Towards the Automatic Application of Physical Attacks Countermeasures

Dr. Francesco Regazzoni
A bit of history

- 1948 Transistor
- Design done by hand
- 1970 Automated place and route
- 1980 Chip design with programming languages
A bit of history

- 1948 Transistor
  - Design done by hand
- 1970 Automated place and route
- 1980 Chip design with programming languages

- Chip is most likely to function correctly
- Chip is easier to be verified
- Designer can handle more complex designs
- Birth of commercial EDA companies
First consideration....

- 1996 Timing Attacks
- 2018...

Now is time!
Why Automation....

Design done by hands

Simple circuits

EDA tools

Complex and large circuits
Why Automation....

Design done by hands
Why Automation....

Design done by hands || Simple circuits
## Why Automation....

<table>
<thead>
<tr>
<th>Design done by hands</th>
<th>Simple circuits</th>
</tr>
</thead>
<tbody>
<tr>
<td>EDA tools</td>
<td></td>
</tr>
</tbody>
</table>

Dr. Francesco Regazzoni  
RISE Spring School, Cambridge, 28 March 2018
Why Automation....

Design done by hands
EDA tools

Simple circuits
Complex and large circuits
Security is very often considered at later stages of design

Cost and Time to Market

Possible Security pitfalls

Handle the Complexity
... for security?

- Security is very often considered at later stages of design
- Cost and Time to Market
- Possible Security pitfalls
- Handle the Complexity

EXTRA CONSTRAINT

Use as much as possible “standard” design commodities!
A bit of history

- 1996 Physical attacks
- Countermeasures done by hand
- 2004 Secured synthesis and place and route
- 2009 Tool driven by a security variable

---

\textsuperscript{a} Kris Tiri, Ingrid Verbauwhede, “A Logic Level Design Methodology for a Secure DPA Resistant ASIC or FPGA Implementation” DATE 2004:

\textsuperscript{b} Francesco Regazzoni, Alessandro Cevrero, François-Xavier Standaert, Stéphane Badel, Theo Kluter, Philip Brisk, Yusuf Leblebici, Paolo Ienne, “A Design Flow and Evaluation Framework for DPA-Resistant Instruction Set Extensions” CHES 2009
A bit of history

- 1996 Physical attacks
- Countermeasures done by hand
- 2004 Secured synthesis and place and route \(^a\)
- 2009 Tool driven by a security variable \(^b\)

\(^a\) Kris Tiri, Ingrid Verbauwhede, “A Logic Level Design Methodology for a Secure DPA Resistant ASIC or FPGA Implementation” DATE 2004:


Still only goals

- Chip would most likely to function securely
- Chip security would be easier to be verified
- Designer could handle more complex designs
Enabling the automatic design for DPA resistance

- Identify sensitive parts
- Partition sensitive / non-sensitive
- Protect sensitive
- Security evaluation

Measure the DPA resistance?
Partition the algorithm?
Asserting the DPA resistance?
Countermeasure and its design flow?
Needed “Basic Blocks”

- Identify sensitive parts
  - Partition Sensitive / Non Sensitive
  - Protect Sensitive
  - Security Evaluation

- Generate useful power traces?
Needed “Basic Blocks”

- Identify sensitive parts
- Partition Sensitive / Non Sensitive
- Protect Sensitive
- Security Evaluation

- Generate useful power traces?
- Measure the DPA resistance?
Needed “Basic Blocks”

- Identify sensitive parts
- Partition Sensitive / Non Sensitive
- Protect Sensitive
- Security Evaluation

- Generate useful power traces?
- Measure the DPA resistance?
- Countermeasure and its design flow?
Needed “Basic Blocks”

- Generate useful power traces?
- Measure the DPA resistance?
- Countermeasure and its design flow?
- Partition the algorithm?
Where are we?
Automated Synthesis

INPUT:
- HDL Description
- Technological Library (area, timing, power)
- Synthetic Library (multipliers...)
- Constraints

OUTPUT:
- DPA resistant Gate Level Netlist
- Estimation of area, timing, power (!)
- Timing constraints
INPUT:

- DPA resistant Gate Level Netlist
- Technological Library
- Estimation of area, timing, power (!)
- Timing constraints
- Secure Place and Route Script

OUTPUT:

- DPA resistant fabrication file
Towards Automatic Application of Countermeasures

**Inputs:**
- Unprotected Algorithm
- Countermeasure

**Output:**
- Algorithm where the countermeasure is Applied
  
  Algorithm where the countermeasure is applied does NOT mean protected Algorithm
// Calculate S-box (plaintext XOR key)
int PRESENT(int plaintext, int key) {
  1 int result = 0;  // initialize the result
  2 plaintext = plaintext ^ key;  // perform the xor with the key
  3 result = S[plaintext];  // perform the S-box
  4 return result;  };  // return the result
Customizable Processors

// Calculate S-box (plaintext XOR key)
int PRESENT(int plaintext, int key) {
  1 int result = 0; // initialize the result
  2 plaintext = plaintext ^ key; // perform the xor with the key
  3 result = S[plaintext]; // perform the S-box
  4 return result; }; // return the result

// Calculate S-box (plaintext XOR key)
int PRESENT_XOR+S-box-ISE(int plaintex) {
  1 int result = 0; // initialize the result
  2 Instr_1(plaintex, result);
  3 return result; }; // return the result
Protected / Non Protected CO-Design!
What about Software?

- Information Leakage Analysis
- Transformation Target Identification
- Code Transformation

**Input Software Implementation**
- `sbcir21,0xfd`
- `ld r25,Y`
- `movw r18,r26`
- `subi r18,0x4f`

**Sensitive Parts**
- `sbcir21,0xfd`
- `ld r25,Y`
- `movw r18,r26`
- `subi r18,0x4f`

**Targets for Protection**
- Example (A)
  - `sbcir21,0xfd`
  - `lds r23,705`
  - `mov r25,r23`
  - `ld r25,Y`
  - `lds r23,705`
  - `mov r18,r23`
  - `mov r19,r23`
  - `movw r18,r26`
  - `subi r18,0x4f`

**Protected Implementation**
- Example (B)

**Targets for Protection**
- Example (B)

**Protected Implementation**
- Example (B)
Information Leakage Analysis

Sensitivity values for unprotected implementation

- ARK
- SB
- SR
- MC

Sensitivity (Mutual information)

Clock cycle

Dr. Francesco Regazzoni
RISE Spring School, Cambridge, 28 March 2018
Example on Software

Instruction – time mapping of unprotected implementation

Clock cycle

Instruction – time mapping of protected implementation
Example on Software

Instruction – time mapping of unprotected implementation

Instruction – time mapping of protected implementation

Sensitivity
(Mutual information)
Clock cycle
Instruction − time mapping of unprotected implementation

Sensitivity
(Mutual information)
Clock cycle
Instruction − time mapping of protected implementation

Dr. Francesco Regazzoni
RISE Spring School, Cambridge, 28 March 2018
P. 27
Security Evaluation

Sensitivity values for protected implementation

Clock cycle

ARK SB SR MC

Sensitivity (Mutual information)

Dr. Francesco Regazzoni
RISE Spring School, Cambridge, 28 March 2018
Step Three
**Inputs:**
- Algorithm where the countermeasure is Applied
- Countermeasure

**Output:**
- Assertion of the Correct Application of the Countermeasure

- Assertion of the correct application of the countermeasure **does NOT** mean protected Algorithm
### Do We Need Verification?

**Dr. Francesco Regazzoni**  
RISE Spring School, Cambridge, 28 March 2018

```c
void maskedARK() {
    unsigned char i;
    for (i=0;i<16;i++){
        st[i] = pt[i] ^ (key[i] ^ mask[i]);
    }
}
```

```assembly
.text
.global ARK
.type ARK, @function

ARK:
/* prologue: function */
/* frame size = 0 */
/* stack size = 0 */
.L__stack_usage = 0

ldi r24,key
ldi r25,pt
eor r24,r25

ldi r25,mask
eor r24,r25
sts st,r24

ldi r24,key+1
ldi r25,pt+1
eor r24,r25

...
```

 AVR-gcc-4.5.3-O3
Goal

Given a program, find the sensitive operations, which leak critical information.

Define three types for variables:

- Secret
- Public
- Random
Methodology

- Represent the program as a graph
- Use satisfiability queries to detect the dependencies and sensitivity
- Is it a **Don’t care** from random point of view?

- If at least one bit is not a don’t care, it is random, so ok.

- Else, check if is a **Don’t care** from some secret variable?

- If at least a bit is not a don’t care, then is sensitive.
Identified Problems

- Compiler problems
- Programmer problems (shift with hamming distance leakage)
- Countermeasure problem (Goubin [2001])
Conclusions

- Physical security is a concern
- Design automation and Verification for physical security is crucial for Embedded systems
- Initial steps for power analysis are promising
- This is just the beginning...
Thank you for your attention!

mail: regazzoni@alari.ch