The First Thorough Side-Channel Hardware Trojan

5. Dec. 2017

Maik Ender, Saman Ghandali, Amir Moradi, Christof Paar

Ruhr University Bochum, Germany
University of Massachusetts Amherst, USA
Hardware Trojan

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

Many rather unpleasant “applications”
Hardware Trojan

- Many potential attack vectors
  - Malicious foundry/company
  - Malicious employee
  - 3rd party IP cores
  - Government request
  - Hacker attacks
  - During shipment
  - ...
Story

- How Trojans work?
  - Small changes at certain points can break/weaken crypto
    - Changes in functionality of the circuit
      - e.g., sending the key out
    - Changes in side-channel leakage of the circuit
      - leaking the sensitive information through DPA

- How to detect?
  - Comparison with a golden chip!
  - Functional testing
  - SCA evaluation
  - Netlist analysis
    - shown at CHES 2016* that if logics appear correct, Trojan can still exist

* Ghandali, Becker, Holcomb, Paar. A Design Methodology for Stealthy Parametric Trojans and Its Application to Bug Attacks
Story

- **How Trojans work?**
  - Small changes at certain points can break/weaken crypto
    - Changes in functionality of the circuit
      - e.g., sending the key out
    - Changes in side-channel leakage of the circuit
      - leaking the sensitive information through DPA

- **How to detect?**
  - Comparison with a golden chip!
  - Functional testing
  - SCA evaluation
  - Netlist analysis
    - shown at CHES 2016* that if logics appear correct, Trojan can still exist

* Ghandali, Becker, Holcomb, Paar. A Design Methodology for Stealthy Parametric Trojans and Its Application to Bug Attacks

---

**this work:**
How to design a circuit which passes the SCA evaluation, but is an SCA-Trojan infected
Are we helping criminals?

- Trojan detection and design are closely related
  - effective detection mechanisms
  - understanding of how hardware Trojans can be built

- We show how particularly SCA parametric Trojans can be introduced to an SCA-secure circuit
  - parametric: becomes active under a certain condition
    - in this work: high clock frequency
Why ours the first thorough

Previous SCA Trojans:
- a change on the PRNG, or
- a transistor-level manipulation (dopant polarity), or
- a separate circuit to leak the key

- None of them can pass SCA evaluation
  - e.g., a leakage detection test
    - popular as the first step in SCA evaluation

Our goal: the device can pass the SCA evaluations
- becomes SCA vulnerable at a high clock frequency
Masking in Hardware

- After several heuristic and ad-hoc solutions
  - Threshold Implementation (TI): security against 1st order attacks even with glitchy circuit
  - Boolean masking + multi-party computation
  - linear functions simple, $l(x^1 \oplus x^2 \ldots x^n) = l(x^1) \oplus l(x^2) \ldots l(x^n)$
  - challenge:
    non-linear functions (Sbox) over Boolean masking
TI Requirements

- correctness
  - $y = S(x), \ y^1 \oplus y^2 \ldots \ y^n = S(x^1 \oplus x^2 \ldots \ x^n)$

- non-completeness
  - each $f$ should be independent of one share

- uniformity
  - if $(x^1, x^2, \ldots, x^n)$ is a uniform sharing of a certain $x$
    - $(y^1, y^2, \ldots, y^n)$ should be also a uniform sharing of $y = S(x)$

- not fulfilling either non-completeness or uniformity
  - no guarantee on first-order security
TI Requirements

- **correctness**
  - \( y = S(x), \ y^1 \oplus y^2 \ldots \ y^n = S(x^1 \oplus x^2 \ldots x^n) \)

- **non-completeness**
  - each \( f \) should be independent of one share

- **uniformity**
  - if \( (x^1, x^2, ..., x^n) \) is a uniform sharing of a certain \( x \)
    - \( (y^1, y^2, ..., y^n) \) should be also a uniform sharing of \( y = S(x) \)

- not fulfilling either non-completeness or uniformity
  - no guarantee on first-order security
Direct Sharing

example: \( x = (a, b, c, d), \ y = (e, f, g, h), \ S_1(a, b, c, d) = e \)
Direct Sharing

example: \( x = (a, b, c, d), \quad y = (e, f, g, h), \quad S_1(a, b, c, d) = e \)

\[
e = a \oplus bc \oplus d = a^1 \oplus a^2 \oplus a^3 \oplus (b^1 \oplus b^2 \oplus b^3) (c^1 \oplus c^2 \oplus c^3) \oplus d^1 \oplus d^2 \oplus d^3
\]
**Direct Sharing**

example: \(x = (a, b, c, d), \ y = (e, f, g, h), \ S_1(a, b, c, d) = e\)

\[
e = a \oplus b c \oplus d = a^1 \oplus a^2 \oplus a^3 \oplus (b^1 \oplus b^2 \oplus b^3) (c^1 \oplus c^2 \oplus c^3) \oplus d^1 \oplus d^2 \oplus d^3
\]

\[
e = a^1 \oplus a^2 \oplus a^3 \oplus b^1 c^1 \oplus b^1 c^2 \oplus b^1 c^3 \oplus b^2 c^1 \oplus b^2 c^2 \oplus b^2 c^3 \oplus b^3 c^1 \oplus b^3 c^2 \oplus b^3 c^3 \oplus d^1 \oplus d^2 \oplus d^3
\]
Direct Sharing

example: \( x=(a, b, c, d), \ y=(e, f, g, h), \ S_1(a, b, c, d) = e \)

\[
e = a \oplus bc \oplus d = a^1 \oplus a^2 \oplus a^3 \oplus (b^1 \oplus b^2 \oplus b^3)(c^1 \oplus c^2 \oplus c^3) \oplus d^1 \oplus d^2 \oplus d^3
\]

\[
e = a^1 \oplus a^2 \oplus a^3 \oplus b^1 c^1 \oplus b^1 c^2 \oplus b^1 c^3 \oplus b^2 c^1 \oplus b^2 c^2 \oplus b^2 c^3 \oplus b^3 c^1 \oplus b^3 c^2 \oplus b^3 c^3 \oplus d^1 \oplus d^2 \oplus d^3
\]

direct sharing:

\[
e^1 = b^2 c^3 \oplus b^3 c^2 \oplus a^2 \oplus d^2 \oplus b^2 c^2
\]

\[
e^2 = b^3 c^1 \oplus b^1 c^3 \oplus a^3 \oplus d^3 \oplus b^3 c^3
\]

\[
e^3 = b^1 c^2 \oplus b^2 c^1 \oplus a^1 \oplus d^1 \oplus b^1 c^1
\]
Direct Sharing

example: $x=(a, b, c, d), y=(e, f, g, h), S_1(a, b, c, d) = e$

$$e = a \oplus bc \oplus d = a^1 \oplus a^2 \oplus a^3 \oplus (b^1 \oplus b^2 \oplus b^3)(c^1 \oplus c^2 \oplus c^3) \oplus d^1 \oplus d^2 \oplus d^3$$

$$e = a^1 \oplus a^2 \oplus a^3 \oplus b^1c^1 \oplus b^1c^2 \oplus b^1c^3 \oplus b^2c^1 \oplus b^2c^2 \oplus b^2c^3 \oplus b^3c^1 \oplus b^3c^2 \oplus b^3c^3 \oplus d^1 \oplus d^2 \oplus d^3$$

direct sharing:

$$e^1 = b^2c^3 \oplus b^3c^2 \oplus a^2 \oplus d^2 \oplus b^2c^2$$

$$e^2 = b^3c^1 \oplus b^1c^3 \oplus a^3 \oplus d^3 \oplus b^3c^3$$

$$e^3 = b^1c^2 \oplus b^2c^1 \oplus a^1 \oplus d^1 \oplus b^1c^1$$

are clear where to go (which $e^i$)
Direct Sharing

example: \[ x = (a, b, c, d), \quad y = (e, f, g, h), \quad S_1(a, b, c, d) = e \]

\[ e = a \oplus bc \oplus d = a^1 \oplus a^2 \oplus a^3 \oplus (b^1 \oplus b^2 \oplus b^3)(c^1 \oplus c^2 \oplus c^3) \oplus d^1 \oplus d^2 \oplus d^3 \]

\[ e = a^1 \oplus a^2 \oplus a^3 \oplus b^1 c^1 \oplus b^1 c^2 \oplus b^1 c^3 \oplus b^2 c^1 \oplus b^2 c^2 \oplus b^2 c^3 \oplus b^3 c^1 \oplus b^3 c^2 \oplus b^3 c^3 \oplus d^1 \oplus d^2 \oplus d^3 \]

can be arbitrarily distributed among two functions \( e^i \) affecting uniformity

direct sharing:

\[ e^1 = b^2 c^3 \oplus b^3 c^2 \oplus a^2 \oplus d^2 \oplus b^2 c^2 \]
\[ e^2 = b^3 c^1 \oplus b^1 c^3 \oplus a^3 \oplus d^3 \oplus b^3 c^3 \]
\[ e^3 = b^1 c^2 \oplus b^2 c^1 \oplus a^1 \oplus d^1 \oplus b^1 c^1 \]

are clear where to go (which \( e^i \))
Direct Sharing

example: $x = (a, b, c, d), \ y = (e, f, g, h), \ S_1(a, b, c, d) = e$

$e = a \oplus bc \oplus d = a^1 \oplus a^2 \oplus a^3 \oplus (b^1 \oplus b^2 \oplus b^3)(c^1 \oplus c^2 \oplus c^3) \oplus d^1 \oplus d^2 \oplus d^3$

$e = a^1 \oplus a^2 \oplus a^3 \oplus b^1 c^1 \oplus b^1 c^2 \oplus b^1 c^3 \oplus b^2 c^1 \oplus b^2 c^2 \oplus b^2 c^3 \oplus b^3 c^1 \oplus b^3 c^2 \oplus b^3 c^3 \oplus d^1 \oplus d^2 \oplus d^3$

can be arbitrarily distributed among two functions $e^i$ affecting uniformity

direct sharing:

$e^1 = b^2 c^3 \oplus b^3 c^2 \oplus a^2 \oplus d^2 \oplus b^2 c^2 \oplus b^2$

$e^2 = b^3 c^1 \oplus b^1 c^3 \oplus a^3 \oplus d^3 \oplus b^3 c^3$

$e^3 = b^1 c^2 \oplus b^2 c^1 \oplus a^1 \oplus d^1 \oplus b^1 c^1 \oplus b^2$

are clear where to go (which $e^i$)

correction term, no effect on functionality, but on uniformity
Direct Sharing

example: \( x=(a, b, c, d), \ y=(e, f, g, h), \ S_1(a, b, c, d) = e \)

\[
e = a \oplus b c \oplus d = a^1 \oplus a^2 \oplus a^3 \oplus (b^1 \oplus b^2 \oplus b^3)(c^1 \oplus c^2 \oplus c^3) \oplus d^1 \oplus d^2 \oplus d^3
\]

\[
e = a^1 \oplus a^2 \oplus a^3 \oplus b^1 c^1 \oplus b^1 c^2 \oplus b^1 c^3 \oplus b^2 c^1 \oplus b^2 c^2 \oplus b^2 c^3 \oplus b^3 c^1 \oplus b^3 c^2 \oplus b^3 c^3 \oplus d^1 \oplus d^2 \oplus d^3
\]

can be arbitrarily distributed among two functions \( e^i \) affecting uniformity

direct sharing:

\[
e^1 = b^2 c^3 \oplus b^3 c^2 \oplus a^2 \oplus d^2 \oplus b^2 c^2 \oplus b^2 \oplus a^1 d^1
\]

\[
e^2 = b^3 c^1 \oplus b^1 c^3 \oplus a^3 \oplus d^3 \oplus b^3 c^3 \oplus a^1 d^1
\]

\[
e^3 = b^1 c^2 \oplus b^2 c^1 \oplus a^1 \oplus d^1 \oplus b^1 c^1 \oplus b^2 \oplus a^1 d^1
\]

correction term, no effect on functionality, but on uniformity

one output bit of the Sbox

are clear where to go (which \( e^i \))
Correction Terms

- What happens if the correction terms are the last coming signals — i.e., the logic is the slowest (critical path)
Correction Terms

- What happens if the correction terms are the last coming signals — i.e., the logic is the slowest (critical path)

- This will be the result

- fault-free, uniform
- unstable
- fault-free, non-uniform
- faulty
Correction Terms

- What happens if the correction terms are the last coming signals – i.e., the logic is the slowest (critical path)

- This will be the result

\[ x^1, x^2, x^3 \rightarrow f^1, f^2, f^3 \rightarrow y^1, y^2, y^3 \]

\[ \text{clock} \]

1. fault-free, uniform
2. unstable
3. fault-free, non-uniform
4. faulty
Case Study (PRESENT 1\textsuperscript{st}-order TI)

- FPGA prototype, Spartan-6 (SAKURA-G), SCA evaluation platform
Case Study (PRESENT 1st-order TI)

- FPGA prototype, Spartan-6 (SAKURA-G), SCA evaluation platform
How Complicated?

- Goal: to make particular logic slow
  - one option: make the routing long (routing in FPGA: active elements)
TI vs. Higher-Order Attacks

- Applicable on higher-order TI, our case study 1\textsuperscript{st}-order TI
  - Higher-order attacks still possible
    - can be hindered by lower SNR (Signal to Noise Ratio)
      - e.g., by introducing noise generator
TI vs. Higher-Order Attacks

- Applicable on higher-order TI, our case study 1\textsuperscript{st}-order TI
  - Higher-order attacks still possible
    - can be hindered by lower SNR (Signal to Noise Ratio)
      - e.g., by introducing noise generator

\[ x^{19} + x^{18} + x^{17} + x^{14} + 1 \]

(2 Slices)
Performance Results

- **SAKURA-G (Spartan-6 FPGA, excl. the noise module)**

<table>
<thead>
<tr>
<th>Design</th>
<th>Method</th>
<th>FF</th>
<th>LUT</th>
<th>Slice</th>
<th>Frequency [MHz]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>-</td>
<td>299</td>
<td>291</td>
<td>35</td>
<td>226</td>
</tr>
</tbody>
</table>

219.2 MHz
## Performance Results

- **SAKURA-G (Spartan-6 FPGA, excl. the noise module)**

<table>
<thead>
<tr>
<th>Design</th>
<th>Method</th>
<th>Utilization</th>
<th>Frequency [MHz]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Malicious</td>
<td>switch box</td>
<td>FF: 299, LUT: 291, Slice: 35, 226</td>
<td>212.8</td>
</tr>
</tbody>
</table>

Diagram illustrating the utilization and frequency for the original and malicious designs.
SCA Results (leakage detection, t-test, fixed vs. random)

@ 168 MHz
100 million traces
SCA Results (leakage detection, t-test, fixed vs. random)

@ 168 MHz
100 million traces
SCA Results (leakage detection, t-test, fixed vs. random)

@ 216 MHz
100 million traces
SCA Results (leakage detection, t-test, fixed vs. random)

@ 216 MHz
100 million traces
SCA Results (DPA attack)

- Target: a key nibble (4-bit PRESENT Sbox)
  - a single Sbox output bit at the first round
  - @216 MHz, 50 million traces
Raising Questions/Conclusions

- How about ASIC?
  - possible by changing the transistors’ characteristics
    - exchanging the gates with their high-threshold variants
    - fabricated @ 90nm and 65nm, under evaluation
- Possible scenario?
  - third IP cores & manufacturers
  - malicious design’s max freq: 196 MHz
    - will not be evaluated SCA by labs at < 197 MHz (not functional)
    - Trojan adversary runs it at 216 MHz → exploits the leakage
- Control over the clock, realistic?
  - FGPA, usually true (external clock + PLL/DCM)
  - ASIC, usually not true, the same effect by lower supply voltage
    - overclocking & power supply reduction should be internally monitored
Thanks!

any questions?

amir.moradi@rub.de

Embedded Security Group, Ruhr University Bochum, Germany