Fault injection and countermeasures

Jasper van Woudenberg
CTO North America
Riscure?

- Independent security lab in San Francisco, Ca and Delft, The Netherlands
  - Founded 2001, now ~45 employees
- Testing
  - Chips and devices with security function
  - Logical, physical, side channel testing
  - Test-to-break testing approach
- Products
  - Inspector: SCA + FI platform
- Research and development
  - Academic publications
  - Tool development
Xbox 360 glitch

Welcome on the Official Website of the x360Glitchip
Overview

• Faults in nature
• Malicious faults
• Attack tools
• Fault effects
• Countermeasures
• Conclusions
What are faults

- Faults cause a chip to malfunction
- Aim for:
  - Temporary faults
  - ‘Small’ errors
Natural faults

• Most fault types also occur in ‘nature’
• Radiation
• Electrical noise
• IC overheating
Radiation effects

- Photons are absorbed by electrons in silicon
- Absorbed photons increase electron energy
  - Increasing the semiconductor conductivity
- Can produce temporary faults
Simulating radiation

- Radiation hardening has been studied for military and (aero)space purposes
- Effects of many types of radiation can be simulated using (pulsed) lasers
- Cause gates to become conductive, and cause temporary faults
Electrical glitches

- A connected IC is disturbed by all electrical devices connected to it
- Disturbed by EM radiation
- Enter decoupling capacitors, shielding
- What if those aren’t there?
Glitch effects

- External clock source glitches
  - Spikes or dips cause temporary extra cycle
  - Instruction / data corruption
- Power supply glitches
  - Long: reset (boring)
  - Short: corruption (interesting)
Temperature faults

- IC running outside of specified temperature range will start to malfunction
- E.g. memory writes or reads will fail
  - Read all 00 / FF
  - No actual write happening
Accidental faults ...

- Discussed ‘natural’ fault causes
- IC’s made fault tolerant
  - Radiation hardening
  - Decoupling capacitors
  - Temperature sensors
  - ...

… vs malicious faults

- Fault tolerance ≠ tamper resistance
- Attackers can very specifically tune fault injection
  - Thereby specifically tune corruption
- Circumvent fault protection
- Circumvent security mechanisms!
Electrical attack tools

- Arbitrary waveform generators
- In our lab:
  - FPGA based solution with 500MHz DA converters
  - Glitches clock or supply voltage
  - Arbitrary patterns @2ns
- Or: ‘unloopers’
EM pulse generator

- EM pulses cause faults
  - From ~>140V
- Different probes have different effectiveness
  - Current research
Clock attack example

- Clock glitches cause one or multiple very fast clocks
- PC increases, but instruction does not write back…
Voltage attack example

- Reading / writing a value to memory takes a certain amount of power
- Drop the supply voltage during memory operations…

Threshold of read value

A power dip at the moment of reading a memory cell
Radiation attack tools

- Lasers have the same effect on IC’s as several radiation types
- Modern diode lasers have high degree of control
  - Timing, spot size, power
  - Our setup: 50ns, 6 x 1.4um, 20W(!)
Front side vs back side
What to do with faults

• What if an attacker can precisely pick instruction / data access to fault?
• Skip branch instructions
• Inject errors in crypto (DFA)
• ...

print("Pin accepted!");
3e:   bf 00 00 00 00 
43:   b8 00 00 00 00 
48:   e8 00 00 00 00 
4d:   eb 0f 

Skip branch instructions

- vi pin.c

```c
int main(int argc, char **argv) {
    if (strcmp(argv[0], "1234") == 0) {
        printf("Pin accepted!\n");
    } else {
        printf("Pin rejected!\n");
    }
    return 0;
}
```
Skip branch instructions

```
3a: 85 c0
3c: 75 11

    test %eax,%eax
    jne 4f <main+0x4f>

    printf("Pin accepted!"): 3e: bf 00 00 00 00
                           mov $0x0,%edi
                           mov $0x0,%eax
                           callq 4d <main+0x4d>
                           jmp 5e <main+0x5e>

} else {

    printf("Pin rejected!"); 4f: bf 00 00 00 00
                           mov $0x0,%edi
                           mov $0x0,%eax
                           callq 5e <main+0x5e>

    return 0;

5e: b8 00 00 00 00

} 63: c9
    leaveq
64: c3
    retq
```
Differential fault analysis

- Corruption in crypto can lead to key leakage
- DES, AES require few dozen to a few hundred successful corruptions to retrieve key
- RSA-CRT requires ONE successful corruption to retrieve key (Bellcore attack)
CRT implementation of RSA

- Efficient signing implementation splits exponentiation
- \( d_p = d \mod (p-1) \)
- \( d_q = d \mod (q-1) \)
- \( K = p^{-1} \mod q \)
- \( S_p = M^{d_p} \mod p \)
- \( S_q = M^{d_q} \mod q \)
- \( S = ( ( (S_q - S_p) * K ) \mod q ) * p + S_p \)
DFA on RSA-CRT

Inject a fault during CRT that corrupts $S_q$:

$S'_q$ is a corrupted result of $S_q$ computation

$S' = ((S'_q - S_p) K \mod q) \cdot p + S_p$

Subtract $S'$ from $S$:

$S - S' = (((S_q - S_p) K \mod q) \cdot p - (((S'_q - S_p) K \mod q) \cdot p$

$= (x_1 - x_2) \cdot p \mod N$

compute $\text{Gcd}(S - S', n) = \text{Gcd}((x_1 - x_2) \cdot p, p \cdot q) = p$

compute $q = n \div p$
Finding fault parameters

- Successful fault requires many parameters to be tuned correctly
- Know (or guess) from design, source, experience
- Temporary fault allows reset & retry
- Scan over parameter ranges
### Finding fault parameters

![VC Gitcher report - DES perturbation module (started 2009-04-29 09:46:54)](image)

<table>
<thead>
<tr>
<th>ut</th>
<th>Glitch offset</th>
<th>Glitch length</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>262</td>
<td>6</td>
<td>8E 81 31 FF 45 4A 43 4F 30 33 31 06 00 00 0E A0 04 00 00 3B 39 D7 EA FA E4 ED A3 00 31 00 00 0A C5 D9 1B 0E 2A D5 6F BE 90 00 BD</td>
<td></td>
</tr>
<tr>
<td>263</td>
<td>9</td>
<td>8E 81 31 FF 45 4A 43 4F 30 33 31 06 00 00 0E A0 04 00 00 3B 39 D7 EA FA E4 ED A3 00 31 00 00 0A C5 D9 1B 0E 2A D5 6F BE 90 00 BD</td>
<td></td>
</tr>
<tr>
<td>264</td>
<td>5</td>
<td>8E 81 31 FF 45 4A 43 4F 30 33 31 06 00 00 0E A0 04 00 00 3B 39 D7 EA FA E4 ED A3 00 31 00 00 0A C5 D9 1B 0E 2A D5 6F BE 90 00 BD</td>
<td></td>
</tr>
<tr>
<td>265</td>
<td>9</td>
<td>8E 81 31 FF 45 4A 43 4F 30 33 31 06 00 00 0E A0 04 00 00 3B 39 D7 EA FA E4 ED A3 00 31 00 00 0A C5 D9 1B 0E 2A D5 6F BE 90 00 BD</td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>9</td>
<td>8E 81 31 FF 45 4A 43 4F 30 33 31 06 00 00 0E A0 04 00 00 3B 39 D7 EA FA E4 ED A3 00 31 00 00 0A C5 D9 1B 0E 2A D5 6F BE 90 00 BD</td>
<td></td>
</tr>
<tr>
<td>267</td>
<td>5</td>
<td>8E 81 31 FF 45 4A 43 4F 30 33 31 06 00 00 0E A0 04 00 00 3B 39 D7 EA FA E4 ED A3 00 31 00 00 0A C5 D9 1B 0E 2A D5 6F BE 90 00 BD</td>
<td></td>
</tr>
<tr>
<td>268</td>
<td>9</td>
<td>8E 81 31 FF 45 4A 43 4F 30 33 31 06 00 00 0E A0 04 00 00 3B 39 D7 EA FA E4 ED A3 00 31 00 00 0A C5 D9 1B 0E 2A D5 6F BE 90 00 BD</td>
<td></td>
</tr>
<tr>
<td>269</td>
<td>5</td>
<td>8E 81 31 FF 45 4A 43 4F 30 33 31 06 00 00 0E A0 04 00 00 3B 39 D7 EA FA E4 ED A3 00 31 00 00 0A C5 D9 1B 0E 2A D5 6F BE 90 00 BD</td>
<td></td>
</tr>
<tr>
<td>270</td>
<td>9</td>
<td>8E 81 31 FF 45 4A 43 4F 30 33 31 06 00 00 0E A0 04 00 00 3B 39 D7 EA FA E4 ED A3 00 31 00 00 0A C5 D9 1B 0E 2A D5 6F BE 90 00 BD</td>
<td></td>
</tr>
<tr>
<td>271</td>
<td>5</td>
<td>8E 81 31 FF 45 4A 43 4F 30 33 31 06 00 00 0E A0 04 00 00 3B 39 D7 EA FA E4 ED A3 00 31 00 00 0A C5 D9 1B 0E 2A D5 6F BE 90 00 BD</td>
<td></td>
</tr>
<tr>
<td>272</td>
<td>9</td>
<td>8E 81 31 FF 45 4A 43 4F 30 33 31 06 00 00 0E A0 04 00 00 3B 39 D7 EA FA E4 ED A3 00 31 00 00 0A C5 D9 1B 0E 2A D5 6F BE 90 00 BD</td>
<td></td>
</tr>
<tr>
<td>273</td>
<td>5</td>
<td>8E 81 31 FF 45 4A 43 4F 30 33 31 06 00 00 0E A0 04 00 00 3B 39 D7 EA FA E4 ED A3 00 31 00 00 0A C5 D9 1B 0E 2A D5 6F BE 90 00 BD</td>
<td></td>
</tr>
<tr>
<td>274</td>
<td>9</td>
<td>8E 81 31 FF 45 4A 43 4F 30 33 31 06 00 00 0E A0 04 00 00 3B 39 D7 EA FA E4 ED A3 00 31 00 00 0A C5 D9 1B 0E 2A D5 6F BE 90 00 BD</td>
<td></td>
</tr>
<tr>
<td>275</td>
<td>5</td>
<td>8E 81 31 FF 45 4A 43 4F 30 33 31 06 00 00 0E A0 04 00 00 3B 39 D7 EA FA E4 ED A3 00 31 00 00 0A C5 D9 1B 0E 2A D5 6F BE 90 00 BD</td>
<td></td>
</tr>
<tr>
<td>276</td>
<td>9</td>
<td>8E 81 31 FF 45 4A 43 4F 30 33 31 06 00 00 0E A0 04 00 00 3B 39 D7 EA FA E4 ED A3 00 31 00 00 0A C5 D9 1B 0E 2A D5 6F BE 90 00 BD</td>
<td></td>
</tr>
<tr>
<td>277</td>
<td>5</td>
<td>8E 81 31 FF 45 4A 43 4F 30 33 31 06 00 00 0E A0 04 00 00 3B 39 D7 EA FA E4 ED A3 00 31 00 00 0A C5 D9 1B 0E 2A D5 6F BE 90 00 BD</td>
<td></td>
</tr>
<tr>
<td>278</td>
<td>9</td>
<td>8E 81 31 FF 45 4A 43 4F 30 33 31 06 00 00 0E A0 04 00 00 3B 39 D7 EA FA E4 ED A3 00 31 00 00 0A C5 D9 1B 0E 2A D5 6F BE 90 00 BD</td>
<td></td>
</tr>
<tr>
<td>279</td>
<td>5</td>
<td>8E 81 31 FF 45 4A 43 4F 30 33 31 06 00 00 0E A0 04 00 00 3B 39 D7 EA FA E4 ED A3 00 31 00 00 0A C5 D9 1B 0E 2A D5 6F BE 90 00 BD</td>
<td></td>
</tr>
</tbody>
</table>

Public 2/9/2012
Tamper resistant IC’s

• Smart cards have traditionally been designed with tamper resistance
• Detection & prevention mechanisms for fault injection attacks
• Countermeasures at both software and hardware level
Countermeasures hardware

- Active and passive shielding
- Supply voltage monitoring, buffering
- Temperature sensors
- Internal clock (unstable)
- Optical sensors on die
Countermeasures software

- Erase secrets when tripping hardware countermeasure
- Multiple checking sensitive results
- Branchless algorithms
- Random wait loops
- Usually risk based because of (computational) cost
Beating multi checking

- Need to hit target… multiple times
- Precise timing gets more critical
Synchronized trigger

- Nice pattern before (and after) DES
Multi-area checks

• Main CPU instructs crypto core to execute crypto n times
• Checks result is the same
• To perform DFA:
  • Fault crypto core in same way -> difficult
  • Fault crypto core AND attack result check on main CPU -> two locations!
Beating multi-area

- Hit it with two lasers!
Countermeasures

- Secure systems use both hardware and software countermeasures
- Forcing an attacker to require multiple faults makes attacking exponentially harder
  - Success rate exponentially decreases with attempts, e.g. 10%, 1%, 0.1%, …
- For each countermeasure there is an attack… ongoing “cat & mouse game”
Smart card vs embedded CPU

• Embedded CPUs and firmware typically not (yet) designed with fault attacks in mind
  • We see little evidence of countermeasures ‘in the field’
• Extra difficulties wrt smart cards
  • High current draw
  • Many cores
  • Higher clock speeds
  • No standardized interfaces
Where do we stand?

- Smart cards are getting progressively harder to break
  - Though we still do
- We regularly break embedded chips
  - Voltage glitching, laser attacks
  - Mobile phone SoC, STB SoC, microcontrollers
- Usually with little design information
- Difficulties lie mostly in getting the fault injection setup to sync with process
Xbox 360 reset glitch

• “We found that by sending a tiny reset pulse to the processor while it is slowed down does not reset it but instead changes the way the code runs, it seems it's very efficient at making bootloaders memcmp functions always return "no differences".”
  • http://www.free60.org/Reset_Glitch_Hack#The_reset_glitch_in_a_few_words
Conclusion

• Fault attacks are a serious threat to any secure system
• “Next logical step” for an attacker if logical flaws absent
  • Already seen in practice
• Countermeasures have been studied and are available!
  – See e.g. our “Secure application programming in the presence of side channel attacks” paper
  – Check patents for hardware countermeasures
What can academia/labs do?

• Research!

• Effectiveness of different EM FI probes
• How do optical emissions relate to optical faults?
• How to reduce the FI parameter search space?
• Relation between feature size and optical fault sensitivity
• Efficient generic countermeasures
References

- Xbox 3660 glitch site http://www.free60.org/Reset_Glitch_Hack#The_reset_glitch_in_a_few_words
Challenge your security

Contact:  Jasper van Woudenberg
CTO North America
jasper@riscure.com

Riscure B.V.
Frontier Building, Delftechpark 49
2628 XJ Delft
The Netherlands
Phone: +31 15 251 40 90

Riscure North America
71 Stevenson Street, Suite 400
San Francisco, CA 94105
USA
Phone: +1 650 646 99 79

www.riscure.com
inforequest@riscure.com