Extracting keys from FPGAs, OTP Tokens and Door Locks

Side-Channel (and other) Attacks in Practice

David Oswald

david.oswald@rub.de
No, I did not do all this stuff alone

- Christof Paar
- Benedikt Driessen
- Timo Kasper
- Gregor Leander
- Amir Moradi
- Falk Schellenberg
- Daehyun Strobel
- Pawel Swierczynski
- Bastian Richter

If you wondered about my shirt: http://fb.com/WorldBeatClubTanzenUndHelfen
Ruhr-University Bochum: beautiful.
Announcement

- Timo at 29C3: „ChameleonMini in 2013“
- As of December 22, 2013:
  
  https://github.com/skuep/ChameleonMini
Embedded systems everywhere
(The life of) a typical pirate

- Pirate hat
- Eye patch
- Pegleg
- Pirate laughter
Implementation Attacks:
...

Based on Skoborogatov
Principle of Side-Channel Analysis
(here: listen to sound)

A Bank Robbery
Principle of Side-Channel Analysis

The world is changing...
Principle of Side-Channel Analysis
(Now: measure the power consumption / EM)

The world is changing ... 

... the tools are, too.
Side-Channel Analysis: Leakage

Power consumption / EM depends on processed data

Data = 1111
Data = 1010
Data = 0000
Simple Power Analysis: Directly analyze (few) traces, for example RSA:
Differential Power Analysis

- Detect **statistical dependency**: Key guess ↔ Side-channel
- Idea: Brute-force w/ additional information
- Use a statistical test...
Then you have 2 theories that predict the data:

Wrong key candidate(s)

Correct key candidate

The problem is that the difference between the two is very small.

It's very hard to distinguish these two with our data.

The predicted effect is tiny.

100 – 1 mio. measurements

Open 24 hours

Over a Bijillion Collisions Served
Implementation Attacks:
From Theory to Practice
Case Studies

Altera Stratix II

Yubikey 2

Locking system
Case Studies

Altera Stratix II

Yubikey 2

Locking system
FPGAs widely used in
- Routers
- Consumer products
- Cars
- Military

Problem:
FPGA design (bitstream) can be easily copied
FPGA Power-Up

Flash → Bitstream → FPGA 1
Problem: Cloning

Flash → Bitstream → FPGA 1 → Clone → FPGA 2
Industry’s Solution

Flash → Encrypted bitstream → FPGA 1

Encrypted bitstream

FPGA 1
Industry’s Solution

Flash

Encrypted bitstream

FPGA 1

= ?
Related Work

- Bitstream encryption scheme of several Xilinx product lines broken
  - Virtex 2 (3DES)
  - Virtex 4 & 5 (AES256)
  - Spartan 6 (AES256)
- Method: Side-Channel Analysis (SCA)
What about Altera?

- **Target**: Stratix II
- **Bitstream encryption** („design security“) uses AES w/ 128-bit key
- **Side-Channel Analysis possible?**
- **Problem**: Proprietary and undocumented mechanisms for key derivation and for encryption
Reverse-Engineering

- Reverse-engineer proprietary mechanisms from Quartus II software
- IDA Pro (disassembler / debugger)
KEY1 128 Bit

KEY2 128 Bit

Black-box

Bitstream

KEY1 / KEY2 file for FPGA

Encrypted Bitstream
Key derivation
real key = f(KEY1,KEY2)

KEY1 / KEY2 file for FPGA

Encrypted Bitstream

KEY1 128 Bit

KEY2 128 Bit

Bitstream
Why this key derivation?

- Real key cannot be set directly
- Key derivation is performed once when programming the FPGA

**Idea:** When real key is extracted, KEY1 and KEY2 cannot be found

→ Prevent cloning: real key of blank FPGA cannot be set
„real key“ = AES_{KEY1}(KEY2)

Is $f(\text{KEY1,KEY2})$ „good“?
Good idea?

- In principle: Yes
- But: AES (in this form) is not one-way:
  - Pick any $\text{KEY}_1^*$
  - $\text{KEY}_2^* = \text{AES}_{\text{KEY}_1^*}^{-1}(\text{real key})$
  - This ($\text{KEY}_1^*, \text{KEY}_2^*$) leads to same real key
real key = $\text{AES}_{\text{KEY1}}(\text{KEY2})$
real key = \text{AES}_{\text{KEY1}}(\text{KEY2})

\text{KEY1} / \text{KEY2 file for FPGA} \quad \text{Encrypted Bitstream}

\text{Bitstream}
Encrypted block \(i = \text{AES128}_{\text{real key}}(\text{IV}_i) \oplus \text{plain block } i\)

Encryption method:
AES in Counter mode
Reverse-Engineering: Summary

- All „obscurity features“ reverse-engineered
- Further details: file format, coding, ...
- Black-box → white box
- Side-channel analysis possible
  (target: 128-bit real key)
Side-Channel Attack on Stratix II
Mean trace for unencrypted and encrypted bitstream
Mean trace for unencrypted and encrypted bitstream
Further experiments ...
Recover the 128-bit AES key with 30,000 traces (~ 3 hours of measurement)
Conclusion

- Full 128-bit AES key of Stratix II can be extracted using 30,000 traces (3 hours)
- Key derivation does not prevent cloning
- Proprietary security mechanisms can be reverse-engineered from software
- **Software reverse-engineering enables hardware attack**
Pirates not only hakk FPGAs
Case Studies

Altera Stratix II

Yubikey 2

Locking system
Token

Black-box

Auth. protocol

Door lock
Turning a Black-box into a White-box

Token

Door lock
Decapping an IC (1)

White Fuming Nitric Acid (99.5%)
Decapping an IC (2)
Decapping an IC (3)
Decapping an IC (4)
ASIC

- Gate Array
- 2µm technology
- 28 pads, 14 bonded
- Mixed-signal
- ~1700/2300 transistors utilized
ASIC – Logic Description
Turning a Black-box into a White-box
Microscopic View (1)
UV-C: Disable Read-Out Protection (1)
UV-C: Disable Read-Out Protection (2)
Extraction + Analysis of Embedded Code

- After read-out protection disabled: code readable with standard programmer
- Reverse-engineering (e.g. IDA Pro)
- After some time: all details of system known
- Black-box → white-box

```assembly
CODE:0234    call    i2c_read_W_byte   ; Reads W & 0x7F bytes
             ; Init I2C
             ; Set read address to Reg 71 / 70
             ; Stores (inverted) read data at location pointed to by 73 / 72
             ;
             ; Sets (75) <- W
             ; Sets bank to 0 (resets bit 5 and 6)
CODE:0235    bsf     BANK0_STATUS, 6
CODE:0236    tstf    byte_DATA_119
CODE:0237    skpnz
CODE:0238    endless_loop_1: ; CODE XREF: maybe_related_to_rewriting_program_memory:endless_
CODE:0238    b       endless_loop_1
CODE:0239    movlw 2
CODE:0239    call    read_value_from_eeprom ; Address in W
             ; Result in W
             ; Switches bank to 0
```
System Design: Weaknesses and Attacks (1)

- Each token has unique key $K_T$
- Each lock has **installation-wide** key $K_M$
- $K_T = f(K_M, ID_T) \rightarrow \text{single point of failure}$
- **Obtaining one lock gives access to all doors:**
  Read-out PIC (as explained before) or perform non-invasive side-channel attack
System Design: Weaknesses and Attacks (2)

- **Problem 1**: System uses proprietary cryptography with „bad“ mathematical properties
- **Problem 2**: Re-use of internal values as „random“ numbers
- **Result**: Mathematical attack allows to recover $K_T$ with 3 (unsuccessful) protocol runs with any door
Conclusion

- Adversary gains full access to any door
- Reasons for security flaws
  - Insecure hardware
  - Proprietary cryptography
  - „Bad“ system design
- **Hardware attacks:** Replace all devices (expensive)
- **Cryptanalytical attacks:** Firmware update (cheap)
- **Hardware reverse-engineering enables mathematical attacks**
Pirates not only hakk
FGA
Har
Har
Locks
Case Studies

Altera Stratix II

Yubikey 2

Locking system
Two-Factor Authentication

**Past:** One factor: Password/PIN

**Today:** Two factors: Password/PIN and additionally
Yubikey 2: Overview

- Simulates USB keyboard
- Generates and enters One-Time Password (OTP) on button press
- Based on AES w/ 128-bit key
Yubikey OTP Generation (1)

dhbgnhfhjcr1 dgbgnhfhjcr1 dgbgnhfhjcr1 dgbgnhfhjcr1 dgbgnhfhjcr1
gukndgttlehvetuunugglkfetdegj
trjddibkbugfhnevdebrddvhhhlullugh
judbdifkcchhjkitgvgvvbinbdiigfd
Yubikey OTP Generation (2)

<table>
<thead>
<tr>
<th>uid</th>
<th>useCtr</th>
<th>tstp</th>
<th>sessionCtrl</th>
<th>rnd</th>
<th>crc</th>
</tr>
</thead>
</table>

| 6   | 2      | 3    | 1           | 2   | 2   |

6 Bytes

\[ \text{AES-128 Encryption} \]

\[ \text{Modhex Encoding} \]
Yubikey Hardware
Measurement Setup

- Resistor in USB ground for power measurement
- EM measurement with near-field probe
- Connecting (capacitive) button to ground triggers the Yubikey
Power vs. EM Measurements

- Trigger on falling edge (Yubikey's LED off)
- EM yields better signal
- AES rounds clearly visible
Key Recovery (EM)

- Attacking final AES round
- Power model \( h_i = \text{HW}(\text{SBOX}^{-1}(C_i \oplus \text{rk})) \)
- \( \approx 700 \) traces needed
- \( \approx 1 \) hour for data acquisition
Implications

- 128-bit AES key of the Yubikey 2 can be recovered (700 EM measurements = 1 hour physical access)
- Attacker can compute OTPs w/o Yubikey
- Impersonate user:
  Username and password still needed
- Denial-of-Service:
  Send an OTP with highly increased useCtr

→ Improved FW version 2.4 for Yubikey 2
Responsible Disclosure
When pirates do good ...
Global Average Temperature vs. Number of Pirates

- Global Average Temperature (°C)
  - 16.5
  - 16.0
  - 15.5
  - 15.0
  - 14.5
  - 14.0
  - 13.5
  - 13.0

- Number of Pirates (Approximate)
  - 35000
  - 45000
  - 20000
  - 15000
  - 5000
  - 400
  - 17

- Data points:
  - 1820
  - 1860
  - 1880
  - 1920
  - 1940
  - 1980
  - 2000

By RedAndr, Wikimedia Commons
Responsible Disclosure

- **Locking system:**
  - Vendor informed ~ 1 year before
  - Deployed patch to fix mathematical attacks

- **Altera:**
  - Informed ~ 6 months before
  - Acknowledged our results

- **Yubikey:**
  - Informed ~ 9 months before
  - Improved firmware version 2.4
Countermeasures
Countermeasures

- Implementation attacks: Practical threat, but:

  - First line of defense: Classical countermeasures
    - Secure hardware (certified devices)
    - Algorithmic level

  - Second line of defense: System level
    - Detect: Shadow accounts, logging
    - Minimize impact (where possible): Key diversification
Different Scenarios, different threats

**Yubikey 2**
- Time per key: 1 h
- Diversified keys (?)
- Each token: new attack
  → Attack does not scale

**Locking system**
- Time per key: 15 min
- All doors: same key
- Attack one door
  → Attack scales
Thanks for your attention
Questions now?
or later: david.oswald@rub.de

If you wondered about my shirt:
http://fb.com/WorldBeatClubTanzenUndHelfen