How to Attack the IoT with Hardware Trojans

RUB

Janet Lackey under CC license

CROSSING Conference

Darmstadt, May 16, 2017

Christof Paar

Ruhr Universität Bochum

hgi
Horst Görtz Institut für IT-Sicherheit

Acknowledgement

RUB

- Georg Becker

- Pawel Swierczynski

- Marc Fyrbiak

Agenda

RUB

- Introduction to Hardware Trojans
- Sub-Transistor ASIC Trojans
- FPGA Trojan
- Key extraction attack
- Auxiliary Stuff

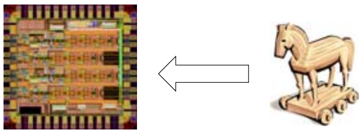
Agenda

RUB

- **Introduction to Hardware Trojans**
- Sub-Transistor ASIC Trojans
- FPGA Trojan
- Key extraction attack
- Auxiliary Stuff

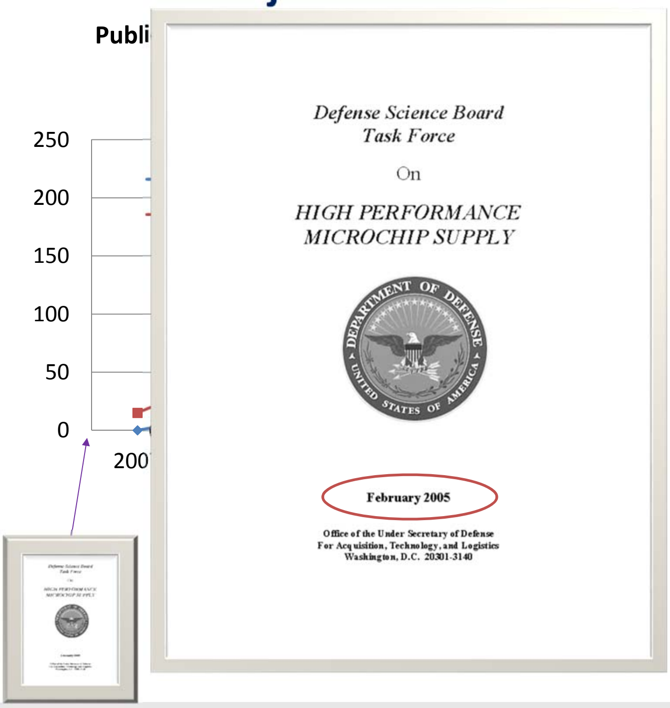
Hardware Trojans

Malicious change or addition to an IC that adds or remove functionality, or reduces reliability



Many rather unpleasant "applications"

Hardware Trojans & the Scientific Community



Defense Science Board
Task Force

On

HIGH PERFORMANCE
MICROCHIP SUPPLY

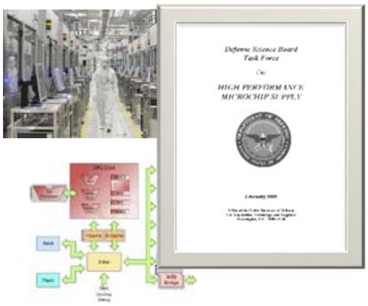
February 2005

Office of the Under Secretary of Defense
For Acquisition, Technology, and Logistics
Washington, D.C. 20301-3140

Trojan Injection & Adversaries Scenarios

RUB

DoD scenario 2005



- **Manufacturing**
  Malicious factory, esp. off-shore (foreign Government)
- **Design Manipulation**
  - 3rd party IP-cores
  - malicious employee

not-so-unlikely 2013

- **During shipment**
  cf. NSA's *interdiction*
- **Built-in**
  backdoors etc.

Where are we with "real" HW Trojans?

RUB

- No true hardware Trojan observed in the wild

- All examples from academia

- Vast majority of publications focus on detection

Our Thoughts ca. 2012

RUB

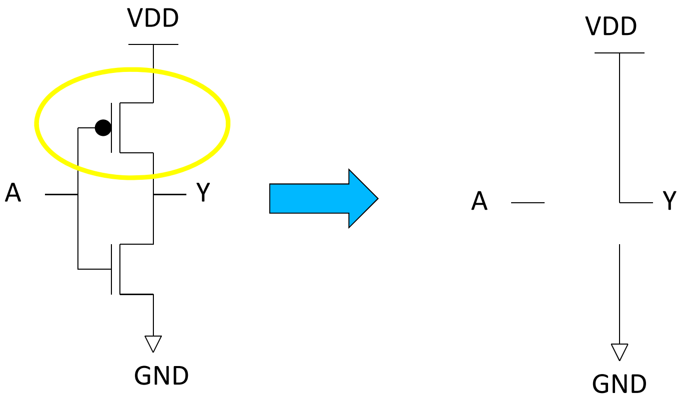
1. *Designing* Trojan could be fun too
2. Especially those that go *undetected*

Simple Example: Inverter Trojan

RUB

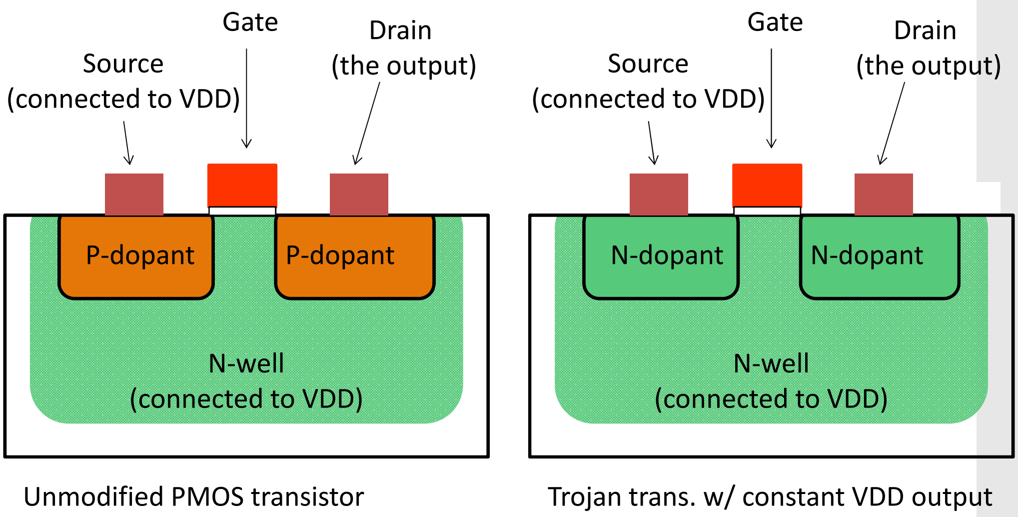
Let's modify an inverter so that it always outputs "1" (VDD) **without visible changes**.

A	Y
0	1
1	0



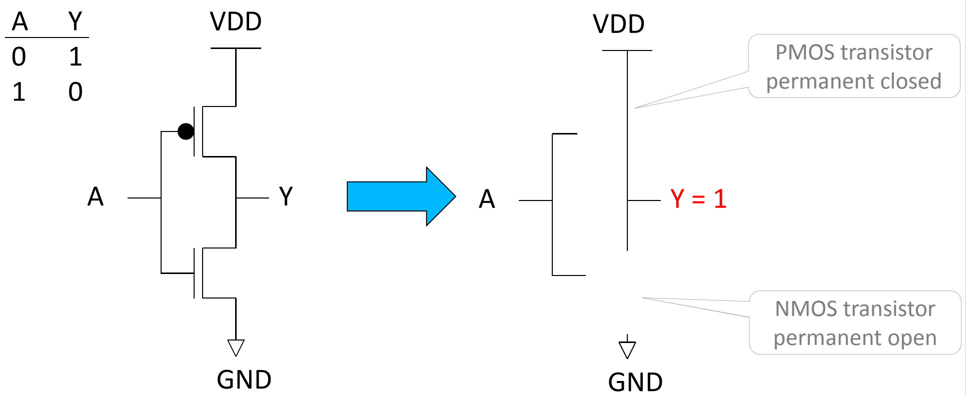
PMOS Transistor Trojan

RUB



"Always One" Trojan Inverter

RUB



Q1: Can the manipulation be detected?

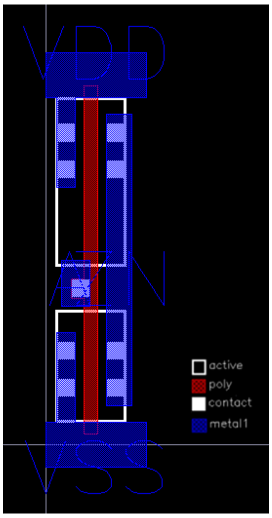
Q2: How to build a useful Trojan from here?

Detection: layout view of Trojan inverter

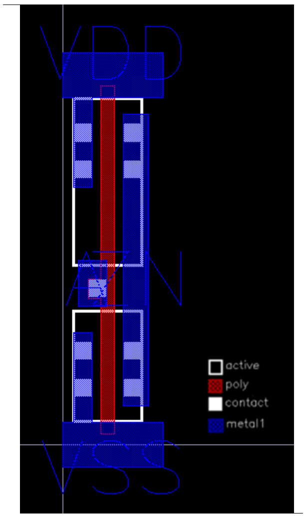
RUB

Which one has the Trojan?

Original Inverter



"Always One" Trojan



Unchanged:

- All metal layers
- Polysilicon layer
- Active area
- Wells

⇒ Dopant changes (very ?) difficult to detect using optical inspection!

"Small" remaining question

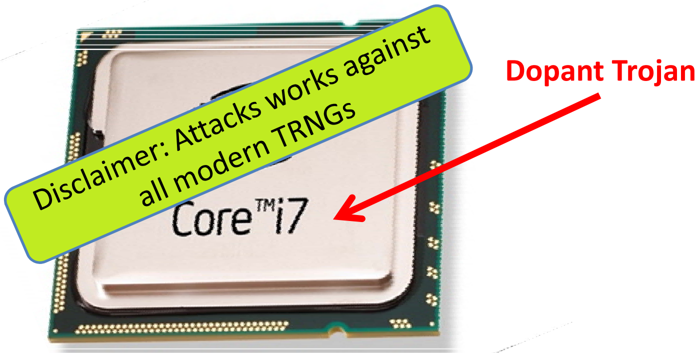
RUB

- Unfortunately, circuits will not function correctly with this simple stuck-at fault …
- … functional testing (after manufacturing) will detect fault right away

Q2: Can we build a **meaningful** Trojan using dopant modifications that passes functional testing?

A Real-World True Random Number Generator

RUB

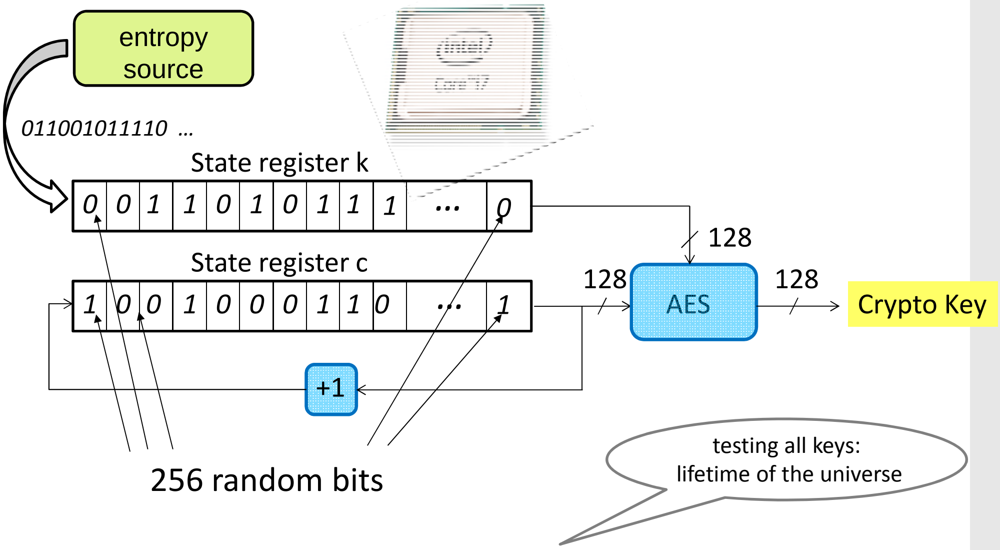


… random numbers generate cryptographic keys for

- secure web browsing
- email encryption
- document certification
- …

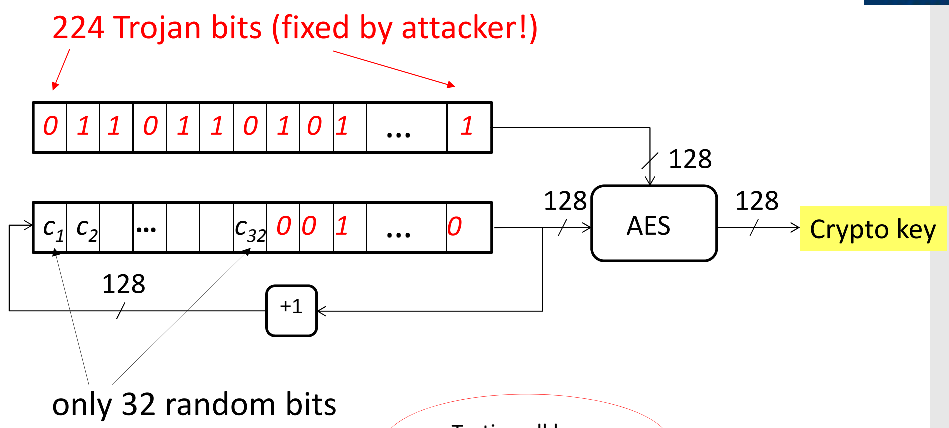
Inside the Random Number Generator

RUB



- 1,000,000,000,000,000,000,000,000,000,000,000,000,000 possible crypto keys

Trojan Random Number Generator

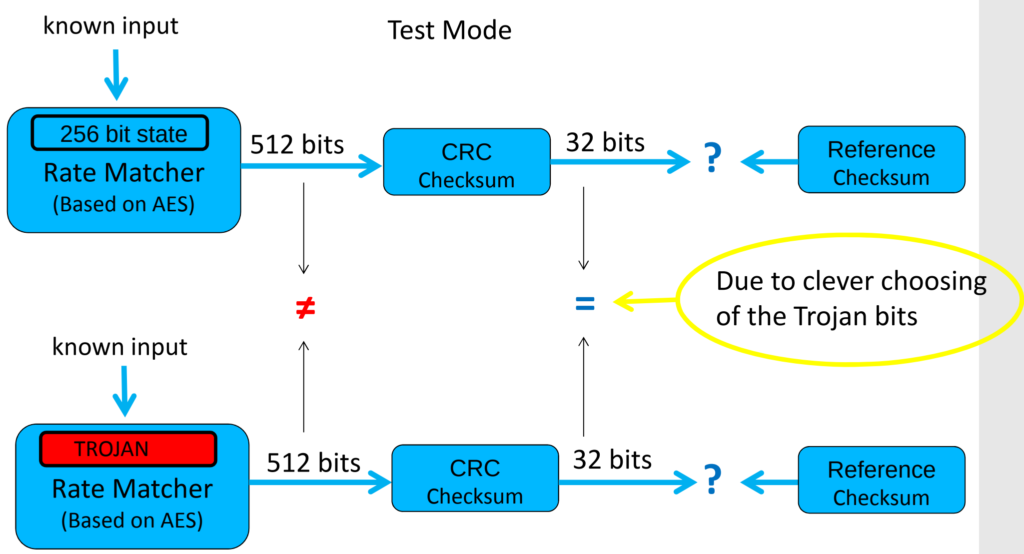


Testing all keys: few seconds

- 1,000,000,000,000,000,000,000,000,000,000,000,000,000 possible crypto keys

… but circuit would still be tested as “faulty” during manufacturing…

Detection prevention through built-in self test



Conclusion

RUB

Slashdot SPIEGEL ONLINE  ars technica COMPUTERWORLD

- Meaningful hardware Trojans are possible without extra logic
- Many detection techniques don't guarantee a Trojan free design!
- Built-in self tests can be dangerous
- More details:
  Becker, Regazzoni, P, Burleson, *Stealthy Dopant-Level Hardware Trojans.* CHES 2013

... but the scientific community functions as it is supposed to do:

- Trojan detection is possible w/ scanning electron microscope
  Sugawara et al., *Reversing Stealthy Dopant-Level Circuits. CHES 2014*

Agenda

RUB

- Introduction to Hardware Trojans
- Sub-Transistor ASIC Trojans
- **FPGA Trojan**
- Key extraction attack
- Auxiliary Stuff

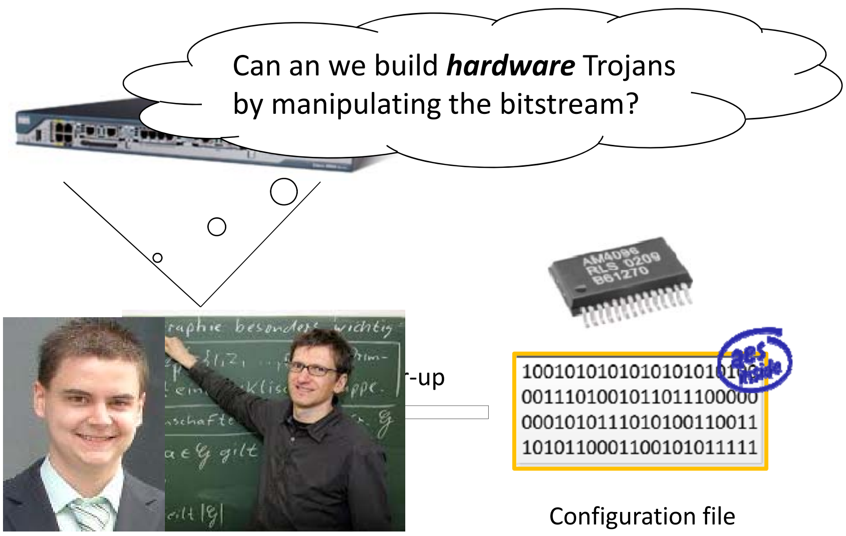
FPGAs = Reconfigurable Hardware … are widely used

RUB

world market:
≈ 5b devices

Configuration during power-up

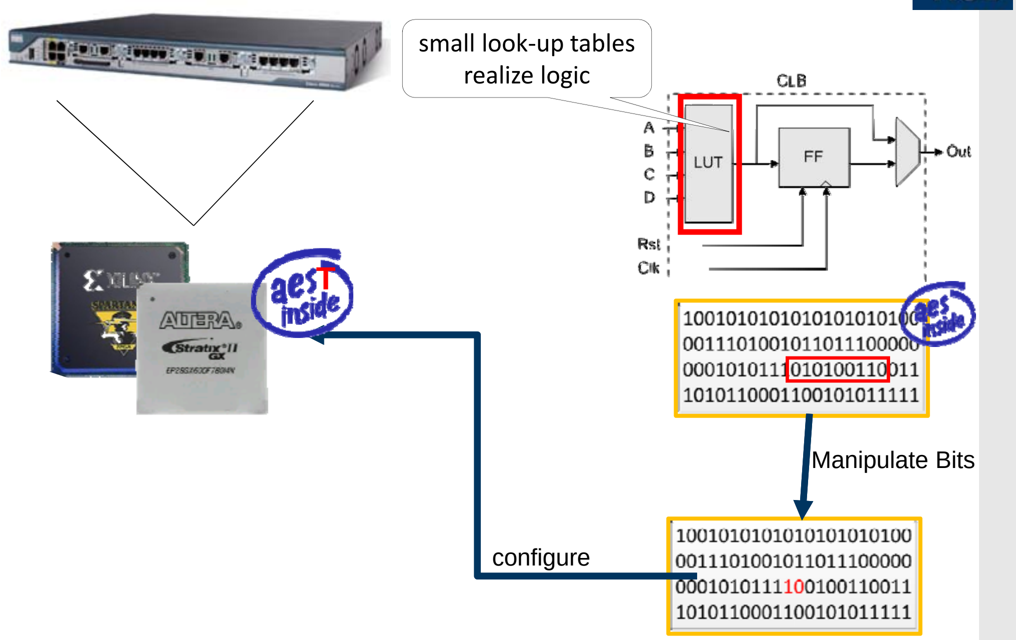
RUB



Configuration file
“bitstream”

Principle of FPGA-based Trojans

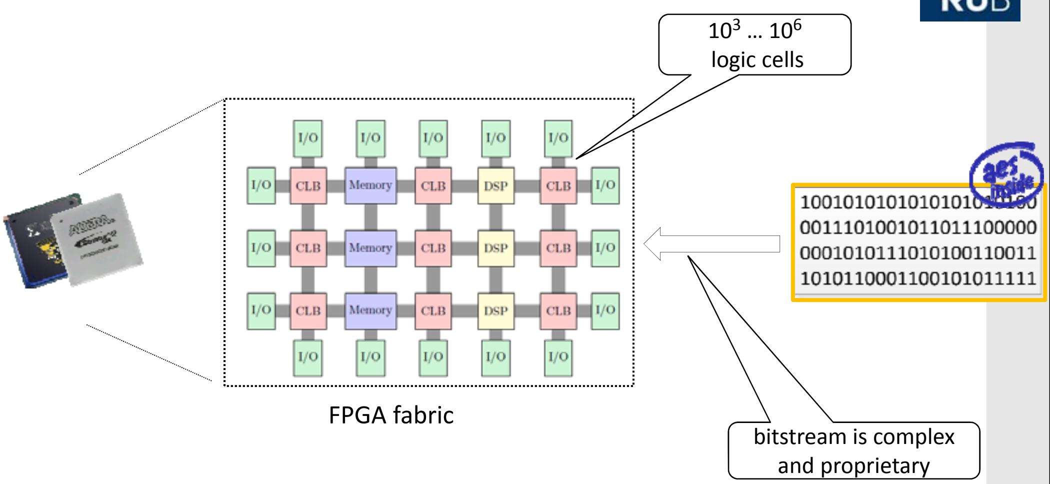
RUB



Source Graphics: SimpleIcon, Xilinx

The Mechanics of FPGAs

RUB

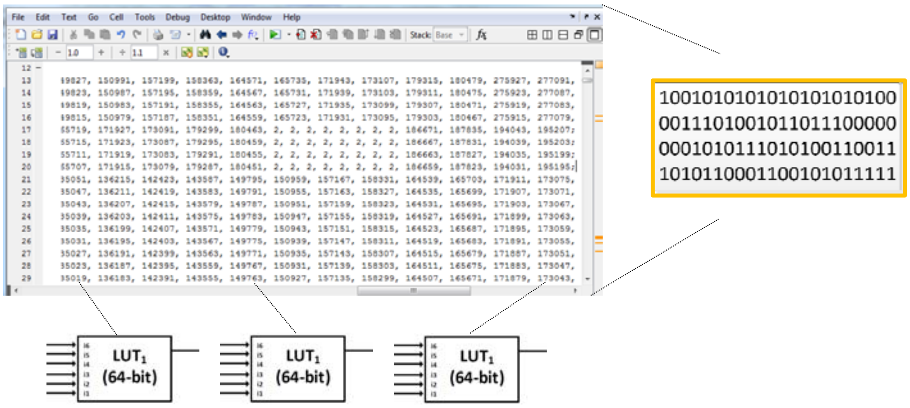


Two challenges

1. find AES in unknown design
2. meaningful manipulation

Finding AES:
Luckily, crypto has very specific components

RUB



- S-boxes are realized as 6x1 look-up tables (LUTs)
- LUT locations can be found in bitstream
- S-box contents is very specific (luckily)

AES detection in practice

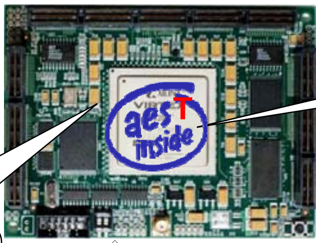
RUB

8 different real-world AES implementations

Impl.	Architecture	AES	LUTs with S-box logic	S-boxes in memory	Detection
#1	Round-based	128	$(16{+}4)\cdot 32 = 640$	no	100 %
#2	$\frac{1}{4}$ Round	128	0	yes	100 %
#3	$\frac{1}{4}$ Round	192	0	yes	100 %
#4	$\frac{1}{4}$ Round	256	0	yes	100 %
#5	Round-based	128	$(0{+}4)\cdot 32 = 128$	yes	100 %
#6	Round-based	128	0	yes	100 %
#7	Round-based	128	0	yes	100 %
#8	Round-based	128	$(16{+}4)\cdot 32 = 640$	no	100 %

TABLE IV: Overview of evaluated AES implementations

Algorithm substitution attack and its implications



1. Inject **weak S-boxes** in bitstream

2. **Trojan AES** is configured

cute work … but not interoperable with regular AES

PT *CT* = AEST (*k*, *PT*)

"Useful" attacks are still possible!

1. **Storage encryption – Plaintext recovery**
   - Attacker can recover plaintext without access to *k*

2. **Temporary device access – Key extraction**
   - switch S-box and recover *k* from *CT*
   - configure orginal S-box

Conclusion

- New attack vector against FPGAs!
- Reconfigurability allows "hardware" Trojans designed in the lab
- Bitstream protection is crucial!
  (but not easy, cf. our work at CCS 2011 & FPGA 2013)
- Details at:
  Swierczynski, Fyrbiak, Koppe, P, *FPGA Trojans through Detecting and Weakening of Cryptographic Primitives*. IEEE TCAD 2015.

Agenda

RUB

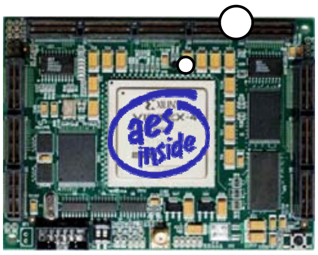
- Introduction to Hardware Trojans
- Sub-Transistor ASIC Trojans
- FPGA Trojan
- **Key extraction attack**
- Auxiliary Stuff

What else can we do with bitstreams?

RUB

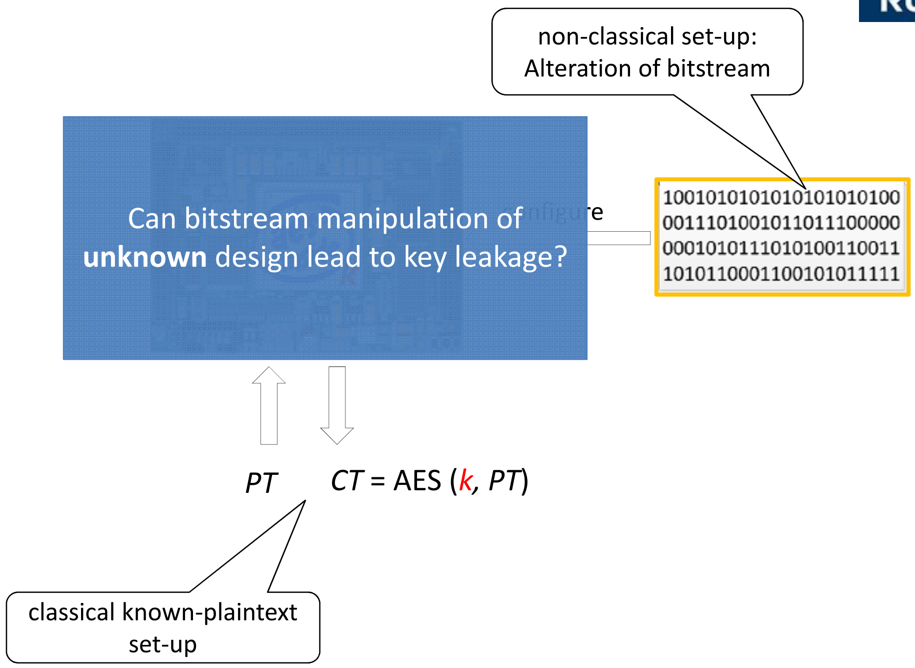
So, bitstream manipulation allows Trojan insertion ...

Hmm, are their other/simpler ways to **extract keys** through bitstreams?



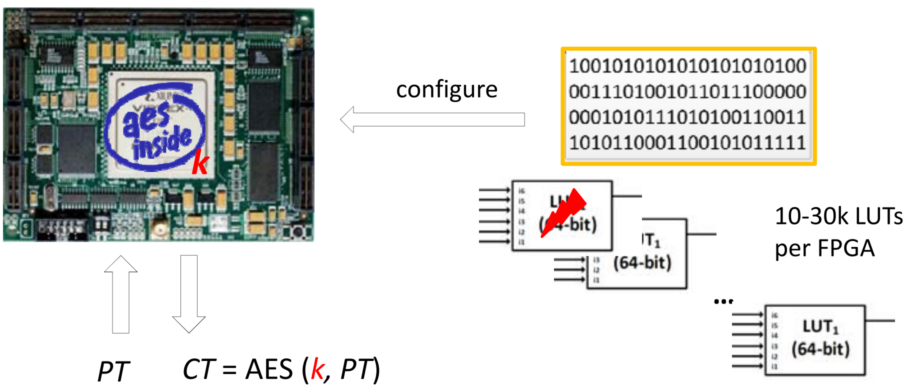
Set-Up

RUB



Bitstream Fault Injections (BiFI)

RUB

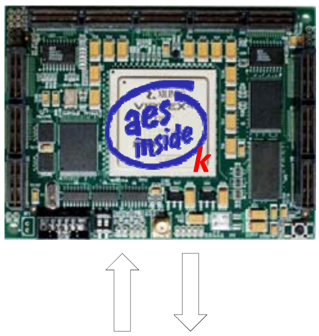


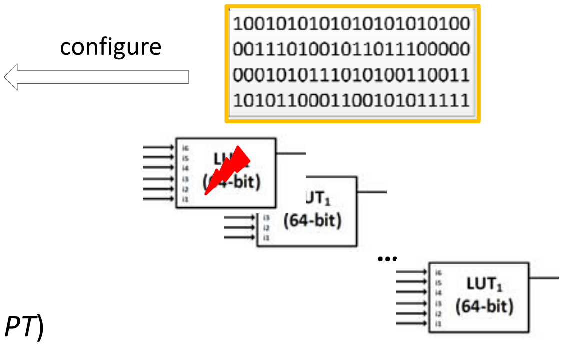
(surprising) attack strategy

1. manipulate 1st LUT table (e.g., all-zero)
2. configure FPGA
3. send PT
4. check: Does CT contain k?
   if not: GOTO 1 and manipulate next LUT

How exactly does the key leak ??

RUB





PT   *CT* = AES (*k*, *PT*)

Many leakage hypotheses

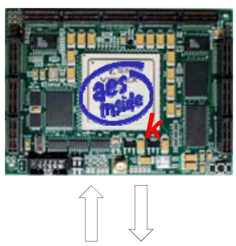
- CT = roundkey
- CT = inverted roundkey
- CT = PT xor roundkey
- …

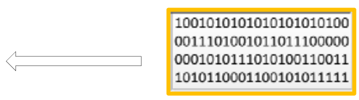
Many LUT manipulations possible

- all-zero
- all-one
- invert
- upper half of LUT all-zero
- …

Results for Bitstream Fault Injections (BiFI)

RUB





Real world attack

- 16 unknown AES designs (Internet)
- 16 different manipulation rules
- ≈ 20k LUTs
- 3.3 sec for configuring and checking one alterations

Results

- successful key extraction for **every** design!
- on average ≈ 2000 configurations (≈ 2h)
- works even for encrypted bitstream (w/o MAC)

Conclusion

RUB

- Bitstream Fault Injections (BiFI) is a new family of fault attacks
- Malleability of bitstream is major weakness for FPGAs!
- Are there more bitstream-based attacks ?
- Details at:
  Swierczynski, Becker, Moradi, P: Bitstream Fault Injections (BiFI) – Automated Fault Attacks against SRAM-based FPGAs. IEEE Transactions on Computers, to appear.

Agenda

RUB

- Introduction to Hardware Trojans
- Sub-Transistor ASIC Trojans
- FPGA Trojan
- Key extraction attack
- **Auxiliary Stuff**

Related Workshops

RUB

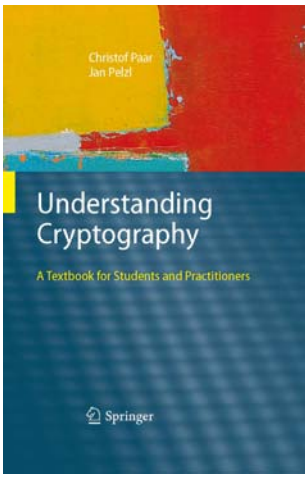
CHES – Cryptographic Hardware & Embedded Systems
25.-28. September 2017, Taiwan

escarUSA – Embedded Security in Cars
Ann Arbor, June 2017

escarEurope – Embedded Security in Cars
Berlin, November 2017

Easy-to-understand book for applied cryptography

RUB



YouTube

Introduction to Cryptography by Christof Paar

24 video lectures

Thank you very much for your attention!

Christof Paar

Ruhr-Universität Bochum