Towards Leakage-Resilient Cryptographic Devices

F.-X. Standaert

UCL Crypto Group, Université catholique de Louvain

UCL CESAME Seminar - May 2010
Outline

1. The cryptographic HW design space
   2.1 Introduction
   2.2 Countermeasures
   2.3 Back to the design space issue
3. Towards a formal understanding of physical security
   3.1 Do we need formal tools?
   3.2 Positive results
   3.3 Back to the design space issue
4. Reshaping certain cryptographic intuitions
5. Conclusions
The cryptographic HW design space
Outline

1. The cryptographic HW design space
   2.1 Introduction
   2.2 Countermeasures
   2.3 Back to the design space issue
3. Towards a formal understanding of physical security
   3.1 Do we need formal tools?
   3.2 Positive results
   3.3 Back to the design space issue
4. Reshaping certain cryptographic intuitions
5. Conclusions
Side-channel attacks

Classical cryptanalysis

\[ P \rightarrow \text{cryptographic algorithm} \rightarrow C \]

\[ K \quad ? \]
Side-channel attacks

Side-channel cryptanalysis

P \rightarrow \text{physical implementation} \rightarrow C

K \rightarrow ?

\text{power consumption} \rightarrow \text{electromagnetic radiations} \rightarrow \ldots
Origin of the attacks

- Any data dependent physical leakage
- e.g. dynamic power consumption in CMOS devices

\[ P_{dyn} = C_L V_{DD}^2 P_{0\rightarrow1} f \]

- \( P_{dyn} \Rightarrow SCA \)
- But \( (P_{dyn} = 0) \nRightarrow \) no side-channel attack!
Exemplary attack against the DES

- The Data Encryption Standard
- FPGA implementation, loop architecture
Exemplary attack against the DES

1. Input selection: random plaintexts
2. Internal values derivation
3. Leakage modeling (Hamming weights)
Exemplary attack against the DES

4. Leakage measurement
5. Leakage reduction (select representative samples)
Exemplary attack against the DES

- In practice, power consumption vs. EM radiation
Exemplary attack against the DES

6. Statistical test
   ▶ e.g. correlation coefficient

<table>
<thead>
<tr>
<th>Key[0…5]</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>corr</td>
<td>-0.09</td>
<td>0.05</td>
<td>0.32</td>
<td>-0.11</td>
</tr>
</tbody>
</table>

Correct key candidate
Improved attacks

- Adaptive selection of the inputs
- Pre-processing of the traces (e.g. averaging, filtering)
- Improved leakage models by profiling, characterization
- Exploitation of multiple samples, multivariate statistics
  - Higher-order attacks
  - Template attacks
- Different statistical tests
  - Difference of mean
  - Correlation analysis
  - Bayesian classification
- ...
Outline

1. The cryptographic HW design space
   2.1 Introduction
   2.2 Countermeasures
   2.3 Back to the design space issue
3. Towards a formal understanding of physical security
   3.1 Do we need formal tools?
   3.2 Positive results
   3.3 Back to the design space issue
4. Reshaping certain cryptographic intuitions
5. Conclusions
Countermeasures

- Physical level
  - Shields, conforming glues, PUFs, detectors
  - Detachable power supplies
- Technological level
  - Dynamic and differential logic styles
  - Noise addition
- Algorithmic level
  - Time randomization, encryption of the buses
  - Hiding, masking
- Protocol level (e.g. key updates)
Example #1: masking

- Goal: have data-independent leakage
- How: by “randomizing” the computation
- e.g. block cipher S-box

\[ S(p \oplus k \oplus m) = S(p \oplus k) \oplus q \]
Example #1: masking

- $R_1(L) \perp k$, $R_2(L) \perp k$

- But $\exists f$ such that $f(R_1(L), R_2(L)) \propto k$
  - Univariate $\rightarrow$ bivariate
  - The rest of the attack remains unchanged
Example #2: hiding

- Goal: have data-independent leakage
- How: by forcing constant leakage
- e.g. dual rail precharged logic WDDL
Example #2: hiding

- Hamming weight/distance models seem meaningless
- But ∃ data dependent leakage variations
- ∃f such that R(L) ∝ f(p, k)
- An efficient attack may require to
  - Change the leakage model
    - But possibly implies a ≠ adversarial context
  - Use device-independent attacks
Outline

1. The cryptographic HW design space
   2.1 Introduction
   2.2 Countermeasures
   2.3 Back to the design space issue
3. Towards a formal understanding of physical security
   3.1 Do we need formal tools?
   3.2 Positive results
   3.3 Back to the design space issue
4. Reshaping certain cryptographic intuitions
5. Conclusions
Physical security as a design criteria

- Countermeasures never perfect
- Have to be traded for implementation cost!
- Device-dependent security ($\Rightarrow$ huge design space)
Outline

1. The cryptographic HW design space
   2.1 Introduction
   2.2 Countermeasures
   2.3 Back to the design space issue
3. Towards a formal understanding of physical security
   3.1 Do we need formal tools?
   3.2 Positive results
   3.3 Back to the design space issue
4. Reshaping certain cryptographic intuitions
5. Conclusions
A motivating example

- Goal: fair evaluation and comparison of two implementations (AES-CMOS and AES-WDDL)
- Tool: adversary $A := \{ \text{correlation, } H_W, \text{ 8-bit target} \}$
  - $\text{Succ}^{\text{sc-kr}}_{A_{AES-CMOS}}(q, \ldots) = 0.9$ for $q = 10$
  - $\text{Succ}^{\text{sc-kr}}_{A_{AES-WDDL}}(q, \ldots) = 0.9$ for $q = 10000$

- What we actually use:
  - CMOS (resp. WDDL): suboptimal distinguisher with suboptimal (resp. meaningless) leakage model

$\Rightarrow$ We perform an evaluation of the adversary rather than a comparison of the implementations
Possible issues

1. Bad leakage model
2. Bad distinguisher, e.g. Pearson’s correlation coefficient measures only *linear* dependencies between two r.v.’s

\[
\rho(X, Y) = \frac{\text{cov}(X, Y)}{\sigma_X \cdot \sigma_Y} = \frac{\mathbb{E}[XY] - \mathbb{E}[X] \cdot \mathbb{E}[Y]}{\sigma_X \cdot \sigma_Y}.
\]
Possible issues

1. Bad leakage model
2. Bad distinguisher, e.g. Pearson’s correlation coefficient measures only *linear* dependencies between two r.v.’s

\[ \rho(X, Y) = \frac{\text{cov}(X, Y)}{\sigma_X \cdot \sigma_Y} = \frac{E[XY] - E[X] \cdot E[Y]}{\sigma_X \cdot \sigma_Y}. \]

\[
\begin{array}{cccccc}
1.0 & 0.8 & 0.4 & 0.0 & -0.4 & -0.8 & -1.0
\end{array}
\]
Possible issues

1. Bad leakage model
2. Bad distinguisher, e.g. Pearson’s correlation coefficient measures only *linear* dependencies between two r.v.’s

\[ \rho(X, Y) = \frac{\text{cov}(X, Y)}{\sigma_X \cdot \sigma_Y} = \frac{E[XY] - E[X] \cdot E[Y]}{\sigma_X \cdot \sigma_Y}. \]
Fair(er) evaluation

- Requires to separate implementations and adversaries

Fair(er) evaluation

In theory:
- $H[K|L]$ captures any leakage dependency
- It relates to the asymptotic success rate of the (strongest possible) Bayesian adversary

In practice:
- Computing $H[K|L]$ requires to approximate the leakage pdf $Pr[K|L]$ (not straightforward because of the large number of samples in the leakage traces)

More precisely...
1. Asymptotic meaning of $H[S|L_q]$

“Can I approximate the leakage probability distribution?”

**Definition 1.** Asymptotic success rate of a side-channel key recovery adversary: $\text{Succ}_{A_{E_K,L}}^{sc-kr-o,S}(q \to \infty)$

**Definition 2.** Bayesian side-channel key recovery adversary: selects $\tilde{s} = \arg\max_{s^*} \Pr[s^*|l_q]$

**Definition 3.** Sound leakage probability distribution $\Pr[L_q|S]$ or approximation $\hat{\Pr}[\tilde{L}_q|S]$: if the first-order asymptotic success rate $\text{Succ}_{A_{E_K,L}}^{sc-kr-1,S}(q \to \infty) = 1$
1. Asymptotic meaning of $H[S|L_q]$ 

Bounded preparation / unbounded exploitation:

$$H^q_{s,s^*} = \begin{pmatrix} h_{1,1} & h_{1,2} & \ldots & h_{1,|S|} \\ h_{2,2} & h_{2,2} & \ldots & h_{2,|S|} \\ \vdots & \vdots & \ddots & \vdots \\ h_{|S|,1} & h_{|S|,2} & \ldots & h_{|S|,|S|} \end{pmatrix}$$

**Theorem 1.** (...) a leakage probability distribution is sound if and only if $\arg\min_{s^*} H^1_{s,s^*} = s, \forall s \in S$

Intuitively: the diagonal elements $h_{s,s}$’s are minimum
Example (AES Rijndael)
2. Comparative meaning of $H[S|L_q]$ 

“Does more entropy imply more security?”

$$H_{s,s^*}^q = \begin{pmatrix}
    h_{1,1} & h_{1,2} & \ldots & h_{1,|S|} \\
    h_{2,1} & h_{2,2} & \ldots & h_{2,|S|} \\
    \vdots & \vdots & \ddots & \vdots \\
    h_{|S|,1} & h_{|S|,2} & \ldots & h_{|S|,|S|}
\end{pmatrix}$$

$h_{s,s}$: residual entropy of a key class $s$

$H[S|L_q] = E_s H_{s,s}^q$ (averaged diagonal of $H_{s,s^*}^q$)
2. Comparative meaning of $H[S|L_q]$

**Definition 4.** $|S|$-target side-channel attack: tries to identify one key candidate out of $|S|$

**Definition 5.** Gaussian leakage distribution: such that $L(C_\alpha, M, R) = L'(C_\alpha, M) + L''(R)$, $L''(R) = \text{gaussian noise}$.

**Definition 6.** Ideal side-channel attack: Bayesian attack in which the leakages are perfectly predicted by the adversary’s approximated probability density function.
2. Comparative meaning of $H[S|L_q]$ 

Unbounded preparation / bounded exploitation

- Does more entropy imply more security?
  - Ideal 2-target attacks with Gaussian leakages: yes
  - Ideal $|S|$-target attacks with “perfect” leakages: yes
  - In general: no
    (a pdf cannot be summarized in a scalar value)

- In practice?
Outline

1. The cryptographic HW design space
   2.1 Introduction
   2.2 Countermeasures
   2.3 Back to the design space issue
3. Towards a formal understanding of physical security
   3.1 Do we need formal tools?
   3.2 Positive results
   3.3 Back to the design space issue
4. Reshaping certain cryptographic intuitions
5. Conclusions
1. Comparing masking schemes

For a fixed (simulated) leakage function (e.g. HD)

- SNR=10 -
1. Comparing masking schemes

- Meaningful intersection of the IT curves

\[ \text{SNR} = 10 \cdot \log_{10} \left( \frac{\varepsilon^2}{\sigma^2} \right) \]

- \text{SNR} = 11 -
2. Leakage dimensionality reduction

- Direct estimation of $\Pr[K|L]$ is computationally hard
- Need to find $R$ such that $\Pr[K|R(L)] \approx \Pr[K|L]$
2. Example: PCA, LDA (first directions)

- Intuitive selection of the points of interest
- EM dominates, power is a “backup” information
2. Example: PCA, LDA (last directions)

- LDA extracts information “better”
2. Experimental results: $\hat{H}[K|L]$

<table>
<thead>
<tr>
<th>Number of components</th>
<th>3</th>
<th>5</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>power (PCA)</td>
<td>4.62</td>
<td>4.49</td>
<td>4.57</td>
</tr>
<tr>
<td>power (LDA)</td>
<td>4.41</td>
<td>4.48</td>
<td>4.62</td>
</tr>
<tr>
<td>EM (PCA)</td>
<td>3.92</td>
<td>3.65</td>
<td>3.55</td>
</tr>
<tr>
<td>EM (LDA)</td>
<td>3.21</td>
<td>3.15</td>
<td>3.24</td>
</tr>
<tr>
<td>power + EM (PCA)</td>
<td>3.57</td>
<td>3.36</td>
<td>3.20</td>
</tr>
<tr>
<td>power + EM (LDA)</td>
<td>2.92</td>
<td>2.80</td>
<td>2.87</td>
</tr>
</tbody>
</table>

- power < EM < power + EM
- PCA < LDA
2. Real leakages vs. simulated models

- e.g. Hamming distance model:

\[
H[S|L_1] = - \sum_{h_d=0}^{4} \frac{2^4 \cdot \binom{4}{h_d}}{2^8} \cdot \log_2 \left( \frac{1}{2^4 \cdot \binom{4}{h_d}} \right) = 5.9694
\]

\[\Rightarrow \text{Significantly less informative than reality}\]
3. Security evaluation of template attacks
3. **IT analysis of template attacks**

- Main interest: can be done for fixed $q = 1$
3. Comparison of the profiling phases

- Y axis: model accuracy - X axis: speed of convergence
Outline

1. The cryptographic HW design space
   2.1 Introduction
   2.2 Countermeasures
   2.3 Back to the design space issue
3. Towards a formal understanding of physical security
   3.1 Do we need formal tools?
   3.2 Positive results
   3.3 Back to the design space issue
4. Reshaping certain cryptographic intuitions
5. Conclusions
**Multidimensional problem**

![Diagram](image.png)

- **Circuit complexity**
- **Algorithm level**
- **Functional block level**
- **Gate level**
- **Logic level**
- **Transistor level**
- **Layout level**
- **Physical level**

**Parameters** (technology, temperature, process, ...)

**AES Rijndael, FPGA implementation**
Too large design space?

- ∃ good tools to analyze every layer (algorithms, protocols, implementations, . . . )
- Best security analysis is physical
- Hard to implement every possible choice

⇒ Simulations are a necessary intermediate step (as for area, frequency, power consumption, . . . )
  - Example: instruction set extensions
Instruction set extensions
Instruction set extensions
Instruction set extensions
The CMOS design flow
The processor customization

software
crypto.c

ISE Extractor

processor HDL code

ISE HDL code

CMOS Synth and P&R

CMOS Library

crypto_ISE.c

SPICE level simulation
The protected design flow

1. Software: crypto.c
2. Processor HDL code
3. ISE HDL code
4. Protected Synth and P&R
5. CMOS Synth and P&R
6. CMOS Library
7. ISE Extractor
8. Protected Library

Diagram:
- Software: crypto.c -> ISE Extractor
- Processor HDL code -> ISE HDL code -> Protected Synth and P&R
- CMOS Synth and P&R
- CMOS Library
- Protected Library

Network:
- crypto.c
- ISE HDL code
- Processor HDL code
- Protected Synth and P&R
- CMOS Synth and P&R
- CMOS Library
- Protected Library

The diagram illustrates the flow from software to hardware, highlighting the protected design process.
The hybrid design flow
The simulation environment

- **Software**: `crypto.c`
- **Processor HDL Code**: `crypto_ISE.c`
- **ISE HDL Code**: `crypto_ISE.c`
- **Protected Library**: `crypto.c`
- **CMOS Library**: `crypto.c`
- **SPICE Level Simulation**: `crypto.c`
- **Synthesis and P&R**: `crypto.c`
- **ISE Extractor**: `crypto.c`
The design evaluation
Example: partitioning of a block cipher

Legend:

<table>
<thead>
<tr>
<th></th>
<th>non protected logic</th>
<th>protected logic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plain Text</td>
<td>key</td>
<td></td>
</tr>
<tr>
<td>sbox</td>
<td>result</td>
<td></td>
</tr>
<tr>
<td>Full CMOS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XOR ISE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S-box ISE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XOR + S-box ISE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>full ISE</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
ISE and its Source Code

```c
// Computes S-box (plaintext XOR key)

int PRESENT(int plaintext, int key) {
    int result = 0; // initialize the result
    plaintext = plaintext ^ key; // perform the xor with the key
    result = S[plaintext]; // perform the S-box
    return result; // return the result
}
```
Example of ISE and its Source Code

```c
// Computes S-box (plaintext XOR key)
int PRESENT(int plaintext, int key) {
    int result = 0;  // initialize the result
    plaintext = plaintext ^ key;  // perform the xor with the key
    result = S[plaintext];  // perform the S-box
    return result;  // return the result
}
```

```c
// Computes S-box (plaintext XOR key)
int PRESENT_XOR+S-box-ISE(int plaintext) {
    int result = 0;  // initialize the result  // instantiate the new instruction s-box(pt ^key)  
    Instr_1(plaintext, result);
    return result;  // return the result
}
```
Security Evaluation

![Graph showing mutual information vs. noise standard deviation for different configurations: full CMOS, XOR ISE, S-box ISE, XOR + S-box ISE, and full ISE. The graph illustrates the impact of noise on the security of cryptographic devices.]
Outline

1. The cryptographic HW design space
   2.1 Introduction
   2.2 Countermeasures
   2.3 Back to the design space issue
3. Towards a formal understanding of physical security
   3.1 Do we need formal tools?
   3.2 Positive results
   3.3 Back to the design space issue
4. Reshaping certain cryptographic intuitions
5. Conclusions
Side-channels vs. classical cryptanalysis

Which information can be used?
Side-channels vs. classical cryptanalysis

Standard DPA uses information from the first or last round (low diffusion)

Which information can be used?
Side-channels vs. classical cryptanalysis

Collision attacks are able to exploit the leakages up to the third round

Which information can be used?
Side-channels vs. classical cryptanalysis

Which information can be used?

What about the middle rounds?

Round 1 → $L_1$
Round 2 → $L_2$
Round 3 → $L_3$
Round 4 → $L_4$?
Round 5 → $L_5$?
Round 6 → $L_6$?
Round 7 → $L_7$?
Round 8 → $L_8$?
Round 9 → $L_9$?
Round 10 → $L_{10}$
Algebraic side-channel attacks

\[ P = x_2 + x_11 + x_14 + x_2x_5 + x_4x_10 \]

\[ 0 = x_5 + x_9 + x_1x_6 + x_3x_10x_13 \]

\[ 1 = x_7 + x_15 + x_16 + x_3x_7 + x_9x_11 \]

\[ 0 = x_10 + x_15 + x_16 + x_6x_12 + x_9x_16 \]
Algebraic side-channel attacks

\[ 0 = x_2 + x_{11} + x_{14} + x_2 x_5 + x_4 x_{10} \]
\[ 0 = x_5 + x_9 + x_1 x_6 + x_3 x_{10} x_{13} \]
\[ 1 = x_7 + x_{15} + x_{16} + x_3 x_7 + x_9 x_{11} \]
Algebraic side-channel attacks

\[ 0 = x_2 + x_{11} + x_{14} + x_2 x_5 + x_4 x_{10} \]
\[ 0 = x_5 + x_9 + x_1 x_6 + x_3 x_{10} x_{13} \]
\[ 1 = x_7 + x_{15} + x_{16} + x_3 x_7 + x_9 x_{11} \]
Algebraic side-channel attacks

\[ 0 = x_2 + x_{11} + x_{14} + x_2x_5 + x_4x_{10} \]
\[ 0 = x_5 + x_9 + x_1x_6 + x_3x_{10}x_{13} \]
\[ 1 = x_7 + x_{15} + x_{16} + x_3x_7 + x_9x_{11} \]
Algebraic side-channel attacks

\[ 0 = x_2 + x_{11} + x_{14} + x_2 x_5 + x_4 x_{10} \]
\[ 0 = x_5 + x_9 + x_1 x_6 + x_3 x_{10} x_{13} \]
\[ 1 = x_7 + x_{15} + x_{16} + x_3 x_7 + x_9 x_{11} \]

\[ 0 = x_7 + x_1 x_5 + x_6 x_{11} + x_4 x_5 x_{12} \]
\[ 1 = x_8 + x_{10} + x_{11} + x_{15} + x_2 x_6 + x_6 x_{10} \]
\[ 0 = x_{10} + x_{15} + x_{16} + x_6 x_{12} + x_9 x_{16} \]
Algebraic side-channel attacks

0 = x_2 + x_{11} + x_{14} + x_2 x_5 + x_4 x_{10}

0 = x_5 + x_9 + x_1 x_6 + x_3 x_{10} x_{13}

1 = x_7 + x_{15} + x_{16} + x_3 x_7 + x_9 x_{11}

0 = x_7 + x_1 x_5 + x_6 x_{11} + x_4 x_5 x_{12}

0 = x_{10} + x_{15} + x_{16} + x_6 x_{12} + x_9 x_{16}

power consumption

time samples

success rate (online phase)

number of target bytes
Comparison

Classical power analysis attack (DPA)

Target 1

8-bit subkey
Comparison

Classical power analysis attack (DPA)

Target 1

8-bit subkey

$L_1$  $K_1$
Comparison

Classical power analysis attack (DPA)

Target 1

8-bit subkey

\[ L_1 \]

\[ L_2 \]

\[ K_1 \]

\[ K_2 \]
Comparison

Classical power analysis attack (DPA)

Target 1

8-bit subkey

$L_1$

$L_2$

$K_1$

$K_2$
Comparison

Algebraic side-channel attack

Target 1

$L_1$

Algebraic cryptanalysis

Master key
Comparison

Algebraic side-channel attack

Target 1

Target 2

$L_1$

Master key

Algebraic cryptanalysis
Comparison

Algebraic side-channel attack

Target 1

Target 2

Target 3

$L_1$

$L_1$

Algebraic cryptanalysis

Master key
Comparison

Algebraic side-channel attack

Target 1
Target 2
Target 3

Master key

Algebraic cryptanalysis

\( L_1 \)

\( L_1 \)
Data complexity $q$

![Diagram showing AES and data complexity](image-url)
Practical impact

Repetition number $n_r$
Summarizing

Number of measurements = \( q \times n_r \)

- Related to the quantity of information we need
- Related to the accuracy of the measurements

Algebraic SCAs only need \( q = 1 \)!
Conclusions

- Security of a system = security of the weakest point
- Physical security implies new design challenges
  - New attacks become possible
  - New tools become useful
  - New theory (sometimes) required
- Standard DPA Attacks: algorithm-independent
- Algebraic Side-Channel Attacks: algorithm-dependent
THANKS

Questions?