Software Fault Resistance is Futile: Effective Single-Glitch Attacks

B. Yuce, N. F. Ghalaty, H. Santapuri, C. Deshpande, C. Patrick, P. Schaumont

Virginia Tech
Bradley Department of ECE
farhady@vt.edu

This research was supported through NSF Grant 1441710, Grant 1115839, and through SRC.
• Fault attacks pose a serious threat to embedded systems:
  • To gain the control of a software program
  • To observe the secret information
• Hardware determines the fault behavior of software.
Software Countermeasures

Faults attacks on embedded software:

① Fault Injection

1. Inst1
2. Inst2 Inst1
3. Inst3 Inst2 Inst1

② Faulty Instructions

0x00: Inst1
0x04: Inst2
0x08: Inst3

③ Faulty Output

Output = K ^ SBOX(S)

Software countermeasures:

algorithm level

instruction level

micro-architectural level
## Fault Behavior in a RISC Pipeline (1)

### 7-Stage RISC Pipeline:

<table>
<thead>
<tr>
<th>Stage</th>
<th>Fetch (F)</th>
<th>Decode (D)</th>
<th>Register Access (A)</th>
<th>Execute (E)</th>
<th>Memory (M)</th>
<th>Exception (X)</th>
<th>Write-Back (W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>F4</td>
<td>D3</td>
<td>A2</td>
<td>E1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Stall</td>
<td>Stall</td>
<td>Stall</td>
<td>E2</td>
<td>M1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>F5</td>
<td>D4</td>
<td>A3</td>
<td>Stall</td>
<td>M2</td>
<td>X1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>F6</td>
<td>D5</td>
<td>A4</td>
<td>Stall</td>
<td>X2</td>
<td>W1</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>F7</td>
<td>D6</td>
<td>A5</td>
<td>E4</td>
<td>M3</td>
<td>Stall</td>
<td>W2</td>
</tr>
<tr>
<td>6</td>
<td>F8</td>
<td>D7</td>
<td>A6</td>
<td>E5</td>
<td>M4</td>
<td>X3 Stall</td>
<td></td>
</tr>
</tbody>
</table>
• If E3 has the **highest** critical path (i.e., fault sensitivity):
How to Tune Glitch Parameters?

- Microprocessor Fault sensitivity Model of each (instruction, pipeline stage)

<table>
<thead>
<tr>
<th>Pipeline Stages</th>
<th>Critical Path Delays (Fault Sensitivity)</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>3.53ns</td>
</tr>
<tr>
<td>D</td>
<td>3.26ns</td>
</tr>
<tr>
<td>A</td>
<td>5.62ns</td>
</tr>
<tr>
<td>E</td>
<td>5.20ns</td>
</tr>
<tr>
<td>M</td>
<td>7.58ns</td>
</tr>
<tr>
<td>X</td>
<td>3.17ns</td>
</tr>
<tr>
<td>W</td>
<td>4.45ns</td>
</tr>
</tbody>
</table>
Fault Attack on Software Countermeasures

Algorithm Level Analysis → Instruction Level Analysis → Micro-architectural Level Analysis
Objective:
Keep two copies of an Instruction, Raise an alarm in case of a mismatch

```
ld [%fp - 12], %g2
ld [%fp - 12], %g3
cmp %g2, %g3
bne .alarm
```
ID Behavior in Pipeline

id [%fp - 12], %g2
id [%fp - 12], %g3
cmp %g2, %g3
bne .alarm

F | D | A | E | M | X | W

Clock cycles 0 LD1
ld [%fp - 12], %g2
ld [%fp - 12], %g3
cmp %g2, %g3
bne .alarm
ID Behavior in Pipeline

\[
\begin{align*}
\text{ld} & \quad [\%fp - 12], \%g2 \\
\text{ld} & \quad [\%fp - 12], \%g3 \\
\text{cmp} & \quad \%g2, \%g3 \\
bne & \quad .\text{alarm}
\end{align*}
\]

Clock cycles:
- 0: LD1
- 1: LD2, LD1
- 2: CMP, LD2, LD1

Pipeline stages:
- F
- D
- A
- E
- M
- X
- W
ld [%fp - 12], %g2
ld [%fp - 12], %g3
cmp %g2, %g3
bne .alarm

Data Dependency

<table>
<thead>
<tr>
<th>clock cycles</th>
<th>LD1</th>
<th>LD2</th>
<th>LD1</th>
<th>CMP</th>
<th>LD2</th>
<th>LD1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

pipeline stages
### ID Behavior in Pipeline

<table>
<thead>
<tr>
<th>Clock Cycles</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>LD1</td>
</tr>
<tr>
<td>1</td>
<td>LD2, LD1</td>
</tr>
<tr>
<td>2</td>
<td>CMP, LD2, LD1</td>
</tr>
<tr>
<td>3</td>
<td>BNE, CMP, LD2, LD1</td>
</tr>
<tr>
<td>4</td>
<td>BNE, CMP, LD2, LD1</td>
</tr>
</tbody>
</table>

Data Dependency:

- LD1
- LD2
- LD1
- CMP
- CMP

Pipeline Stages:

F | D | A | E | M | X | W

ld  [%fp - 12], %g2
ld  [%fp - 12], %g3
cmp %g2, %g3
bne  .alarm
ID Behavior in Pipeline

```
ld  [%fp - 12], %g2
ld  [%fp - 12], %g3
cmp  %g2, %g3
bne  .alarm

<table>
<thead>
<tr>
<th>F</th>
<th>D</th>
<th>A</th>
<th>E</th>
<th>M</th>
<th>X</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>LD1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>LD1</td>
<td>LD1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>LD2</td>
<td>LD2</td>
<td>LD1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td>LD2</td>
<td>LD1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Clock cycles

Data Dependency

Branch Interlock
```
ID Behavior in Pipeline

```
ld [%fp - 12], %g2
ld [%fp - 12], %g3
cmp %g2, %g3
bne .alarm
```

Clock cycles:
- 0: LD1
- 1: LD2, LD1
- 2: CMP, LD2, LD1
- 3: BNE, CMP, LD2, LD1
- 4: BNE, CMP, CMP, LD2, LD1
- 5: NOP, BNE, CMP, CMP, LD2, LD1
- 6: NOP, BNE, BNE, CMP, CMP, LD2, LD1

Data Dependency:
- LD1 depends on [%fp - 12], %g2
- LD2 depends on [%fp - 12], %g3
- CMP depends on %g2, %g3
- BNE depends on LD1

Branch Interlock:
- NOP before BNE
- BNE before CMP
- CMP before LD2, LD1

ID Behavior in Pipeline

ld [%fp - 12], %g2
ld [%fp - 12], %g3
cmp %g2, %g3
bne .alarm
ID Behavior in Pipeline

ld [%fp - 12], %g2
ld [%fp - 12], %g3
cmp %g2, %g3
bne .alarm

Clock cycles:
- 0: LD1
- 1: LD2, LD1
- 2: CMP, LD2, LD1
- 3: BNE, CMP, LD2, LD1
- 4: BNE, CMP, LD2, LD1
- 5: NOP, BNE, CMP, CMP, LD2, LD1
- 6: NOP, BNE, BNE, CMP, CMP, LD2, LD1
- 7: NOP, BNE, BNE, BNE, CMP, CMP, LD2
- 8: RET, NOP, BNE, BNE, BNE, CMP, CMP
- 9: RET, NOP, BNE, BNE, CMP

Pipeline stages:
- F
- D
- A
- E
- M
- X
- W

Data Dependency:

Branch Interlock:
- 5: NOP, BNE, CMP, CMP, LD2, LD1
- 6: NOP, BNE, BNE, CMP, CMP, LD2, LD1
- 7: NOP, BNE, BNE, BNE, CMP, CMP, LD2

Branch Interlock:
- 8: RET, NOP, BNE, BNE, BNE, CMP, CMP
- 9: RET, NOP, BNE, BNE, CMP
Fault Model

- Inject fault in %g2
- Avoid raising the alarm

![Diagram showing the fault model with nodes LD1, LD2, CMP, BNE, and Memory address [%FP-12].]
Scenario 1: **Single Glitch**
- Instruction Fault in CMP
- Computation Fault in LD1

<table>
<thead>
<tr>
<th>Stage</th>
<th>F</th>
<th>D</th>
<th>A</th>
<th>E</th>
<th>M</th>
<th>X</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>LD1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>LD2</td>
<td>LD1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CMP</td>
<td>LD2</td>
<td>LD1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>BNE</td>
<td><strong>CMP</strong></td>
<td>LD2</td>
<td>LD1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>BNE</td>
<td>CMP</td>
<td><strong>CMP</strong></td>
<td>LD2</td>
<td>LD1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>NOP</td>
<td><strong>BNE</strong></td>
<td>CMP</td>
<td><strong>CMP</strong></td>
<td>LD2</td>
<td>LD1</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>NOP</td>
<td><strong>BNE</strong></td>
<td><strong>BNE</strong></td>
<td>CMP</td>
<td><strong>CMP</strong></td>
<td>LD2</td>
<td>LD1</td>
</tr>
<tr>
<td>7</td>
<td>NOP</td>
<td><strong>BNE</strong></td>
<td><strong>BNE</strong></td>
<td>CMP</td>
<td><strong>BNE</strong></td>
<td>CMP</td>
<td>LD2</td>
</tr>
<tr>
<td>8</td>
<td>RET</td>
<td>NOP</td>
<td><strong>BNE</strong></td>
<td><strong>BNE</strong></td>
<td><strong>BNE</strong></td>
<td>CMP</td>
<td><strong>CMP</strong></td>
</tr>
<tr>
<td>9</td>
<td>RET</td>
<td>NOP</td>
<td><strong>BNE</strong></td>
<td><strong>BNE</strong></td>
<td><strong>BNE</strong></td>
<td>CMP</td>
<td><strong>CMP</strong></td>
</tr>
</tbody>
</table>

**Data Dependency**

**Branch Interlock**
Fault Model

- Inject fault in \%g2
- Avoid raising the alarm
Attack Scenario

Scenario 2: Single Glitch
- Instruction Fault in BNE
- Computation Fault in LD1
Scenario 3: **Multiple Glitch**

- Computation Fault in LD1
- Instruction Fault in BNE

<table>
<thead>
<tr>
<th>clock cycles</th>
<th>F</th>
<th>D</th>
<th>A</th>
<th>E</th>
<th>M</th>
<th>X</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>LD1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>LD2</td>
<td>LD1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CMP</td>
<td>LD2</td>
<td>LD1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>BNE</td>
<td>CMP</td>
<td>LD2</td>
<td>LD1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>BNE</td>
<td>CMP</td>
<td>CMP</td>
<td>LD2</td>
<td>LD1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>NOP</td>
<td>BNE</td>
<td>CMP</td>
<td>CMP</td>
<td>LD2</td>
<td>LD1</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>NOP</td>
<td>BNE</td>
<td>BNE</td>
<td>CMP</td>
<td>CMP</td>
<td>LD2</td>
<td>LD1</td>
</tr>
<tr>
<td>7</td>
<td>NOP</td>
<td>BNE</td>
<td>BNE</td>
<td>BNE</td>
<td>CMP</td>
<td>CMP</td>
<td>LD2</td>
</tr>
<tr>
<td>8</td>
<td>RET</td>
<td>NOP</td>
<td>BNE</td>
<td>BNE</td>
<td>BNE</td>
<td>CMP</td>
<td>CMP</td>
</tr>
<tr>
<td>9</td>
<td>RET</td>
<td>NOP</td>
<td>BNE</td>
<td>BNE</td>
<td>BNE</td>
<td>CMP</td>
<td></td>
</tr>
</tbody>
</table>
Experimental Setup

- Spartan-6 XC6SLX75
  - Debug Support Unit (DSU)
  - Instruction Trace Buffer (ITB)
  - Pipeline Trace Register (PTR)
  - DUT (LEON3)

- USB/Serial i/f (FTDI)

- Clock Glitch Controller
- Clock Glitch Injector

- Glitch Injector
- Glitch-free clock

- Pulse Generator (Agilent 81110A)

- Python Scripts

- GRMON Debug Monitor

- Control PC

- JTAG

- USB

- Glitch Trigger
  - Control
  - Capture
  - Glitchy clock
Verification on Prototype

```plaintext
ld [%fp - 12], %g2
ld [%fp - 12], %g3
cmp %g2, %g3
bne .error
```

<table>
<thead>
<tr>
<th>Glitch FI (ns)</th>
<th>Impacted Instruction</th>
<th>Fault Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scenario 1</td>
<td>9 – 12.6</td>
<td>LD1 (A) CMP (D)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Faulty %g2 CMP → SRL</td>
</tr>
<tr>
<td>Scenario 2</td>
<td>12 – 14.6</td>
<td>LD1 (M) BNE (F)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Faulty %g2 BNE → NOP</td>
</tr>
<tr>
<td>Scenario 3</td>
<td>14.7-14.9 12.6 – 13.5</td>
<td>LD1 (E) BNE (D)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Faulty %g2 BNE → NOP</td>
</tr>
</tbody>
</table>
Experimental Results

• We applied the same strategy to several software countermeasures including
  • Instruction Duplication
  • Instruction Triplication
  • Parity Checking
  • Instruction Skip Countermeasures

• We successfully launched the DFIA attack on a software implementation of the LED algorithm, protected with parity checking and Instruction Duplication.

• We changed the width of the glitch from 31.2\text{ns} to 33.6\text{ns} with step size of 0.1\text{ns}. Using these fault injections, we obtained 5 faulty ciphertexts and retrieved 2 nibbles of the key.
Conclusions

• Efficient fault attacks on embedded software consider:
  • Architectural properties
  • Micro-architectural properties

• Microprocessor Fault Sensitivity Model is instrumental to predict the fault response of software.

• Traditional software countermeasures are vulnerable to single glitch fault attacks.
Thank you!

N. F. Ghalaty
farhady@vt.edu
• Case Study:
  • Fault Analysis: Differential Fault Intensity Analysis (DFIA)
  • Software: LED
  • Hardware: LEON3 Processor

• DFIA [Ghalaty et. al, FDTC’14]:
  • Relies on a biased fault behavior
  • Gradual fault behavior in proportion to the fault intensity
• Parity Checking Countermeasure
  • Algorithm Level
  • Exploits a parity prediction circuit to predict parity for each input, raises an alarm if the computed parity does not match the predicted one

• Instruction Duplication
  • Instruction Level
  • Duplicates selected critical parts of the code such as Add Round Key Function
How to Tune Glitch Parameters?

- Microprocessor Fault sensitivity Model of each (instruction, pipeline stage)

<table>
<thead>
<tr>
<th>Pipeline Stages</th>
<th>Critical Path Delays (Fault Sensitivity)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDI</td>
<td>Critical Path Delays (Fault Sensitivity)</td>
</tr>
<tr>
<td>F</td>
<td>3.53ns</td>
</tr>
<tr>
<td>D</td>
<td>3.26ns</td>
</tr>
<tr>
<td>A</td>
<td>5.62ns</td>
</tr>
<tr>
<td>E</td>
<td>5.20ns</td>
</tr>
<tr>
<td>M</td>
<td>7.58ns</td>
</tr>
<tr>
<td>X</td>
<td>3.17ns</td>
</tr>
<tr>
<td>W</td>
<td>4.45ns</td>
</tr>
</tbody>
</table>
; Unprotected code for AddRoundConstant
LDD [%fp + -56], %g2 ;LDD1
XOR %o4, %g2, %g2
XOR %o5, %g3, %g3
STD %g2, [%fp + -56]

; Instruction Duplication on LDD1
LDD [%fp-56], %g2 ;LDD1
LDD [%fp-56], %g4 ;LDD2
CMP %g2, %g4
BNE .error
Pipeline behavior of ID for LED

<table>
<thead>
<tr>
<th>Glitch width (ns)</th>
<th>Effected Inst. in Pipeline</th>
<th>Observed Faulty Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>31.2-33.6</td>
<td>LDD1, W</td>
<td>Fault in %g2</td>
</tr>
<tr>
<td></td>
<td>LDD2, M</td>
<td>Fault in %g4</td>
</tr>
<tr>
<td></td>
<td>BNE, F</td>
<td>BNE to OR</td>
</tr>
</tbody>
</table>
Fault Attack on Microprocessors

• Traditional Fault Attacks

1. Cryptographic Algorithm

2. Fault Model

3. Fault Injection

4. Faulty Output
• Traditional Fault Attacks

1. Cryptographic Algorithm
2. Fault Model
3. Micro-architectural Characteristics
4. Pipeline Fault Sensitivity Model
5. Fault Injection
6. Faulty Output