Experimental Analysis of the EM Instruction Skip Fault Model

A. Menu\textsuperscript{1, 2}, J.-M. Dutertre\textsuperscript{1}, O. Potin\textsuperscript{1}, J.-B. Rigaud\textsuperscript{1}, J.-L. Danger\textsuperscript{2}

Thursday, April 2\textsuperscript{nd}, 2020

This research has been partially supported by the European Commission under H2020 SPARTA (Grant Agreement 830892)
Hardware attacks

- i.e. physical attacks targeting integrated circuits

Fault injection attacks

- Active perturbation attacks
- **Goal**: induce an information leakage or gain unauthorized access
- **Injection technique**: EM perturbation
- **Target**: microcontrollers
- **Fault model**:
  - what an attacker can do,
  - instruction skip.
I. EM-induced instruction skip fault model

II. Experimental setup

III. Experimental results

IV. Discussion

V. Conclusion
I. EM-induced instruction skip fault model

Instruction skip fault model

Analysis of the EM-induced instruction skip fault model:

- Program Counter increase (PC → PC + 1)?

```
ld r16, 0x39
ld r17, 0x38
ld r18, 0x37
ld r19, 0x36
...
ld r25, 0x30
```

EM

```
ld r16, 0x39
ld r18, 0x37
ld r19, 0x36
...
ld r25, 0x30
```

PC → PC+1
I. EM-induced instruction skip fault model

Instruction skip fault model

Analysis of the EM-induced instruction skip fault model:

• Instruction modification (no operation, nop)?

```
ld r16, 0x39
ld r17, 0x38
ld r18, 0x37
ld r19, 0x36
...  ld r25, 0x30
```

```
EM     EM
ld r16, 0x39  nop
nop
ld r18, 0x37
ld r19, 0x36
...  ld r25, 0x30
```
I. EM-induced instruction skip fault model

- State of the art
  - EM-induced instruction skips
    - Single instruction skip:
      - 8-bit uCTRL [BEC19] (same target)
      - 32-bit uCTRL [MOR14]
    - Several instruction skips:
      - replay of 4 instr. at readout from cache [RIV15]
      - pipeline’s instruction [YUC16]
    ⇒ Side effect of the target’s microarchitecture

[BEC19] Beckers et al., Characterization of EM faults on ATmega328P, IEEE SEC 2019
[MOR14] Moro et al., Experimental evaluation of two software countermeasures against fault attacks, HOST 2014
[RIV15] Rivière et al., High precision fault attacks on the instruction cache of ARMv7-M architectures, HOST 2015
[YUC16] Yuce et al., Software fault resistance is futile: Effective single-glitch attacks, FDTC 2016
I. EM-induced instruction skip fault model

☑ State of the art

▪ Laser-induced instruction skips
  
  • Single instruction skip:
    - 8-bit uCTRL [BRE15, KUM18] (same target)
    - 32-bit uCTRL, CRT-RSA [TRI10], 2 skips distant from 58 ms

  ⇔ State of the art until 2019 = single instruction skip

  • Multiple instruction skips:
    - 8-bit uCTRL [DUT19] (same target):

  ⇔ Arbitrary number of consecutive (or not) instruction skips

[BRE15] Breier et al., Testing feasibility of back-side laser fault injection on a microcontroller, WESS’15
[BRE15] Breier et al., Laser profiling for the back-side fault attacks: With a practical laser skip instruction attack on AES, CPSS’15
[KUM18] Kumar et al. An in-depth and black-box characterization of the effects of laser pulses on ATmega328P, CARDIS 2018
[TRI10] Trichina et al., Multi fault laser attacks on protected CRT-RSA, FDTC 2010
[DUT19] Dutertre et al., Experimental Analysis of the Laser-induced Instruction Skip Fault Model, NordSec 2019
I. EM-induced instruction skip fault model

Research objective

- **EM-induced single nop**
  - accuracy, success rate, etc.

- **Ability to extend the fault model** (w.r.t. countermeasure design)?

```
ld r16, 0x39
ld r17, 0x38
ld r18, 0x37
ld r19, 0x36
...
ld r25, 0x30
```

**EM**

```
ld r16, 0x39
nop
nop
nop
...
ld r25, 0x30
```

Single nop

Several consecutive nops
I. EM-induced instruction skip fault model

- Research objective
  - EM-induced single nop
    - accuracy, success rate, etc.
  - Ability to extend the fault model (w.r.t. countermeasure design)?

```
1d r16, 0x39          EM          1d r16, 0x39
1d r17, 0x38          nop        1d r18, 0x37
1d r18, 0x37          nop        1d r19, 0x36
1d r19, 0x36
```

Single nop
Several consecutive nops
Several non-consecutive nops
I. EM-induced instruction skip fault model
II. Experimental setup
III. Experimental results
IV. Discussion
V. Conclusion
EM-induced instruction skips

- The EM injection bench

Target close up
II. Experimental setup

- The EM injection bench

![Diagram of EM injection bench with pulse generator, voltage pulse, EM injection probe, and target.]

- EM injection probe mounted on XYZ displacement stages (µm accuracy)

> ruby -I ./lib
> python …

USB serial
II. Experimental setup

The target: as simple as possible

- 8bit microcontroller: ATmega328P, 16 MHz (no pipeline)

RAM 2kB
Flash 3kB
EEPROM 1kB
32 gp registers
Fetch/execute
The test code

- **Goal: instruction skip analysis**

### Initialization

<table>
<thead>
<tr>
<th>Register</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
<th>23</th>
<th>24</th>
<th>25</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expected</td>
<td>0x39</td>
<td>0x38</td>
<td>0x37</td>
<td>0x36</td>
<td>0x35</td>
<td>0x34</td>
<td>0x33</td>
<td>0x32</td>
<td>0x31</td>
<td>0x30</td>
</tr>
</tbody>
</table>

### Test code part

- `ld r16, Z+`
- `ld r17, Z+`
- `ld r18, Z+`
- `ld r19, Z+`
- `ld r20, Z+`
- `ld r25, Z+`
The test code

- Goal: instruction skip analysis, instruction skip's effect

**Initialization**

```
  ldi r16, 0x55  Z  →  0x39
  ldi r17, 0x55   0x38
  ldi r18, 0x55   0x37
  ...
  ldi r25, 0x55   0x30
```

**Test code part**

```
  ld r16, Z+
  ld r17, Z+
  ld r18, Z+
  ...
  skip
  ld r20, Z+
  ...
  ld r25, Z+
```

<table>
<thead>
<tr>
<th>Register</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
<th>23</th>
<th>24</th>
<th>25</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expected</td>
<td>0x39</td>
<td>0x38</td>
<td>0x37</td>
<td>0x36</td>
<td>0x35</td>
<td>0x34</td>
<td>0x33</td>
<td>0x32</td>
<td>0x31</td>
<td>0x30</td>
</tr>
<tr>
<td>Read</td>
<td>0x39</td>
<td>0x38</td>
<td>0x37</td>
<td>0x55</td>
<td>0x36</td>
<td>0x35</td>
<td>0x34</td>
<td>0x33</td>
<td>0x32</td>
<td>0x31</td>
</tr>
</tbody>
</table>

(increment of address Z was skipped)
II. Experimental setup

- **The test code**
  - **Synchronization**: 2 trigger signals (red and blue)

![Diagram showing synchronization and test code]

**Initialization**
- `ldi r16, 0x55`  
- `ldi r17, 0x55`  
- `ldi r18, 0x55`  
- `ldi r25, 0x55`

**Test code part**
- `ld r16, Z`  
- `ld r17, Z`  
- `ld r18, Z`  
- `ld r19, Z`  
- `ld r20, Z`  
- `ld r25, Z`

**Registers readback**
I. EM-induced instruction skip fault model
II. Experimental setup
III. Experimental results
IV. Discussion
V. Conclusion
III. Experimental results

- EM-induced instruction skip
  - EM injection probe location

EM sensitive area (used for reported exp. results)

Laser sensitive area
III. Experimental results

- EM-induced instruction skip
EM perturbation

V_{pulse}: -200 \text{ V} / 100 \text{ ns}

With EM perturbation

Normal operation

EM-induced instruction skip

III. Experimental results
III. Experimental results

- **EM-induced instruction skip** *(Vpulse: - 200 V / 100 ns)*

1. Registers readback:

<table>
<thead>
<tr>
<th>Register</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
<th>23</th>
<th>24</th>
<th>25</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expected</td>
<td>0x39</td>
<td>0x38</td>
<td>0x37</td>
<td>0x36</td>
<td>0x35</td>
<td>0x34</td>
<td>0x33</td>
<td>0x32</td>
<td>0x31</td>
<td>0x30</td>
</tr>
<tr>
<td>Read</td>
<td>0x39</td>
<td>0x38</td>
<td>0x37</td>
<td>0x55</td>
<td>0x36</td>
<td>0x35</td>
<td>0x34</td>
<td>0x33</td>
<td>0x32</td>
<td>0x31</td>
</tr>
</tbody>
</table>

2. Decrement of trigger duration: $1 \times T_{\text{clk}}$

```
... ld r18, Z+  EM  ld r18, Z+ ...
... ld r19, Z+  nop  ld r20, Z+ ...
... ld r20, Z+  ... ...
```

Instructions: \( \text{ld} \rightarrow 2 \times T_{\text{clk}} \) while \( \text{nop} \rightarrow 1 \times T_{\text{clk}} \)

**Instruction skip: due to a nop** *(not to an increment of the PC)*
III. Experimental results

- EM-induced instruction skip (Vpulse: -200 V / 100 ns)
- Timing accuracy, single skip
III. Experimental results

- EM-induced instruction skip (Vpulse: -250 V / 100 ns)
- Double skip, obtained with an increased stress
III. Experimental results

- EM-induced instruction skips (Vpulse: - 400 V)
- Effect of modifying the voltage pulse width

→ Up to 6 consecutive instruction skips were obtained (other settings)
III. Experimental results

- **EM-induced instruction skips** ($V_{\text{pulse}}: +150 \text{ V}, 10 \text{ ns}, \Delta = 570 \text{ ns}$)
- **Skipping none-consecutive instructions**
  - Double pulse voltage generator: min. $\Delta = 150 \text{ ns}$ (! usual repetition rate : 20 kHz)

![Graph showing voltage changes over time with markers r19 and r22]
III. Experimental results - Application to a verify PIN function

- Target’s microcode: verify PIN function (x4: 0–9)

```c
#define BOOL_TRUE 0xAA  \10101010
#define BOOL_FALSE 0x55 \01010101

void VerifyPIN()
{
    g_authenticated = BOOL_FALSE;

    if(g_ptc > 0) {
        if(byteArrayCompare(g_userPin, g_cardPin) == BOOL_TRUE) {
            g_ptc = 3;
            g_authenticated = BOOL_TRUE; // Authentication();
        }
    } else {
        g_ptc--;
    }
}
```

From the ANR SERTIF project – http://sertif-projet.forge.imag.fr
III. Experimental results - Application to a verify PIN function

- Target’s microcode: verify PIN function (x4: 0–9)

```c
#define BOOL_TRUE 0xAA  \10101010
#define BOOL_FALSE 0x55  \01010101

void VerifyPIN()
{
    g_authenticated = BOOL_FALSE;

    if(g_ptc > 0) {
        if(byteArrayCompare(g_userPin, g_cardPin) == BOOL_TRUE) {
            g_ptc = 3;
            g_authenticated = BOOL_TRUE;  // Authentication()
        }
    } else {
        g_ptc--;
    }
}
```

- `g_authenticated` global var.
  - FALSE → user
  - TRUE → admin.
III. Experimental results - Application to a verify PIN function

- Target’s microcode: verify PIN function (x4: 0–9)

```c
#define BOOL_TRUE 0xAA  "10101010"
#define BOOL_FALSE 0x55  "01010101"

void VerifyPIN()
{
    g_authenticated = BOOL_FALSE;
    if(g_ptc > 0) {
        if(byteArrayCompare(g_userPin, g_cardPin) == BOOL_TRUE) {
            g_ptc = 3;
            g_authenticated = BOOL_TRUE; // Authentication();
        }
        else {
            g_ptc--;
        }
    }
}
```

- g_ptc
- pin try counter
- max. = 3
Target’s microcode: verify PIN function (x4: 0–9)

```c
#define BOOL_TRUE 0xAA \10101010
#define BOOL_FALSE 0x55 \01010101

void VerifyPIN()
{
    g_authenticated = BOOL_FALSE;

    if(g_ptc > 0) {
        if(byteArrayCompare(g_userPin, g_cardPin) == BOOL_TRUE) {
            g_ptc = 3;
            g_authenticated = BOOL_TRUE; // Authentication();
        } else {
            g_ptc--;
        }
    }
}
```
III. Experimental results - Application to a verify PIN function

- Target's microcode: verify PIN function (x4: 0–9)

```c
#define BOOL_TRUE 0xAA  \10101010
#define BOOL_FALSE 0x55  \01010101

void VerifyPIN()
{
  g_authenticated = BOOL_FALSE;
  
  if(g_ptc > 0) {
    if(byteArrayCompare(g_userPin, g_cardPin) == BOOL_TRUE) {
      g_ptc = 3;
      g_authenticated = BOOL_TRUE; // Authentication();
    }
    else {
      g_ptc--;
    }
  }
}
```
III. Experimental results - Application to a verify PIN function

- Attack path: test of the remaining number of tries
  - Force strings comparison → PIN exhaustive search (brute force attack)
  - EM-induced instruction skip

```c
void VerifyPIN()
{
    g_authenticated = BOOL_FALSE;
    if(g_ptc > 0) {
        if(byteArrayCompare(g_userPin, g_cardPin) == BOOL_TRUE) {
            g_ptc = 3;
            g_authenticated = BOOL_TRUE; // Authentication()
        } else {
            g_ptc--;
        }
    }
}
```
III. Experimental results - Application to a verify PIN function

- Trigger waveforms: $g_{ptc} > 0$

![Graph showing waveforms with labels VerifyPIN() function and byteArrayCompare() function.]
III. Experimental results - Application to a verify PIN function

- Trigger waveforms: $g_{ptc} = 0$
III. Experimental results - Application to a verify PIN function

- $g_{ptc} = 0 +$ EM attack $\Rightarrow$ exhaustive search (success rate 100%)
I. EM-induced instruction skip fault model
II. Experimental setup
III. Experimental results
IV. Discussion
V. Conclusion
EM-induced instruction skip fault model

A powerful fault model

Experimental assessment on an 8-bit microcontroller

- ability to skip a (chosen) single instruction with 100% success rate
- ability to skip several successive instructions (6 successive skips, on exp. Basis)
- ability to skip different instructions in different sections of a microcode

Requirement: synchronization with the target

Denomination of the fault model

Instructions are not skipped, rather turned into a \texttt{nop}

Also instruction modification (turning an instruction into another one irrelevant to the actual part of the program).
IV. Discussion

- **Comparison with the laser-induced instr. skip fault model**
  
  Laser Fault Model: erase arbitrary parts of a uCTRL’s microcode
  
  EM Fault Model:
  
  - less potent & versatile,
  - ! EM injection more common/affordable

- **Synchronization**
  
  Easy when using trigger signal (white box approach)
  
  Black box?
  
  - Use of a smart trigger (real time recognition of a side channel signal, [WOU11])

[WOU11] van Woudenberg et al., Practical optical fault injection on secure microcontrollers, FDTC 2011
I. EM-induced instruction skip fault model
II. Experimental setup
III. Experimental results
IV. Discussion
V. Conclusion
V. Conclusion

- Experimental assessment of an extended EM-induced instruction skip fault model

  Ability to erase a chosen instruction (or small group of instr.) in a microcontroller program at runtime

- To be considered when designing countermeasures

  Software CMs may be also skipped if not designed carefully
Thank you for your attention
dutertre@emse.fr

This research has been partially supported by the European Commission under H2020 SPARTA (Grant Agreement 830892)