x20000421</td>
<td>0x2[FAUTE]</td>
<td>0x1[FAUTE]</td>
<td>0x4010c10</td>
<td>0x4010c14</td>
<td>0x100</td>
</tr>
</tbody>
</table>

- Two lines are equal \( \Leftrightarrow \) R0 to R12 + XPSR + result + SP + PC are equal
Example of simulation of a 16-bit instruction replacement

- Very long for an exhaustive simulation over the whole instruction set
- Two lines are equal $\iff$ R0 to R12 + XPSR + result are equal
I. Experimental setup

II. General approach

III. Study of the injection parameters

IV. Register-transfer level fault model

V. Conclusion
INFLUENCE OF THE ANTENNA’S POSITION

- **Green**: hardware interrupts have been triggered
- **Red**: faults on the output value have been obtained

**Frequency 56 MHz – Pulse width 10 ns – Pulse voltage 190V – Period 17ns**

- **Target instruction**: single LOAD instruction that loads 0x12345678 into R8
- **20 ns** time interval, by steps of **200 ps** - **3 mm** square, by steps of **200 µm**
- **Variable increase** of the Hamming weight of the loaded piece of data
- **No fault** on other registers than **R8** (except for very few faults on R0)
Example of temporal cartography on an addition loop

Observations:
- One power of two has not been added
- BusFault or UsageFault interrupts

→ Does our fault injection have an effect on the data flow or the control flow?

Test program:
loop to sum the elements of an array that contains eight powers of two 3.5 µs, by steps of 200 ps

Expected result: 0xFF

```
addition_loop:  
ldr r4, [r2, r1, 1s1 #2] ; r4 = array[i]  
ldr r3, [r0, #0] ; r3 = result  
add r3, r4 ; r3 = r3 + r4  
str r3, [r0, #0] ; result = r3  
add r1, r1, #1 ; r1 = r1 + 1  
cmp r1, #8 ; r1 == 8 ?
blt addition_loop
```
**INFLUENCE OF THE PULSE’S VOLTAGE**

**LDR R4, PC#44** with 0x12345678 at the address PC#44

<table>
<thead>
<tr>
<th>Pulse voltage</th>
<th>Output value</th>
<th>Occurrence rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>172V</td>
<td>1234 5678</td>
<td>100 %</td>
</tr>
<tr>
<td>174V</td>
<td>9234 5678</td>
<td>73 %</td>
</tr>
<tr>
<td>176V</td>
<td>FE34 5678</td>
<td>30 %</td>
</tr>
<tr>
<td>178V</td>
<td>FFF4 5678</td>
<td>53 %</td>
</tr>
<tr>
<td>180V</td>
<td>FFFD 5678</td>
<td>50 %</td>
</tr>
<tr>
<td>182V</td>
<td>FFFF 7F78</td>
<td>46 %</td>
</tr>
<tr>
<td>184V</td>
<td>FFFF FFFB</td>
<td>40 %</td>
</tr>
<tr>
<td>186V</td>
<td>FFFF FFFF</td>
<td>100 %</td>
</tr>
</tbody>
</table>

- Simulation: corresponds to **no instruction replacement**
- Looks like a **set at 1** fault model on the **Flash memory data transfers**
- Possible **precharge of the data bus** on this architecture
I. Experimental setup

II. General approach

III. Study of the injection parameters

IV. Register-transfer level fault model

V. Conclusion
Experiments with a sequence of **NOP** (BF 00)

Four kinds of faults
- Fault on **R7**
- The program does not stop
- **UsageFault** exceptions (Invalid Instruction / No Coprocessor)
- Fault on **R0**

- Sometimes a modification of the **number of executed cycles**
- Simulation on the ISA: some instructions can explain the results
- Some faults only equivalent to a **STR R0, [R0, #0] instruction**

<table>
<thead>
<tr>
<th>NOP - BF00</th>
<th>1011 1111 0000 0000</th>
</tr>
</thead>
<tbody>
<tr>
<td>STR R0, [R0, #0] - 6000</td>
<td>0110 0000 0000 0000</td>
</tr>
<tr>
<td>NOP - BF00</td>
<td>1011 1111 0000 0000</td>
</tr>
<tr>
<td>NOP - BF00</td>
<td>1011 1111 0000 0000</td>
</tr>
</tbody>
</table>
Normal behaviour

Clock 56MHz

HADDRI

HRDATAI

0x00 NOP - BF00

0x02 NOP - BF00

0x04 NOP - BF00

0x06 NOP - BF00

0x08 NOP - BF00

0x0A NOP - BF00

0x0C NOP - BF00

0x0E NOP - BF00

0x10 NOP - BF00

EXECUTE

DECODE

EXECUTE

DECODE

EXECUTE

DECODE

DECODE

DECODE

DECODE

DECODE

DECODE

DECODE

DECODE

DECODE
With an electromagnetic fault injection

**EM Glitch**

Clock 56MHz

HADDRI

HRDATAI

<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>NOP</td>
</tr>
<tr>
<td>0x02</td>
<td>NOP</td>
</tr>
<tr>
<td>0x04</td>
<td>NOP</td>
</tr>
<tr>
<td>0x06</td>
<td>NOP</td>
</tr>
<tr>
<td>0x08</td>
<td>NOP</td>
</tr>
<tr>
<td>0x0A</td>
<td>NOP</td>
</tr>
<tr>
<td>0x0C</td>
<td>NOP</td>
</tr>
<tr>
<td>0x0E</td>
<td>NOP</td>
</tr>
<tr>
<td>0x10</td>
<td>NOP</td>
</tr>
</tbody>
</table>

**FDTC 2013 – Santa Barbara, USA**
Normal behaviour

Clock 56MHz

<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
<th>Memory Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>NOP</td>
<td>BF00</td>
</tr>
<tr>
<td>0x02</td>
<td>LDR R4, [PC #44]</td>
<td>4C04</td>
</tr>
<tr>
<td>0x04</td>
<td>NOP</td>
<td>BF00</td>
</tr>
<tr>
<td>0x06</td>
<td>NOP</td>
<td>BF00</td>
</tr>
<tr>
<td>0x08</td>
<td>NOP</td>
<td>BF00</td>
</tr>
<tr>
<td>0x0A</td>
<td>NOP</td>
<td>BF00</td>
</tr>
<tr>
<td>0x0C</td>
<td>NOP</td>
<td>BF00</td>
</tr>
<tr>
<td>0x0E</td>
<td>NOP</td>
<td>BF00</td>
</tr>
<tr>
<td>0x10</td>
<td>NOP</td>
<td>BF00</td>
</tr>
</tbody>
</table>

PC+44
With an electromagnetic fault injection

EM Glitch

Clock 56MHz

HADDR

HRDATA

0x00 NOP - BF00

0x02 LDR R4, [PC #44] - 4C04

0x04 NOP - BF00

0x06 NOP - BF00

0x08 NOP - BF00

0x0A NOP - BF00

0x0C NOP - BF00

0x0E NOP - BF00

0x10 NOP - BF00

PC+44

EXECUTE

DECODE

EXECUTE

DECODE

FETCH

FETCH

FETCH

FETCH

FETCH

FETCH
Possible to fault the transfers from the Flash memory

Consequences regarding the instruction flow
- Instructions replacements
- Instruction skips under certain conditions (~ 20-30% of time)
- Some instructions may be more sensitive than others
- Some registers seem to be more sensitive than others

Consequences regarding the data flow
- Corruption of the LOAD instructions from the Flash memory (encryption keys,...)
- Some metastability phenomena, but deterministic under some conditions
- Faulty values with higher Hamming weight (on this architecture)
I. Experimental setup

II. General approach

III. Study of the injection parameters

IV. Register-transfer level fault model

V. Conclusion
CONCLUSION AND PERSPECTIVES

- A first attempt of **fault model** for EM fault injection on a 32-bit μC
- Corruption of the **transfers from the Flash memory** on the buses
- The obtained effects seem **very similar** to the ones obtained with **clock glitches or other fault injection means**
- **Similar effects** obtained previously on a very different architecture (Atmel AVR ATmega128 8-bit microcontroller)
- Possibility to perform **instruction skips** under some specific conditions

**Perspectives**
- Use more advanced debug techniques to understand better instruction replacements
- Define a higher-level fault model that can be used for theoretical attacks
Any questions?