TeleTrusT-interner Workshop

Bochum, 27./28.06.2013

Christof Paar
Horst Görtz Institute for IT-Security
Ruhr University Bochum
Impulsvortrag

Embedded Security for the Internet of Things

TeleTrusT Workshop
Zentrum für IT-Sicherheit
Bochum, 27.6. 2013

Christof Paar
Horst Görtz Institute for IT-Security
Ruhr University Bochum

Acknowledgement

- Benedikt Driessen
- Markus Kasper
- Timo Kasper
- Gregor Leander
- Amir Moradi
- David Oswald
- Axel Poschmann

Agenda

- General Thoughts on Embedded Security
- Constructive: Bar Codes and SP Ciphers
- Destructive 1: Cell phones in the Desert
- Destructive 2: Routers and AES
- Auxiliary stuff

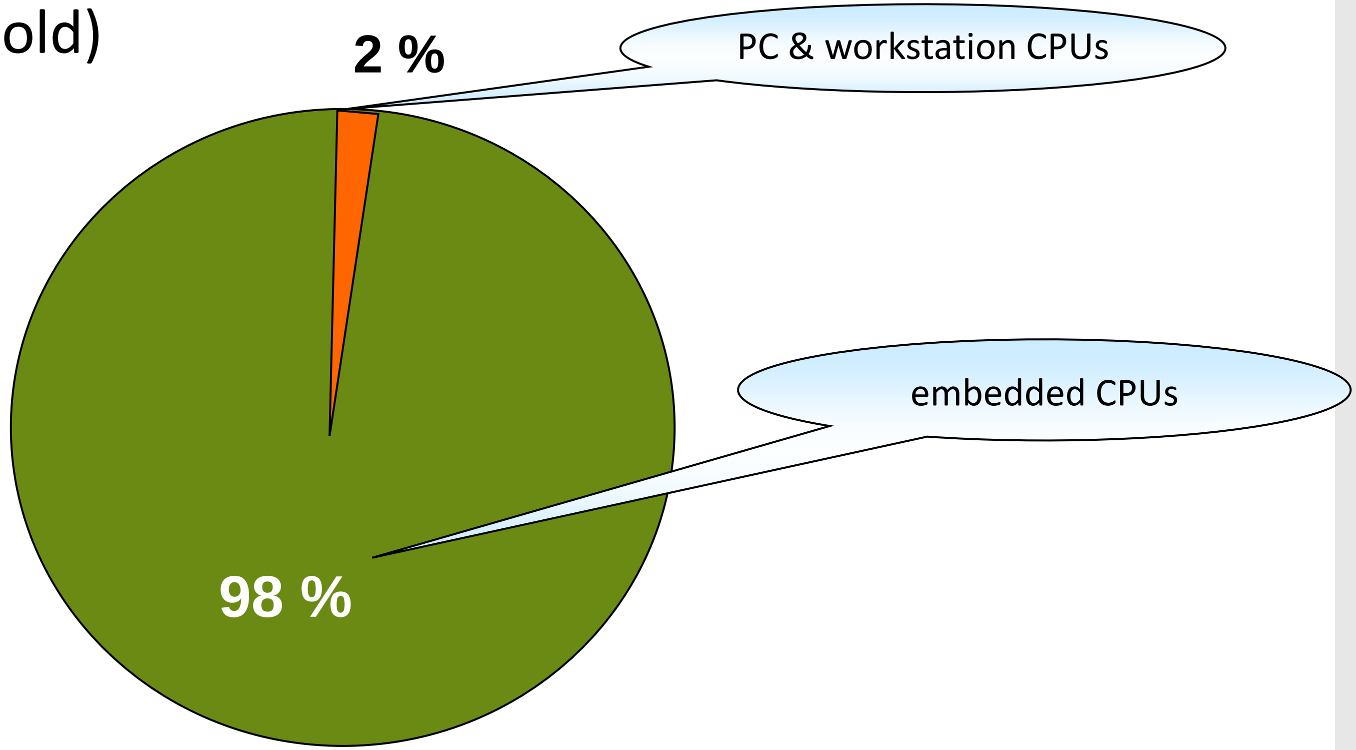
Agenda

RUB

- **General Thoughts on Embedded Security**
- Constructive: Bar Codes and SP Ciphers
- Destructive 1: Cell phones in the Desert
- Destructive 2: Routers and AES
- Auxiliary stuff

Who cares about *embedded systems*?

CPU market (units sold)



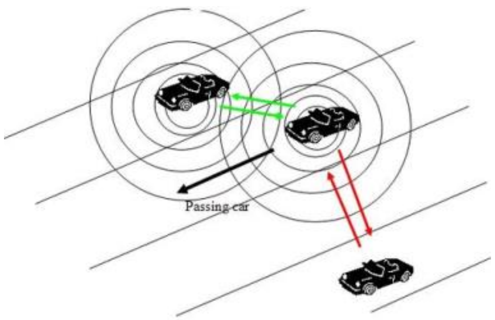
Q: But security ?

Embedded Security – Examples

Embedded DRM applications (iTunes, Kindle, …)

Telemedicine

Privacy & security of car2car communication



Electronic IDs and e-health cards

Agenda

- General Thoughts on Embedded Security
- **Constructive: Bar Codes and SP Ciphers**
- Destructive 1: Cell phones in the Desert
- Destructive 2: Routers and AES
- Auxiliary stuff

Lightweight Cryptography

RUB

1. “We need RFID security with less than 2000 gates”
   Sanjay Sarma, AUTO-ID Labs, CHES 2002

2. Securing sensor networks, e.g., infrastructure

3. US$ 3 trillions annually due to product piracy* (> US budget)

*Source: www.bascap.com

Needs authentication & identification

$\Rightarrow$ can both be fixed with standard cryptography

Strong Identification (symmetric crypto)

r

e_k (r) = y

1. random challenge r
2. encrypted response y
3. verification

   e_k (r) = y‘

   y == y‘

Challenge: Encryption function e() at extremely low cost

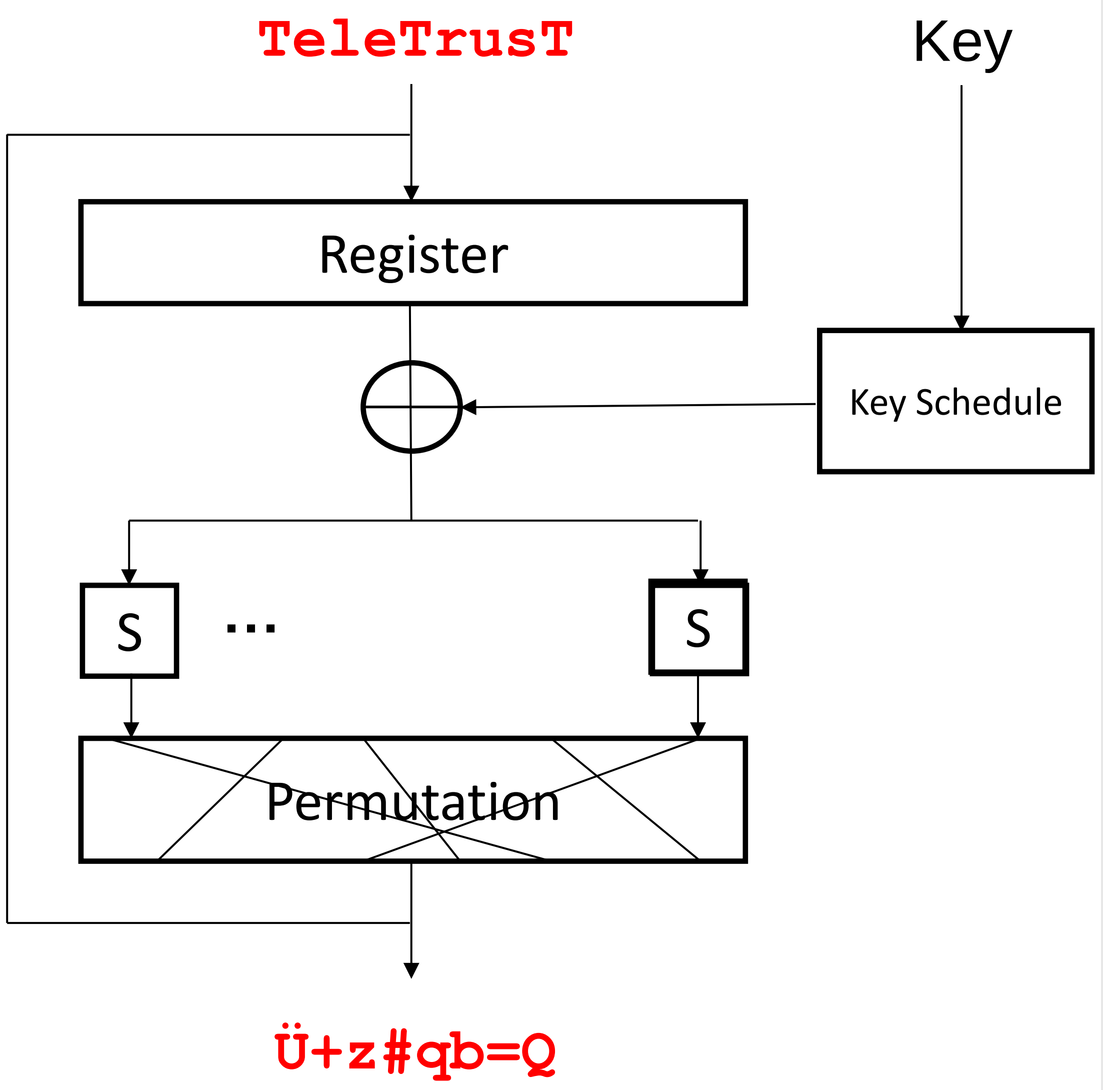
→ almost all existing ciphers not optimized for cost …

→ **Q: How cheap can we make cryptography?**

PRESENT – An agressively cost-otimized block cipher for RFID

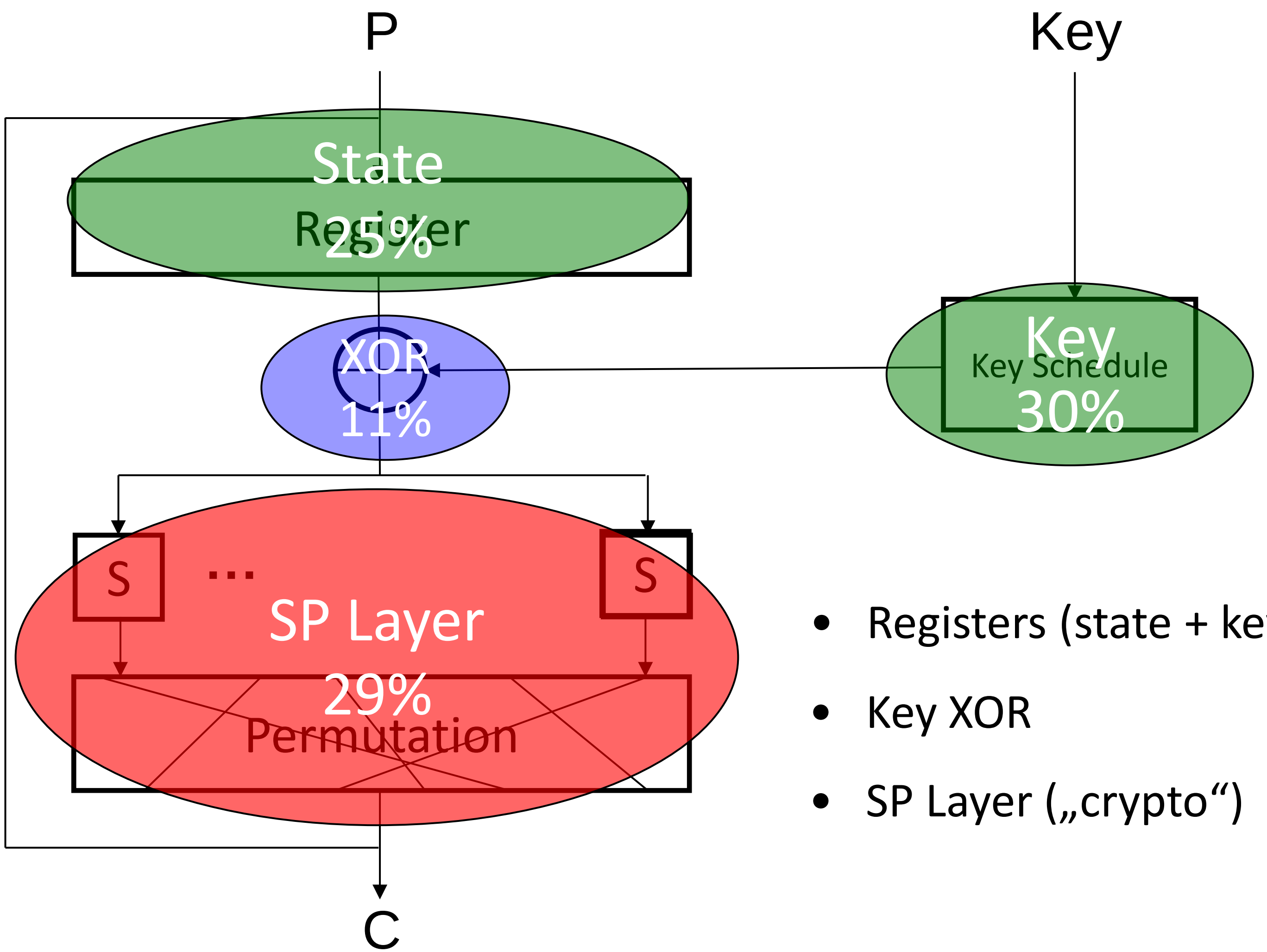
RUB

- 64 bit block, 80/128 bit key
- pure substitution-permutation network
- 4-4 bit Sbox
- 31 round (32 clks)
- secure against all known attacks
- joint work with Lars Knudsen, Gregor Leander, Matt Robshaw, …



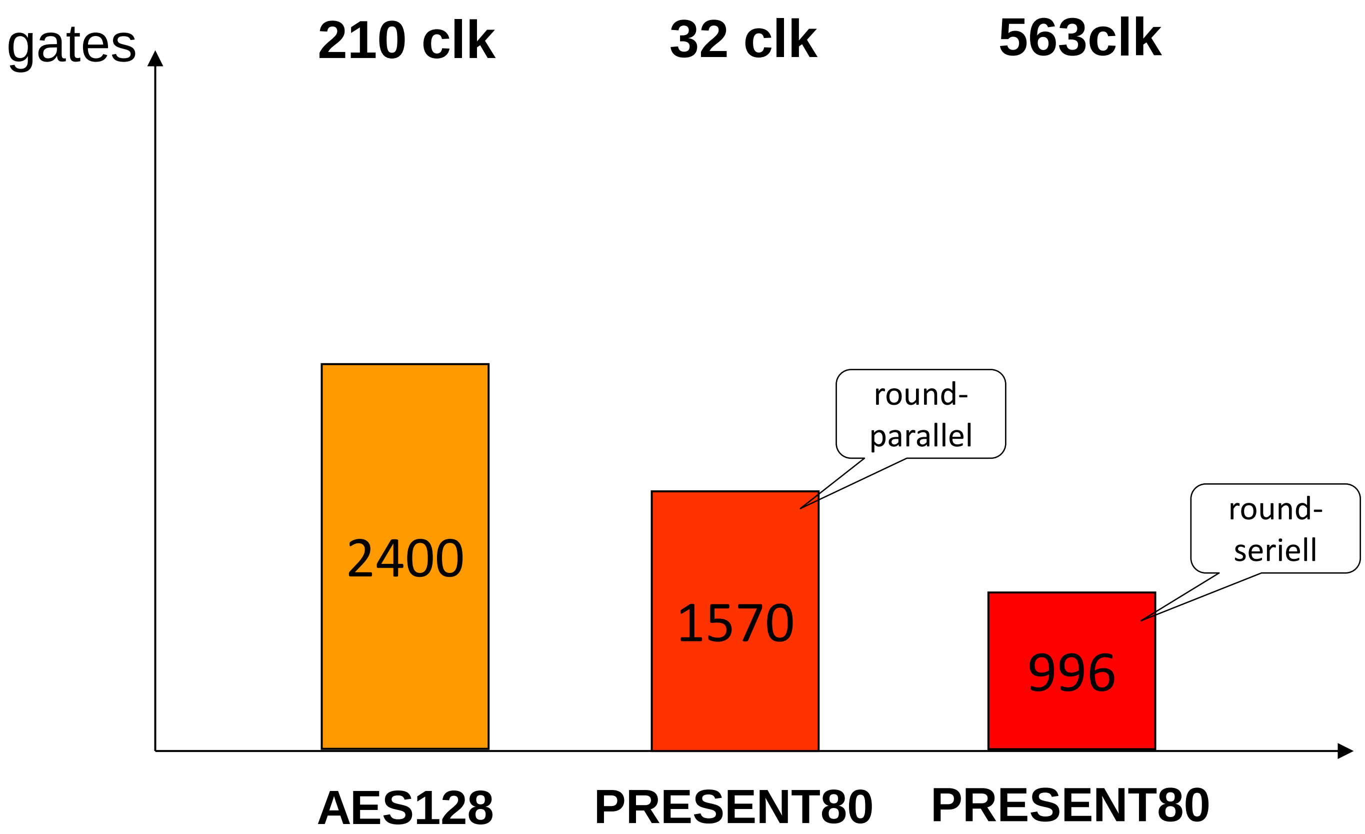
Resource use within PRESENT

Round-parallel implementation (1570ge)



- Registers (state + key) 55%
- Key XOR 11%
- SP Layer („crypto“) 29%

Results – PRESENT



- ≈ 90% less energy than small AES
- smallest secure cipher (20+ cryptanalytical publications)
- ISO standardized (2012)
- Serial implementation approaches theoretical complexity limit: almost all area is used for the 144 bit state (key + data path)
- Many related proposals: CLEFIA, Hight, KATAN, KTANTAN, Klein, mCrypton, Piccolo, …

Further Reading

RUB

- Bogdanov, Knudsen, Leander, P, Poschmann, Robshaw, Seurin, Vikkelsoe: PRESENT: An Ultra-Lightweight Block Cipher. CHES 2007.
- Rolfes, Poschmann, Leander, P: Ultra-Lightweight Implementations for Smart Devices – Security for 1000 Gate Equivalents. CARDIS 2008.
- Borghoff et al.: PRINCE - A Low-Latency Block Cipher for Pervasive Computing Applications. ASIACRYPT 2012.

Agenda

- General Thoughts on Embedded Security
- Constructive: Bar Codes and SP Ciphers
- **Destructive 1: Cell phones in the Desert**
- Destructive 2: Routers and AES
- Auxiliary stuff

Mobile Satellite Telephony

RUB

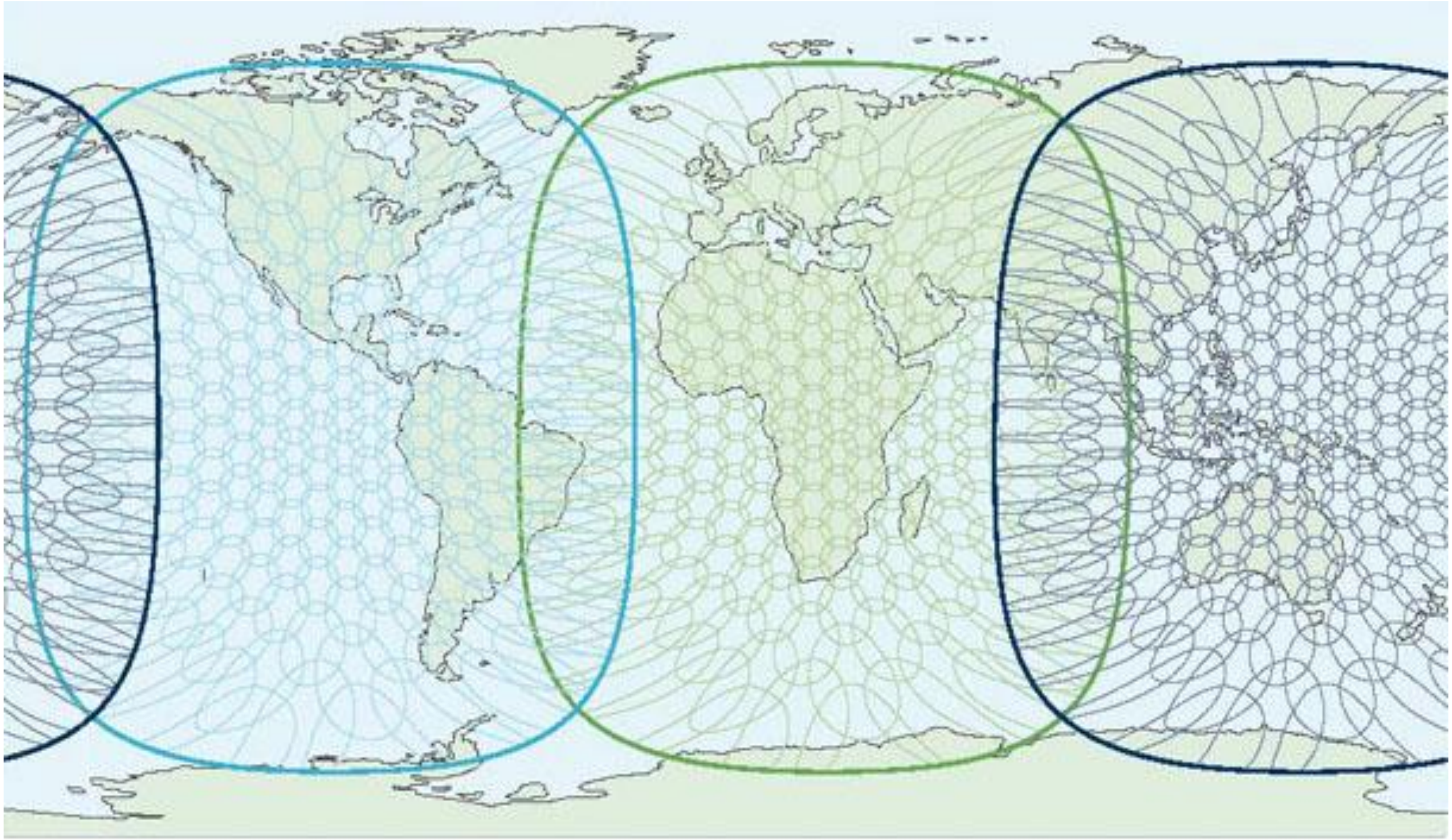
- Cellphone communication not available in many **remote places**
  - crew on oil rig or ships on open sea
  - airplanes
  - many humanitarian missions
  - research expeditions
  - and many military applications …

- **Satellite** telephony has been around since the 1990s
- **Direct** communication between phones and satellites

Satellite Phones

Many models

Inmarsat Spotbeam Coverage



Standards and Specifications

RUB

- Satellite phone systems standardized by **ETSI**
- 2 major standards coexist:
  - **GMR-1**: based on GSM, de-facto standard used by most providers
  - **GMR-2** (aka GMR-2+): based on GSM, used by Inmarsat (and ACeS)
- Specifications are **freely** available from ETSI
  - Both standards are very close to **GSM**
  - Cover signaling, encoding, etc.
  - … **except the security parts of the standard**

Is Satphone Communication Secure?

RUB

Starting situation

- GSM algorithms have been (essentially) **broken**
- Q: Are the GMR algorithms **vulnerable** to similar attacks?

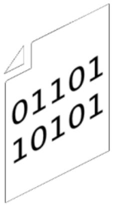
Research statement

Identify/extract the A5-GMR algorithms from satphones and perform cryptanalysis

Choosing a Target

- Several GMR-1 phones on the market
- Our victim: **Thuraya SO-2510** (for no specific reason)
- Firmware update publically available
- We didn't (have to) look at any other GMR-1 firmware
- Analysis was done **statically** only, we had no real phone at our disposal!

General Attack Procedure



1) Dump firmware image from firmware updater

2) Obtain information on the phone‘s actual hardware architecture

3) Dump DSP code from firmware image

4) Find cipher code

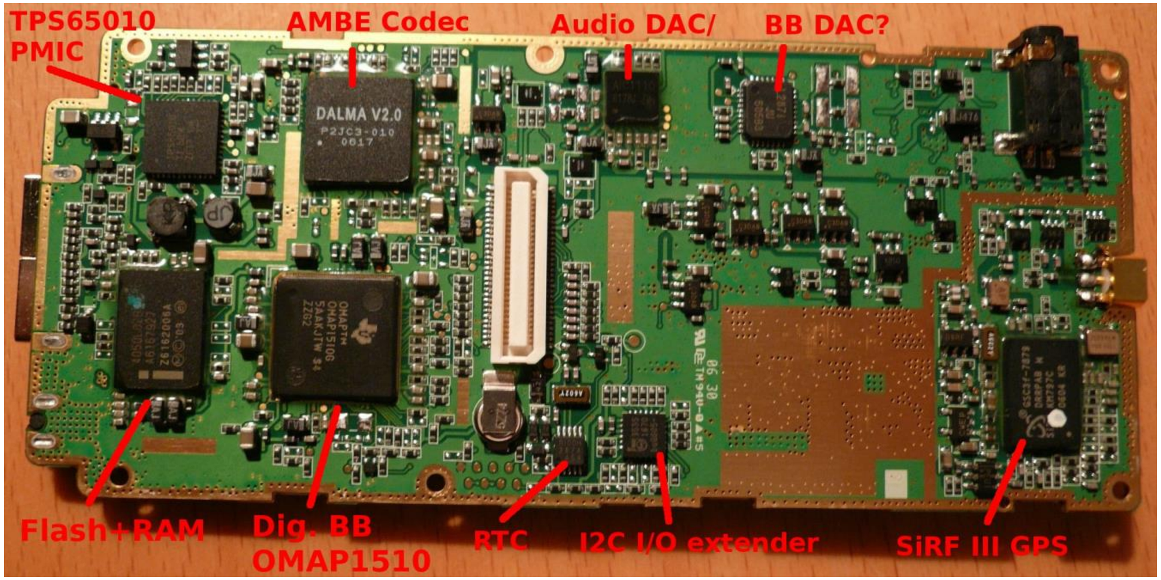
5) Perform cryptanalysis

Steps 1–4: Reverse engineering

Firmware and Hardware

RUB

- Firmware provided **unencrypted/unpacked** (some meta data needed to be stripped)
- Thuraya SO-2510 runs a TI **OMAP1510 platform** (aka OMAP5910): ARM + TI C55X DSP
- fairly well documented platform
- OS is **VxWorks**



Strings

- Surpisingly, firmware image contains plenty of **assertion/log strings**
- Allows to deduce the name of some functions

```
ROM:0139B094 ; =============== S U B R O U T I N E =======================================
ROM:0139B094
ROM:0139B094
ROM:0139B094 vptr_slist_last                         ; CODE XREF: v_grm_dl_initialise_reassembly_buffer+1E4↓p
ROM:0139B094
ROM:0139B094 var_14          = -0x14
ROM:0139B094
ROM:0139B094                 STMFD   SP!, {R4-R6,LR}
ROM:0139B098                 SUB     SP, SP, #4
ROM:0139B09C                 MOV     R5, R0
ROM:0139B0A0                 MOV     R4, R1
ROM:0139B0A4                 MOV     R6, #0
ROM:0139B0A8                 LDR     R3, =aEnteringFunctionVptr__0 ; "Entering function vptr_slist_last()"
ROM:0139B0AC                 STR     R3, [SP,#0x14+var_14]
ROM:0139B0B0                 LDR     R0, =0x201
ROM:0139B0B4                 LDR     R1, =a__CodeGrm_single_linkli ; "../code/grm_single_linklist.c"
ROM:0139B0B8                 LDR     R2, =0x16D
ROM:0139B0BC                 LDR     R3, =aGrmS      ; "[GRM]: %s"
ROM:0139B0C0                 BL      Log2
ROM:0139B0C4                 CMP     R5, R6
ROM:0139B0C8                 BEQ     loc_139B0E4
ROM:0139B0CC                 LDR     R3, [R5,#8]
ROM:0139B0D0                 STR     R3, [R4]
ROM:0139B0D4                 LDR     R3, [R4]
ROM:0139B0D8                 CMP     R3, R6
ROM:0139B0DC                 LDRNE   R3, [R4]
ROM:0139B0E0                 LDRNE   R6, [R3]
ROM:0139B0E4
ROM:0139B0E4 loc_139B0E4                             ; CODE XREF: vptr_slist_last+34↑j
ROM:0139B0E4                 LDR     R3, =aExitingFunctionVptr_s_0 ; "Exiting function vptr_slist_last()"
ROM:0139B0E8                 STR     R3, [SP,#0x14+var_14]
ROM:0139B0EC                 LDR     R0, =0x201
ROM:0139B0F0                 LDR     R1, =a__CodeGrm_single_linkli ; "../code/grm_single_linklist.c"
ROM:0139B0F4                 LDR     R2, =0x17F
ROM:0139B0F8                 LDR     R3, =aGrmS      ; "[GRM]: %s"
ROM:0139B0FC                 BL      Log2
ROM:0139B100                 MOV     R0, R6
ROM:0139B104                 ADD     SP, SP, #4
ROM:0139B108                 LDMFD   SP!, {R4-R6,PC}
ROM:0139B108 ; End of function vptr_slist_last
ROM:0139B108
ROM:0139B108 ; ---------------------------------------------------------------------------
```

Find Cipher Code

RUB

- Yields **240kb** of DSP code
- **TI C55x** assembler
  - Code hard to understand
  - Needs some initial training
- **No** strings, symbols, whatsoever

```
ROM:19BA7          mov     AC0, T0
ROM:19BA9          mov     #0, AC0
ROM:19BAB          rptb    loc_19BE3
ROM:19BAE          add     *AR0+ << #16, AC0
ROM:19BB1          sftl    AC0, #-16, AC1
ROM:19BB4          xor     AC0 >> #14, AC1
ROM:19BB7          xor     AC0 >> #13, AC1
ROM:19BBA          xor     AC0 >> #10, AC1
ROM:19BBD          mov     AC1, AC2
ROM:19BBF          xor     AC0 >> #11, AC1
ROM:19BC2          xor     AC0 >> #15, AC2
ROM:19BC5          bfxpa   #5555h, AC1, T1
ROM:19BC9          bfxpa   #0AAAAh, AC2, AC3
ROM:19BCD          or      T1, AC3
ROM:19BCF          mov     AC3, *AR1+
```

- How do we find the cipher code **conveniently**?
- A5-GSM cipher relies heavily on **linear feedback shift registers** (LFSRs)
- Typically uses many XOR and shift operations …

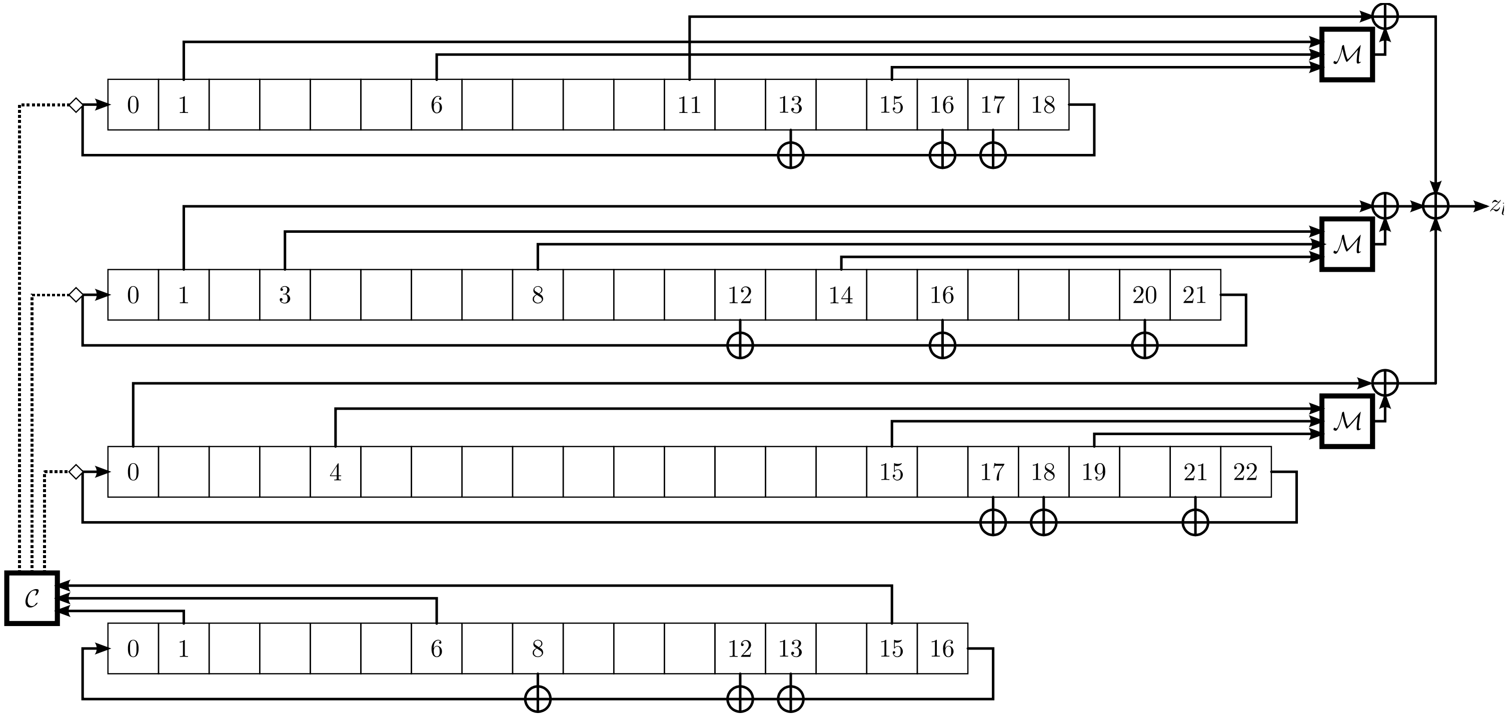
One identified function

```
sub_1CFC8:                                      ; CODE XREF: sub_1D0FC:loc_1D17C↓p
                                                ; sub_1D24C+8B↓p
                mov     dbl(*abs16(#reg1)), AC1
                sftl    AC1, #-1, AC2
                mov     dbl(*abs16(#reg1)), AC1
                sftl    AC1, #-3, AC3
                mov     dbl(*abs16(#reg1)), AC1
                xor     AC3, AC1
                bfxtr   #0FFFCh, AC1, AR1
                mov     dbl(*abs16(#reg1)), AC1
                and     #1, AR1, AC3
                xor     AC2, AC1
                and     #1, AC1, AC1
                xor     AC3, AC1
                xor     AC0, AC1
                sftl    AC1, #18, AC0
                xor     AC2, AC0
                mov     AC0, dbl(*abs16(#reg1))
                ret
; End of function sub_1CFC8
```

- 4 such functions exist
- Each function does **exactly** one LFSR operation
- Reverse engineering of the 4 functions reveals the cipher…

The Cipher !

Looks familiar…



- A5-GMR-1 is basically „A5/2-GSM“
  - Feedback polynomials changed
  - Position of output taps changed
  - Initialization process changed slightly

Cryptanalysis
Ciphertext-Only Attack

- Ciphertext-only attack is possible
  - Based on ideas and [Barkan/Biham/Keller 2003]
- Adapt attack to GMR-1
  - Guess parts of R1, R2 and R3 to reduce variables and equations (increases the number of guesses…)

Results

- **Attack on voice channel possible with 16 frames + 2^{21} guesses**
- Experimental set-up cf. next slides

Attack Set-up with Software-Defined Radio

RUB

Further Reading

RUB

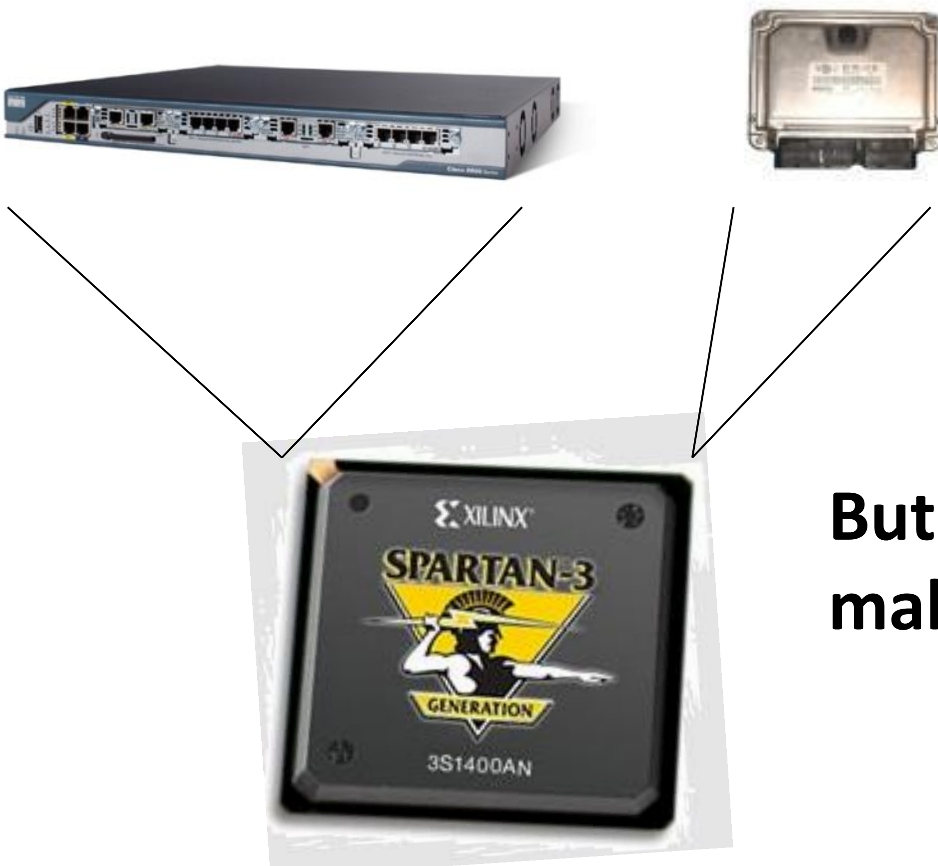
- Driessen, Hund, Willems, P, Holz: Don't Trust Satellite Phones: A Security Analysis of Two Satphone Standards.
  IEEE Symposium on Security and Privacy 2012

Agenda

RUB

- General Thoughts on Embedded Security
- Constructive: Bar Codes and SP Ciphers
- Destructive 1: Cell phones in the Desert
- **Destructive 2: Routers and AES**
- Auxiliary stuff

FPGAs = Reconfigurable Hardware

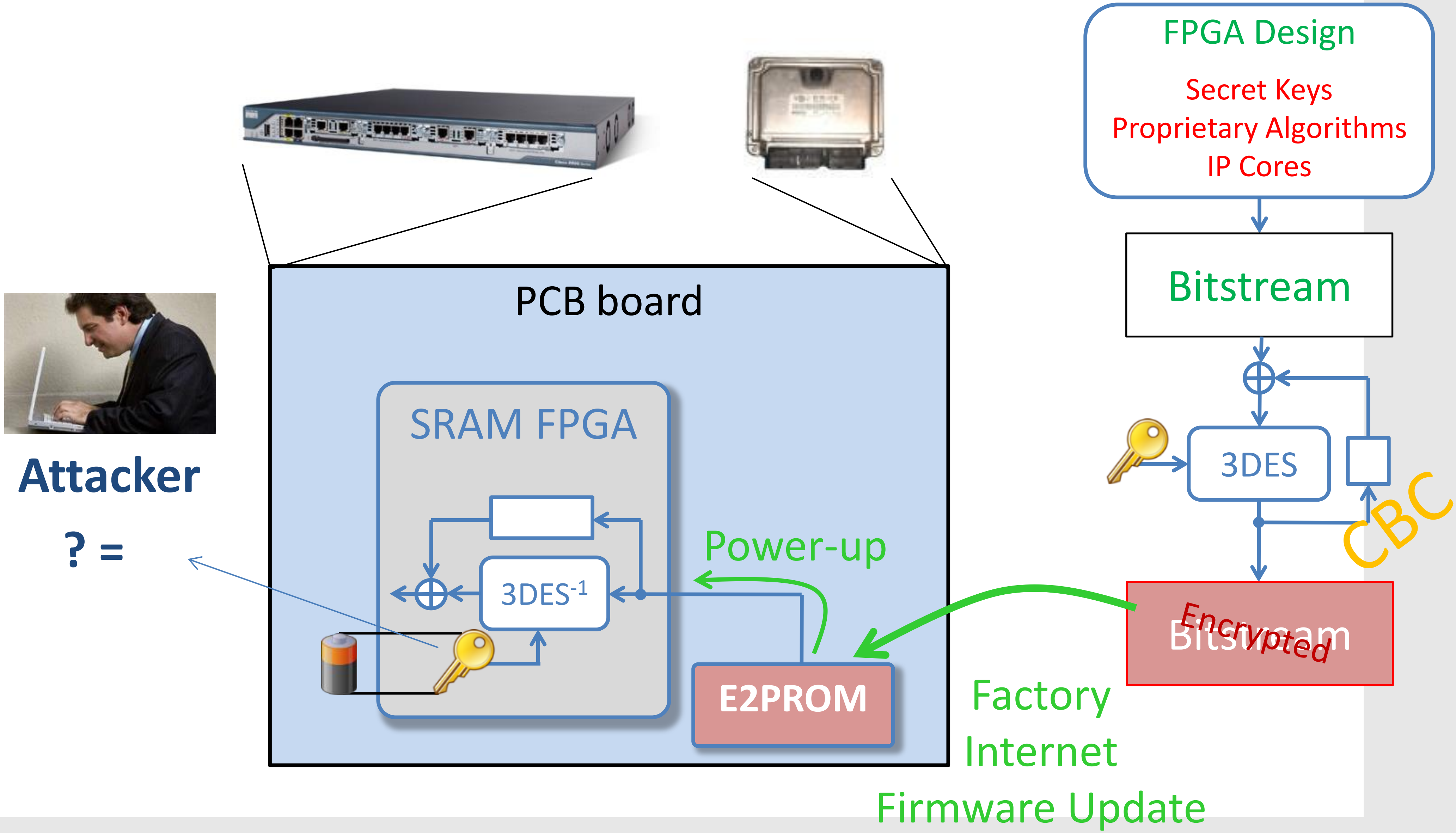


Widely used in

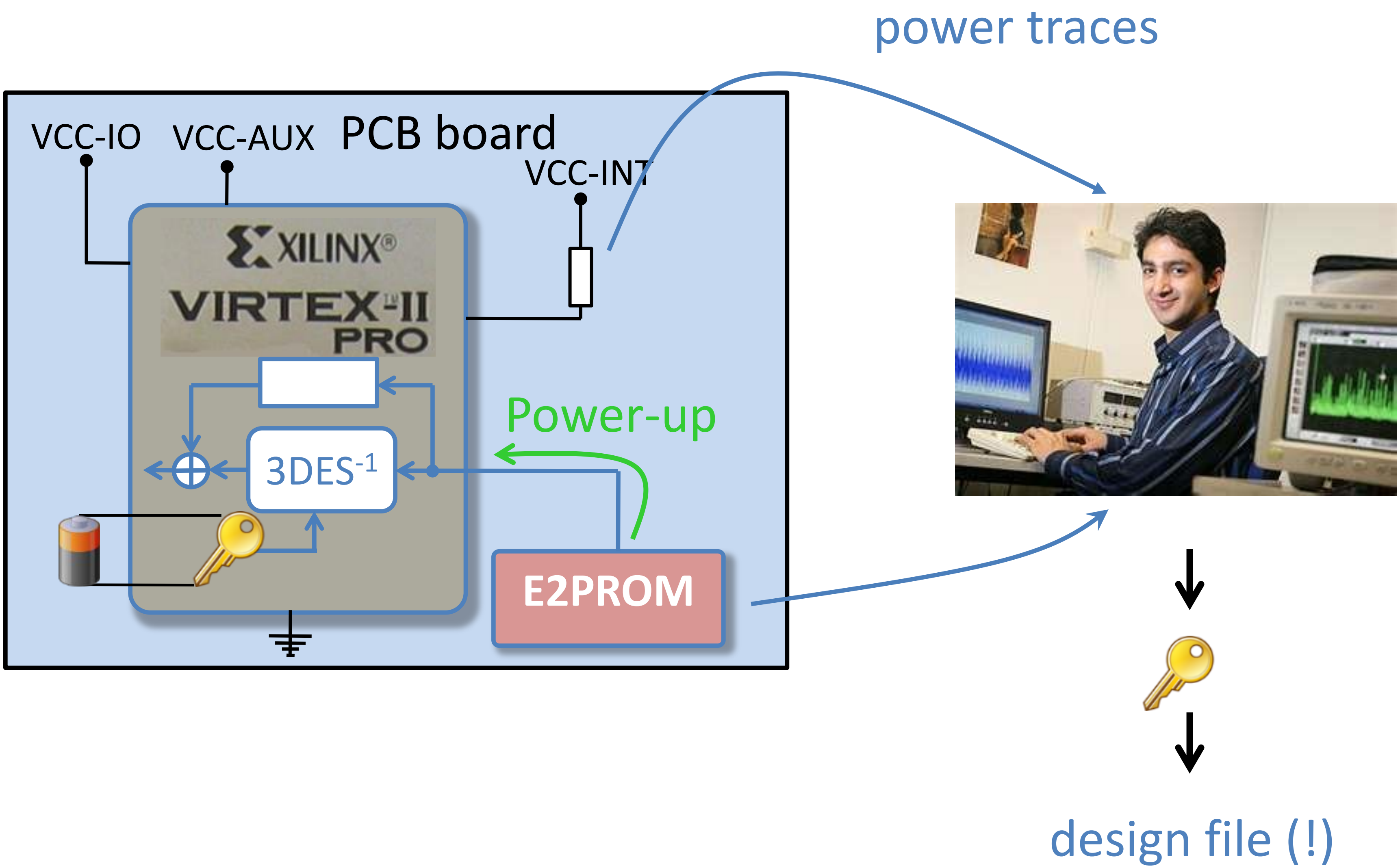
- routers
- consumer products
- automotive, machinery
- military

But: Copying the configuration files makes hardware counterfeiting easy!

Solution: Bitstream encryption



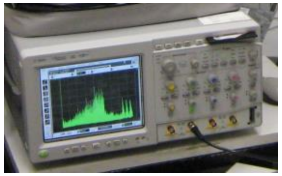
Let’s try side-channel analysis



Side-Channel Attacks (1-slide version)

Analyze cipher

- Find a suited predictable intermediate value in the cipher



Measurements

- Measure the power consumption

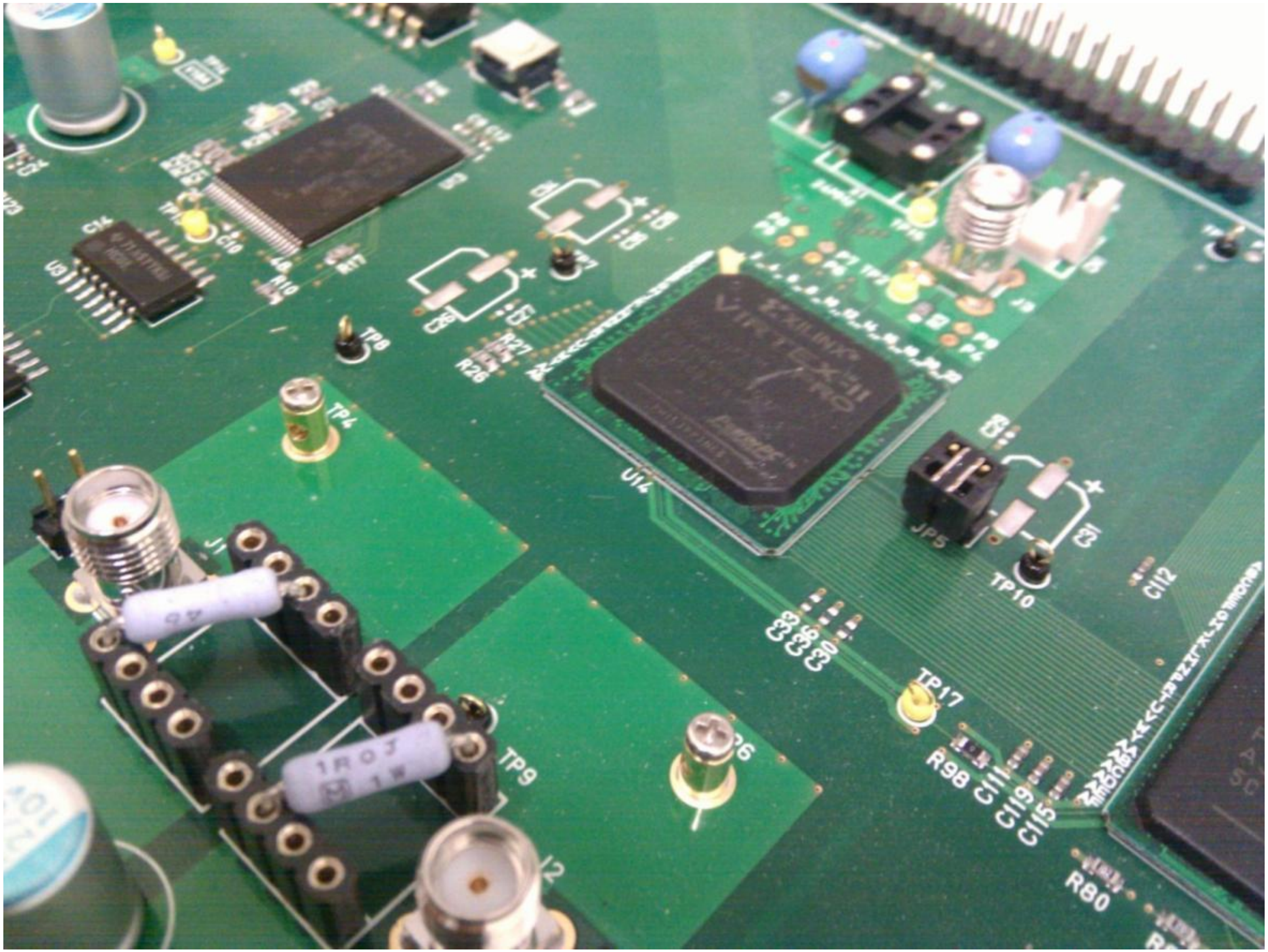
Post Processing

- Post-process acquired data

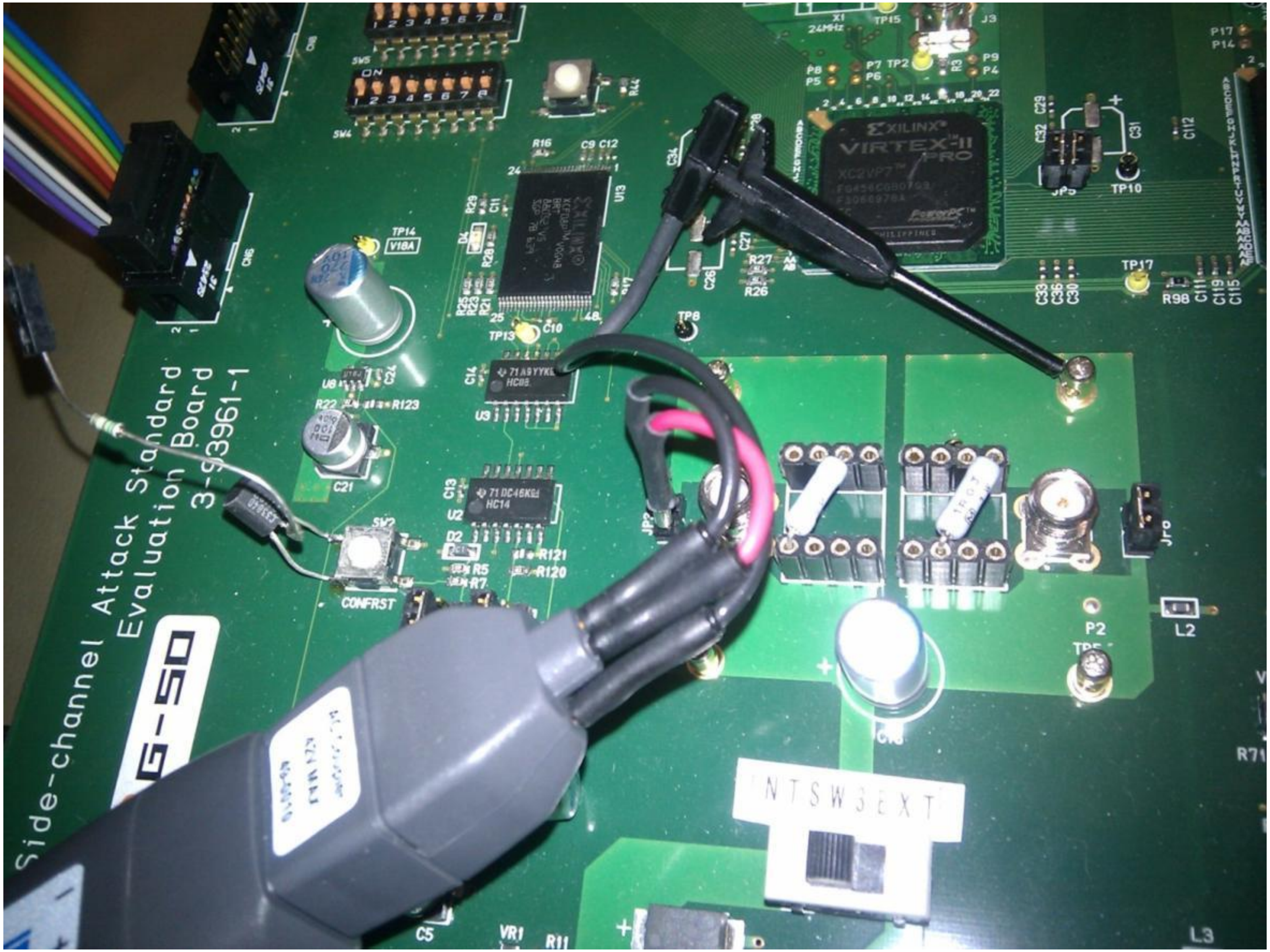
Key Recovery

- Perform the attack to recover the key

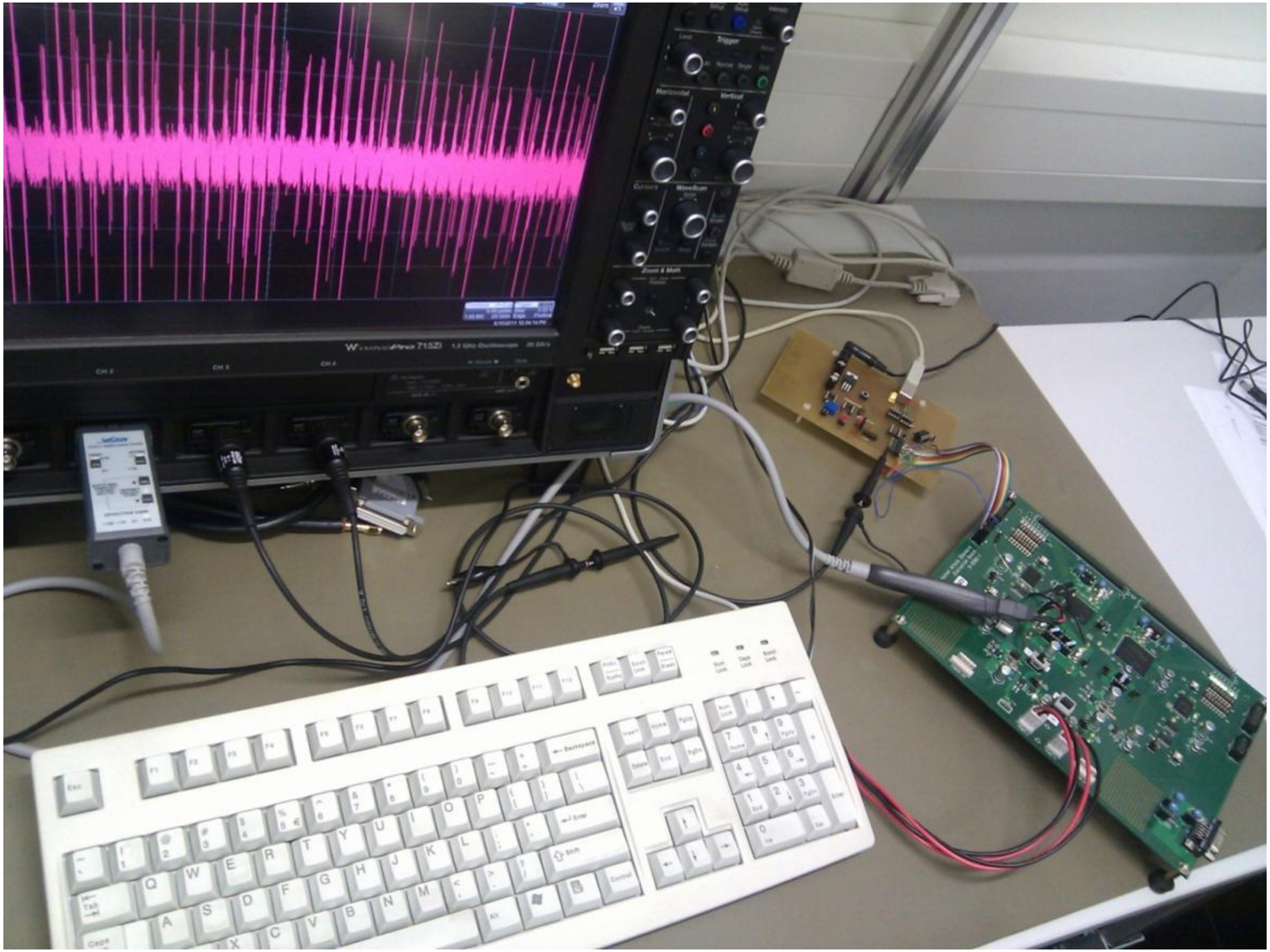
Our measurement set-up



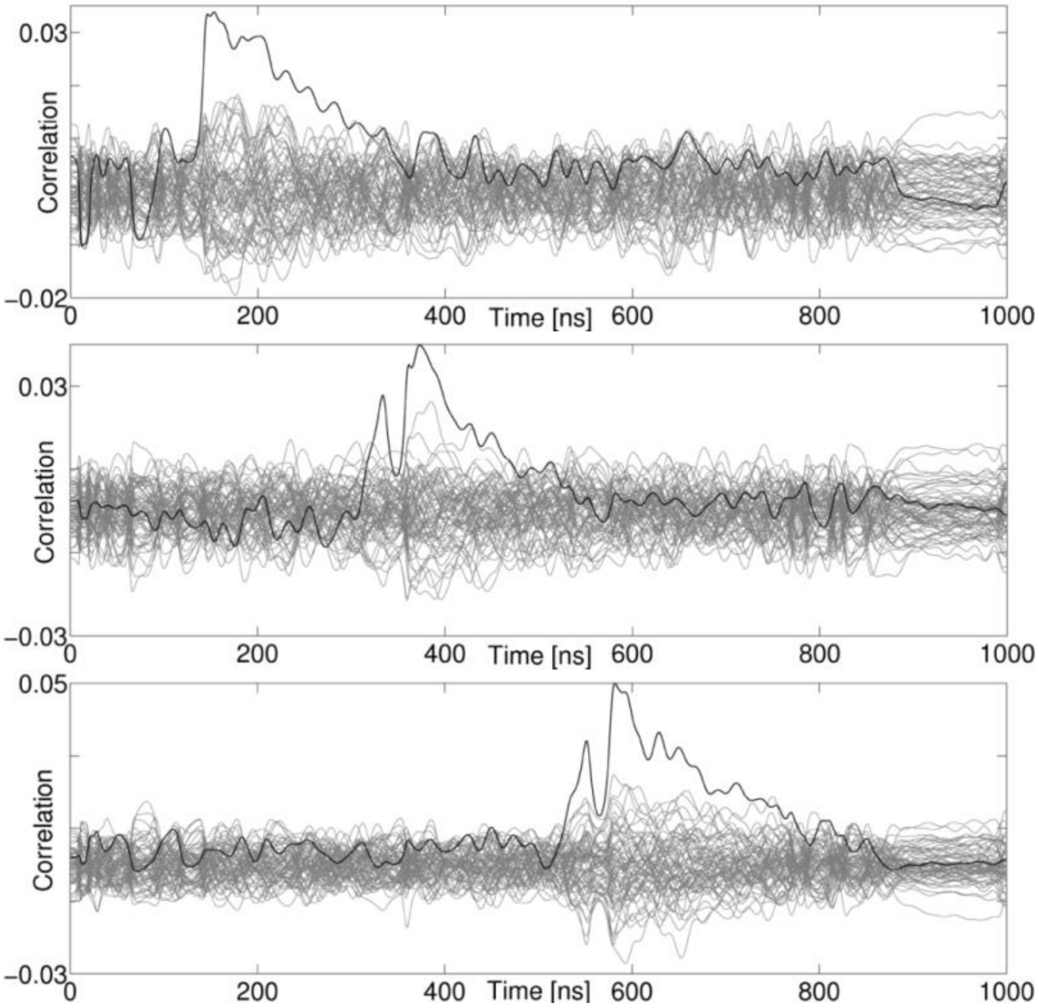
Our measurement set-up



Signal acquisition



… 6 months later



key of 1st DES

key of 2nd DES

key of 3rd DES

Long story made short: Decryption of “secret” designs is easy!

RUB

- Requires *single* power-up (≈ 50,000 traces)
- Complete 3DES key recovered with 2-3 min of computation
- Attack possible even though 3DES is only very small part of chip (< 1%)
- Attack requires some experience, but
    - cheap equipment
    - easy to repeat

Implications

- Cloning of product
- Reverse engineering of design internals
- Alterations of design (chip tuning)
- Trojan hardware (i.e., malicious hardware functions)
- …

Further Reading

RUB

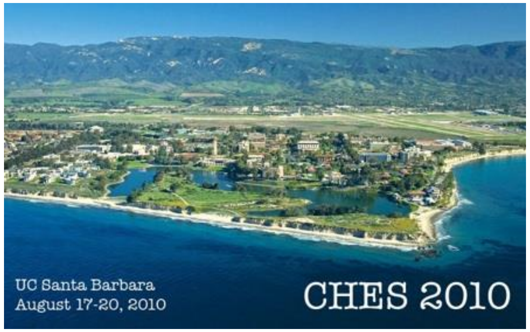
- Moradi, Barenghi, Kasper, P: On the vulnerability of FPGA bitstream encryption against power analysis attacks: extracting keys from Xilinx Virtex-II FPGAs.
  ACM CCS 2011
- Moradi, Oswald, P, Swierczynski: Side-channel attacks on the bitstream encryption mechanism of Altera Stratix II: facilitating black-box analysis using software reverse-engineering.
  FPGA 2013

Agenda

RUB

- General Thoughts on Embedded Security
- Constructive: Bar Codes and SP Ciphers
- Destructive 1: Cell phones in the Desert
- Destructive 2: Routers and AES
- **Auxiliary stuff**

Related Workshops



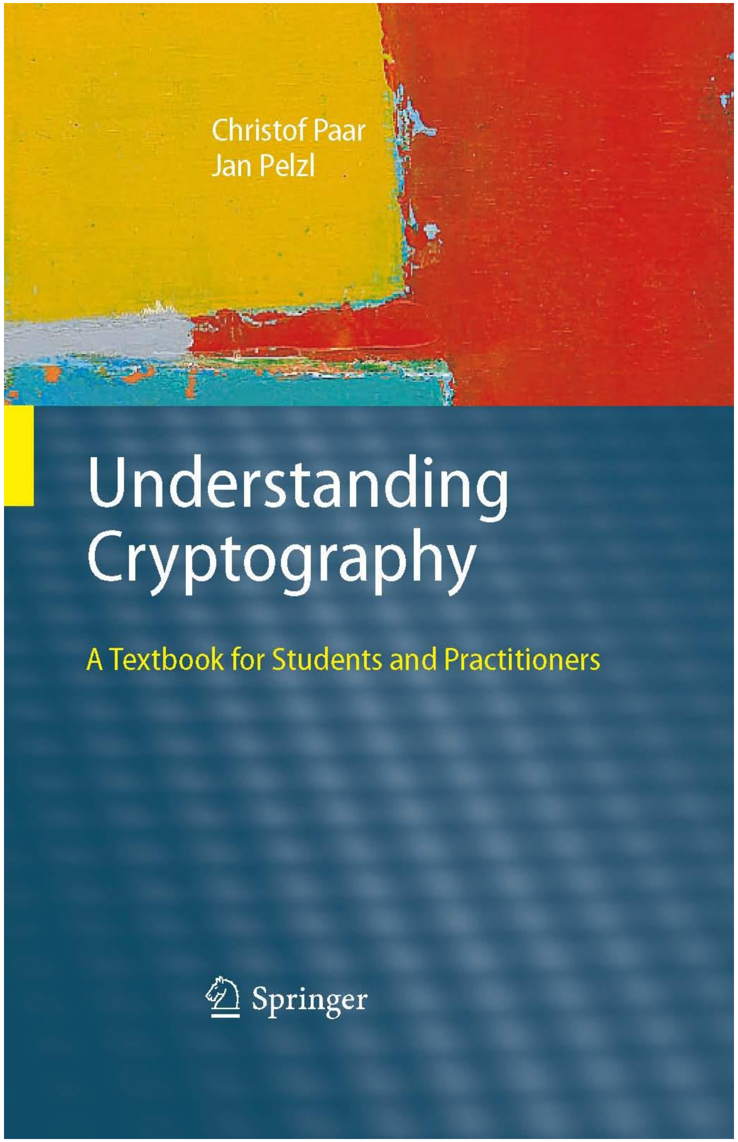
CHES – Cryptographic Hardware & Embedded Systems
August 2013, Santa Barbara, CA, USA

RFIDsec 2013
July 2013, Graz, Austria

RFIDsec

escar USA – Embedded Security in Cars
November, Frankfurt, Germany

yet another textbook on cryptography (but this one targets engineers)



www.crypto-textbook.com

- includes videos of 2-semester course (in English)
- complete set of slides
- many further resources