
Jean-Luc Danger

Electrical Engineering Artificial Intelligence Day

19 Novembre 2020
Physical security of embedded systems

- **Side Channel Analysis, or Passive Attacks:**
  - Exploit the observation of non-functional channels: power consumption, electromagnetic radiations, cache timing, ...

- **Fault Injection Attacks, or Active Attacks**
  - Disturb the computation to create faults on sensitive operations: clock glitches, electromagnetic pulses or harmonics, laser shot, ...

- **Hardware Trojan Horses**
  - Malevolent Design modification to make the system inoperative, controllable or with leakages.

- **Reverse Engineering, probing,**

**Many Physical threats !**
Machine Learning for Physical Security

- **ML is a relevant tool:**
  - For security analysis
    - The designer looks for vulnerabilities and the security level, thus can better protect the most sensitive parts
    - Can also be used by an attacker
  - For detection of abnormal situations
    - IDS (Intrusion Detection System)
    - Real time security monitoring
    - Presence of Hardware Trojan Horse

- The security of ML implementation can be compromised by physical attacks
Outline

- **ML for hardware security**
  - Example of analysis:
    - PUF
  - Example of detection
    - Hardware Trojan Horse

- **Security of ML**
  - Example of a CNN implementation
Example of ML analysis  
Physically Unclonable Function: PUF

- Function returning the **fingerprint** of a device  
  - Physical function,  
  - which exploits **material randomness**, during fabrication  
  - and is **unclonable**: same structure for each device

PUFs are instantiations of blueprints by a fab plant

**GDS2** (blueprint) → factory ➔ i.i.d. (ideally) → PUF 1 ➔ PUF 2 ➔ PUF M

a PUF ID is unique to each device
**PUF delivers a "Fingerprint"**

- List of pairs challenges / responses,

```
| Challenge | PUF | Response |
```

Many challenges => The PUF is "strong" => CRP protocol

- or unique identifier

```
| PUF | ID |
```

Few challenges => The PUF is "weak" => cryptographic protocol

ID can be used as a cryptographic key!
The most famous PUF: the Arbiter-PUF

- Delay difference between two identical paths:

![Diagram of the Arbiter-PUF]

- "Strong" PUF: many challenges for the CRP protocol

But attacked by Machine Learning!

- The arbiter PUF can be modelled as:

\[ B_i = \text{sign}(c_i \cdot X) \]

- Attack by Logistic regression (supervised ML)
  - The ML is trained by CRPs

<table>
<thead>
<tr>
<th>ML Method</th>
<th>No. of Stages</th>
<th>Prediction Rate</th>
<th>CRPs</th>
<th>Training Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>LR</td>
<td>64</td>
<td>95%</td>
<td>640</td>
<td>0.01 sec</td>
</tr>
<tr>
<td></td>
<td></td>
<td>99%</td>
<td>2,555</td>
<td>0.13 sec</td>
</tr>
<tr>
<td></td>
<td></td>
<td>99.9%</td>
<td>18,050</td>
<td>0.60 sec</td>
</tr>
<tr>
<td>LR</td>
<td>128</td>
<td>95%</td>
<td>1,350</td>
<td>0.06 sec</td>
</tr>
<tr>
<td></td>
<td></td>
<td>99%</td>
<td>5,570</td>
<td>0.51 sec</td>
</tr>
<tr>
<td></td>
<td></td>
<td>99.9%</td>
<td>39,200</td>
<td>2.10 sec</td>
</tr>
</tbody>
</table>

Very easy to attack by ML!

This attack is called modeling attack

Ulrich Rührmair, Frank Sehnke, Jan Sölter, Gideon Dror, Srinivas Devadas, and Jürgen Schmidhuber. “Modeling attacks on physical unclonable functions”. In Proceedings of the 17th ACM
The arbiter PUF has to be protected

Lightweight secure PUF  XOR PUF

Feed Forward PUF

The response of arbiter 1 is used as a challenge bit of a cascaded arbiter PUF
But modeling attack still works in reasonable time

<table>
<thead>
<tr>
<th>No. of Stages</th>
<th>Pred. Rate</th>
<th>No. of XORs</th>
<th>CRPs</th>
<th>Training Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>99%</td>
<td>3</td>
<td>6,000</td>
<td>8.9 sec</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>12,000</td>
<td>1:28 hrs</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>300,000</td>
<td>13:06 hrs</td>
</tr>
<tr>
<td>128</td>
<td>99%</td>
<td>3</td>
<td>15,000</td>
<td>40 sec</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>500,000</td>
<td>59:42 min</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>$10^6$</td>
<td>267 days</td>
</tr>
</tbody>
</table>

Lightweight secure PUF

XOR PUF

<table>
<thead>
<tr>
<th>ML Method</th>
<th>No. of Stages</th>
<th>Pred. Rate</th>
<th>No. of XORs</th>
<th>CRPs</th>
<th>Training Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>LR</td>
<td>64</td>
<td>99%</td>
<td>4</td>
<td>12,000</td>
<td>3:42 min</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td>80,000</td>
<td>2:08 hrs</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6</td>
<td>200,000</td>
<td>31:01 hrs</td>
</tr>
<tr>
<td>LR</td>
<td>128</td>
<td>99%</td>
<td>4</td>
<td>24,000</td>
<td>2:52 hrs</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td>500,000</td>
<td>16:36 hrs</td>
</tr>
</tbody>
</table>

FF PUF

<table>
<thead>
<tr>
<th>No. of Stages</th>
<th>FF-loops</th>
<th>Pred. Rate</th>
<th>CRPs</th>
<th>Training Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>6</td>
<td>97.72%</td>
<td>50,000</td>
<td>27:20 hrs</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>97.37%</td>
<td>50,000</td>
<td>27:20 hrs</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>95.46%</td>
<td>50,000</td>
<td>27:20 hrs</td>
</tr>
</tbody>
</table>
Protection by challenge obfuscation

Challenge obfuscation

But ML attack can exploit Power traces

Combined ML + side-channel attack

Simulation without noise noise

ML attacks works even high noise level

realistic noise in a circuit $\sigma \sim 10^{-4}$

<table>
<thead>
<tr>
<th></th>
<th>#Traces training</th>
<th>#Traces attacking</th>
<th>Train acc $\sigma = 0$</th>
<th>Attack acc $\sigma = 0$</th>
<th>Train acc $\sigma = 2.5$</th>
<th>Attack acc $\sigma = 2.5$</th>
<th>Train acc $\sigma = 16$</th>
<th>Attack acc $\sigma = 16$</th>
<th>Train acc $\sigma = 32$</th>
<th>Attack acc $\sigma = 32$</th>
<th>Train acc $\sigma = 64$</th>
<th>Attack acc $\sigma = 64$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
<td>500</td>
<td>500</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>0.9936</td>
<td>0.9964</td>
<td>1.0000</td>
<td>0.8740</td>
</tr>
<tr>
<td></td>
<td>2000</td>
<td></td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>0.9956</td>
<td>0.9960</td>
<td>1.0000</td>
<td>0.9656</td>
</tr>
<tr>
<td></td>
<td>5000</td>
<td></td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>0.9980</td>
<td>0.9980</td>
<td>1.0000</td>
<td>0.9098</td>
</tr>
<tr>
<td>DT</td>
<td>500</td>
<td>500</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>0.9994</td>
<td>1.0000</td>
<td>0.7996</td>
<td>1.0000</td>
<td>0.6640</td>
<td>1.0000</td>
<td>0.5700</td>
</tr>
<tr>
<td></td>
<td>2000</td>
<td></td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>0.9996</td>
<td>1.0000</td>
<td>0.8356</td>
<td>1.0000</td>
<td>0.6820</td>
<td>1.0000</td>
<td>0.5842</td>
</tr>
<tr>
<td></td>
<td>5000</td>
<td></td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>0.8448</td>
<td>1.0000</td>
<td>0.7114</td>
<td>1.0000</td>
<td>0.5916</td>
</tr>
<tr>
<td>RF</td>
<td>500</td>
<td>500</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>0.9996</td>
<td>1.0000</td>
<td>0.9618</td>
<td>0.9980</td>
<td>0.7310</td>
</tr>
<tr>
<td></td>
<td>2000</td>
<td></td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>0.9996</td>
<td>0.9644</td>
<td>0.9740</td>
<td>0.9728</td>
<td>0.7928</td>
</tr>
<tr>
<td></td>
<td>5000</td>
<td></td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>0.9930</td>
<td>0.9610</td>
<td>0.9604</td>
<td>0.8605</td>
<td></td>
</tr>
</tbody>
</table>

The training sequence is a set of power traces of different challenges on a reference PUF.

$\sigma = 16e^{-4}$

No necessity to preprocess the traces to reduce the noise.
Necessity to protect against ML+SCA attack

Balancing the power with the dual DFF

Random initialization of the initial state

Outline

- **ML for hardware security**
  - Example of analysis:
    - PUF
  - Example of detection
    - Hardware Trojan Horse

- **Security of ML**
  - Example of a CNN implementation
Hardware Trojan Horse

- Potential attack due to outsourcing
  - Design center, fabrication, validation …
  - Small hardware block to change add malevolent features (DoS, performance loss, high power, spying,…)

<table>
<thead>
<tr>
<th>Year</th>
<th>Reporter</th>
<th>HTs detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>2018</td>
<td>Bloomberg</td>
<td>China used a tiny Chip to infiltrate 30 big U.S. Companies</td>
</tr>
<tr>
<td>2014</td>
<td>Defensenum.com</td>
<td>Specific US-made components designed to intercept the satellites’ communications in France-UAE satellite</td>
</tr>
<tr>
<td></td>
<td>Edward Snowden</td>
<td>NSA planted back-doors in Cisco products as routers</td>
</tr>
<tr>
<td></td>
<td>Arstechnica and Spiegel</td>
<td>NSA secret toolbox used for inserting the backdoors and spy gadgets in different products</td>
</tr>
<tr>
<td>2012</td>
<td>Sergei Skorobogatov &amp; Christopher Woods</td>
<td>The discovery of a backdoor inserted into the Actel/Microsemi ProASIC3 chips (military grade chip)</td>
</tr>
<tr>
<td></td>
<td>Jonathan Brossard</td>
<td>A concept of a hardware backdoor called &quot;Rakshasa&quot; that China could embed in every computer</td>
</tr>
<tr>
<td></td>
<td>Kryptowire</td>
<td>Found a backdoor on ZTE Android phones</td>
</tr>
<tr>
<td>From 2007</td>
<td>Academic</td>
<td>Many examples of HT on different targets (cryptography IPs, processors, Wireless etc.)</td>
</tr>
</tbody>
</table>
HTH Countermeasures

HT Protection

Prevention (Pre-silicon)
- Heuristic Method
- Provable Method
- Supportive Design
  - Test Methods
  - Run-time Method

Detection (Pre-silicon)

Detection (Post-Silicon)
- Destructive
- Non Destructive
  - Optical
  - Logic Testing
  - Side-Channel

non invasive
HTH detection by ML

- **State of the art of HTH detection**
  - Statistical tests (T-Test) to compare the equality of population according to the null hypothesis.

- **Test example**
  - 3 HTHs of different sizes in RISC-V CPU running in FPGA:
    - 2 HTHs (HT1 & HT2) are inserted PicoRV32 target
    - 1 HTH (HT3) is inserted in Freedom E300 target

<table>
<thead>
<tr>
<th></th>
<th>Target design</th>
<th>Insertion phase</th>
<th>Overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>HT1</td>
<td>PicoRV32</td>
<td>P&amp;R</td>
<td>0.53%</td>
</tr>
<tr>
<td>HT2</td>
<td>PicoRV32</td>
<td>P&amp;R</td>
<td>0.27%</td>
</tr>
<tr>
<td>HT3</td>
<td>Freedom</td>
<td>RTL</td>
<td>0.1%</td>
</tr>
</tbody>
</table>

HT1 insertion

HT1 location: in red (HT ALM logic cell) in blue (The corresponding Logic blocks)

Logic blocks used by design

Unused logic blocks

PicoRV32 without HT

PicoRV32 with HT1
ML Detection Methodology

- **Acquire data for training**
  - 2 FPGAs are used: Reference and HT
  - The dataset comes from N cartographies of the device.
  - Each cartography is a matrix of 13 * 13 points having each EM traces of 5000 samples

- **Train with supervised ML algorithms**
  - SVM, Multi-Layer Perceptron, Decision Tree, KNN

- **Acquire data on target FPGA**

- **Apply the trained models to decide if there is a HTH in the target FPGA**
Results with T-test

Accuracy < 80%  Many false positives
Results in ML 1/2

(a) Support Vector Machine

(b) Multi-layer Perceptron
Results in ML 2/2

(c) Decision Tree Classification

Accuracy >80% even for a tiny HTH
Outline

- **ML for hardware security**
  - Example of analysis:
    - PUF
  - Example of detection
    - Hardware Trojan Horse

- **Security of ML**
  - Example of a CNN implementation
Attack of CNN implementation

- The CNN security requires:
  - Protection of the trained model which is often patented
  - Protection of the user privacy, when personal input data are computed with CNN
  - Protection of the output to prevent adversarial attacks

- But the implementation can be attacked by side-channel: the cache timing attack
Cache Timing attack example: Flush and Reload

Cache HIT: The victim used the probed address

Cache MISS: The victim did not use the probed address
Example: Cache Telepathy Attack

- Computation of convolutional layers are transformed into single matrix multiplications by using GEMM:

![Diagram of convolutional layer computation]

Side channel leakage when using Gemm

- 3 functions are repeatedly used
  - Kernel, itcopy, oncropy
  - They form specific patterns according to the iteration type and length.

- The cache attack allows to count the function calls and determine the number of layers, the input, output, and filter size.

- Protections
  - Active research*

* TP: Linda Guiga CIFRE PhD with Idemia
Conclusion

- ML algorithms provide powerful tools for the security of embedded systems:
  - Point out design weaknesses, as modeling and cloning unclonable physical functions.
  - Efficient leakage analysis by profiling and combining with side-channels traces.
  - No necessity of preprocessing to reduce noise.
  - Detection of abnormal behavior as those coming from stealthy Hardware Trojan Horses.
  - Active research for IDS in connected cars*

- But its implementation can be vulnerable to physical attacks.

* TP: Natasha AlKhatib PhD in the C3S chair
Thank you for your attention