Test and Hardware Security
Marion Doulcier, Giorgio Di Natale, Marie-Lise Flottes, Bruno Rouzeyre

To cite this version:

HAL Id: lirmm-00365276
https://hal-lirmm.ccsd.cnrs.fr/lirmm-00365276
Submitted on 2 Mar 2009

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Test & Security

G. DiNatale, M. Doulcier, M–L. Flottes, B. Rouzeyre

Pastis 2008
Circuit testing is mandatory to guarantee a good security level

A hardware defect may induce some security vulnerability

- But

<table>
<thead>
<tr>
<th></th>
<th>Test</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>Observability</td>
<td>![Green Arrow]</td>
<td>![Red Arrow]</td>
</tr>
<tr>
<td>Controlability</td>
<td>![Green Arrow]</td>
<td>![Red Arrow]</td>
</tr>
</tbody>
</table>
External Test + Scan path

- High fault coverage
- Automatic generation of scan chains
- Easy test sequence generation

Vulnerability
- Control and observation of internal states of CUT
  - => secret data retrieval
- **Built-in Self Test (BIST)**

  - No control/observation from the outside
  - Area overhead
  - Fault coverage (pseudo-random testing)?
Securing the scan chain

- **Goal**
  - ✓ No observation or control of the functional data processed by the secure system

- **Principle**
  - ✓ Prevent illegal scan shift operations

- **Solutions**
  - ✓ Test mode protection
    - Scan protocol
    - Test Patterns watermarking
  - ✓ System mode protection
    - Scan chain scrambling
    - Scan enable tree protection
    - Spy FFs

- Protection against illegal usage of the test mode
- Protection against scan chain probing attacks
Scan protocol

- The circuit is initialized before and after test mode
- Initialization is checked before switching to another mode
- Switch between the 2 modes, bypassing the initialization, is detected
Test mode protection

- Test pattern watermarking
  - Test patterns embed authentication keys
  - Keys are dynamically changed (e.g. LFSR-based)
System mode protection

- Scrambling method
  - Scan path with a prefixed segment organization during test mode
    - Time T1
      - Segment 1 → Segment 2 → Segment 3 → Segment 4
  - Scan path with random segment organization if shift during system mode
    - Time T2
      - Segment 1 → Segment 2 → Segment 3 → Segment 4
System mode protection

- **Scan–Enable Tree Protection**
  - Compare the scan enable signals at different locations

```
Test Controller

Scan Enable

Check the state of the test controller to any switch to 1

To Scan FFs

If 1 then error
```
- Spy Flip–Flops
  - Include Spy cells in the scan chain
  - Control the spy cells to a constant value
  - Observe the spy cells states
# Experimental results

<table>
<thead>
<tr>
<th>Insertion flow</th>
<th>Scrambling</th>
<th>Scan enable</th>
<th>Spy cell</th>
<th>Pattern watermarking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test</td>
<td>Test time</td>
<td>0%</td>
<td>1%</td>
<td>5%</td>
</tr>
<tr>
<td>Design</td>
<td>Area</td>
<td>0.2%</td>
<td>0.3%</td>
<td>1.8%</td>
</tr>
<tr>
<td></td>
<td>power c.</td>
<td>7%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td></td>
<td>Security</td>
<td>+++</td>
<td>++</td>
<td>++</td>
</tr>
</tbody>
</table>
To resume

- Countermeasures address two kinds of attack
  - Legal activation of the test circuitry
    - corruption of the authentication scheme
    - malfunction of the security
  - insider attack
  - Physical access to the chip
    - high knowledge of the circuit
    - very expensive equipment
BIST

- Reduced ATE cost
- In-situ testing
- Reduced external access

But

- Circuitry overhead
- Self-test of crypto-core
- Use the crypto-core as a test resource (TPG/SA)
- AES/DES
"Randomness" of cipher

- 1 vector per encryption

\[ \approx 1 \text{ vector every 10 clock cycles} \]
"Randomness" of cipher

- 1 vector per encryption

\[ \approx 1 \text{ vector every 10 clock cycles} \]
"Randomness" of cipher

- 1 vector per round cycle

"Randomness"? (Diffusion, Confusion, Bijection)

Checked by NIST statistical package suite (15 randomness tests)
AES round / DES round : as good random pattern generators as LFSRs
- Looped Crypto-core $\leftrightarrow$ random number generator

- First step
  - $1^{\text{st}}$ cycle
- Second step

- Cycles 2, 3, ……, n
Theoretical result

- AES: Fault-coverage = 100% after $n \in \{2520, \ldots, 2590\}$ clock cycles
- DES: Fault-coverage = 100% after $n \in \{620, \ldots, 710\}$ clock cycles

In practice

- AES
  - Fault-coverage = 100% after 2200 clock cycles ($\forall$ key, $\forall$ clear text)
- DES
  - Fault-coverage = 100% after 560 clock cycles ($\forall$ key (not wk), $\forall$ clear text)
- STUMPS Architecture

- Proposed solution

TPG for other cores
Test response compactor for other cores
TPG : ISCAS'89 benchmarks

- **s9234**

- **s13207**

- **s38548**
- Response compaction mode:
  - $SA = \text{Selection} = 1$

- Functional mode:
  - $SA = 0$
Fault-masking probability

- **AES/DES**

\[
P(M_n) = \frac{1}{2^m} - \left(\frac{1}{2^m}\right)^n
\]

\[
P(M_{128}) \xrightarrow{n \to \infty} \frac{1}{2^{128}} \approx 10^{-40}
\]

- **MISR**

\[
P(M_n) = \frac{2^{n-1} - 1}{2^{m+n-1} - 1}
\]

\[
P(M_{128}) \xrightarrow{n \to \infty} \frac{1}{2^{128}} \approx 10^{-40}
\]

\(n = \# \text{test responses and } m = 128 \text{ or } 64\)
Cryto-core (AES/DES) as a test resource:

😊 Test Fault Coverage: \( \cong \) LFSR

😊 Error Masking Probability: \( \cong \) MISR

😊 Reduced area overhead

😊 No impact on ciphering frequency/latency

😢 Potential attacks (2 successive round results observable))

⇒ use a specific key for test 😊
Simultaneous TPG and Compaction

Core 1  Core 2  Core 3

Core 4  Crypto processor AES  Core 5

Controller

Round Key generator

Sub Bytes
Shift Rows
Mix Columns
Add Round Key

Register R1

Register R2

SA

CUT responses

Key
Selection

Start

Round Key

End ciphering

Test vectors
## Area overhead

<table>
<thead>
<tr>
<th>Round</th>
<th>AES</th>
<th>AES generator</th>
<th>AES compactor</th>
<th>AES Self-test</th>
<th>AES 4 modes</th>
</tr>
</thead>
<tbody>
<tr>
<td>SubBytes</td>
<td>803 734</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>ShiftRows</td>
<td>0</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>MixColumns</td>
<td>59 847</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>AddRoundKey</td>
<td>49 945</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>Controler</td>
<td>6 345</td>
<td>+ 5.72%</td>
<td>+ 8.72%</td>
<td>+ 6.58%</td>
<td>+ 9.58%</td>
</tr>
<tr>
<td>Key generator</td>
<td>301 162</td>
<td>+ 0.015%</td>
<td>+ 0.015%</td>
<td>+ 0.015%</td>
<td>+ 0.015%</td>
</tr>
<tr>
<td>Glue logic</td>
<td>153 620</td>
<td>+ 0.04%</td>
<td>+ 17.95%</td>
<td>+ 0.04%</td>
<td>+ 18.36%</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>1 374 655</strong></td>
<td><strong>+0.03%</strong></td>
<td><strong>+2.05%</strong></td>
<td><strong>+0.04%</strong></td>
<td><strong>+2.10%</strong></td>
</tr>
</tbody>
</table>

**Overhead 2.1%**

Synthesis: VHDL + Synopsys Design Compiler
Technology: 0.35 um CMOS libraries (AMS)
## Area overhead

<table>
<thead>
<tr>
<th>Round</th>
<th>AES generator</th>
<th>AES compactor</th>
<th>AES Self-test</th>
<th>AES 4 modes</th>
</tr>
</thead>
<tbody>
<tr>
<td>SubBytes</td>
<td>803 734</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>ShiftRows</td>
<td>0</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>MixColumns</td>
<td>50 645</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>1 374 655</strong></td>
<td><strong>+0.03%</strong></td>
<td><strong>+2.05%</strong></td>
<td><strong>+0.04%</strong></td>
</tr>
</tbody>
</table>

**Overhead 2.1%**

For comparaison:

- Implementing a LFSR ⇒ 3.67%
- Implementing a BILBO ⇒ 7.64%

Synthesis: VHDL + Synopsys Design Compiler
Technology: 0.35 um CMOS libraries (AMS)
Special attention must be paid when testing secure circuits

- Scan–based designs
  - Counter–measures

- Bist (random test)
  - Self–test
  - Test resource
  - ECC
Publications

## SCAN

- [DATE'06] "Secure Scan Design" Design, Automation and Test in Europe, 2006
- [ETS'05] "Test Control for Secure Scan Designs" European Test Symposium, 2005
- [IOLTS'04] "Scan Design and Secure Chip" On–Line Testing Symposium, 2004

## BIST

References

- [Yan04]: B. Yang, K. Wu, R. Karri, Polytechnic University, "Scan-based Side-Channel Attack on Dedicated Hardware Implementations on Data Encryption Standard", International Test Conference (ITC 2004), Charlottes, USA, October 26–28, pp 339–344

- [Yan05]: B. Yang, K. Wu and R. Karri, Secure Scan: A Design-for-Test Architecture for Crypto Chips, Design Automation Conference (DAC 2005), Anaheim, July 12–14 pp 135–140, 2005

- [Yan, FDTC 05]: B. Yang & R. Karri, "Crypto BIST: A Built-In Self Test Architecture for Crypto Chips", 2nd Workshop on fault diagnosis and tolerance in cryptography (FDTC 2005), pp 95–108

- [NIST 800–22]: A statistical test suite for random and pseudorandom number generators for cryptographic applications NIST Special Publication 800–22 (with revisions dated May 15, 2001)