Scan chain encryption, a countermeasure against scan attacks
Mathieu da Silva, Giorgio Di Natale, Marie-Lise Flottes, Bruno Rouzeyre

To cite this version:

HAL Id: lirmm-01882565
https://hal-lirmm.ccsd.cnrs.fr/lirmm-01882565v2
Submitted on 10 Oct 2018

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.
SCAN CHAIN ENCRYPTION, A COUNTERMEASURE AGAINST SCAN ATTACKS

Mathieu Da Silva, Marie-Lise Flottes, Giorgio Di Natale, Bruno Rouzeyre

PHISIC 2018
Test of circuit is a mandatory step in IC production
Most popular method for Design-for-Test = Scan chains

- Replace original FF by Scan FF connected serially together
- Extra port « Scan-In » => controllability on internal states
- Extra port « Scan-Out » => observability on internal states
**Test standards**

- **IEEE 1149 (JTAG)** for board testing
- **IEEE 1500** for cores testing in a SoC
- **IEEE 1687 (IJTAG)** for embedded instruments
**CONTEXT**

- **Threats**

  - **Untrusted devices**
    - *Rosenfeld et al., Attacks and Defenses for JTAG, IEEE Design & Test 2010*

  - **Malicious users**
    - (example: scan attacks)
    - *Yang et al., Secure Scan: A Design-for-Test Architecture for Crypto Chips, TCAD’06*
1) Scan attacks

2) A new countermeasure: Scan chain encryption

3) Implementation with block cipher

4) Implementation with stream cipher

5) Conclusion
SCAN ATTACK PRINCIPLE

- Goal: Retrieve embedded secret data
- Exploit observability or controllability offered by scan chains
- Principle: switch between functional and scan modes
- Main target: secret key of crypto-processors (example: AES)
SCAN ATTACK ON AES

- Advanced Encryption Standard (AES)
  - Ciphertext after 10 rounds
  - Not secure after 1 round

- Attack pre-requisites
  - Attacker can switch between functional and test modes
  - Scan chain includes FFs of the round register

- Attack principle
  - Observation of the scan chain after 1 round

Yang et al., Secure Scan: A Design-for-Test Architecture for Crypto Chips, TCAD’06
24/05/2018
**DIFFERENTIAL ATTACK**

- Application of a first vector
  1) Reset
  2) Normal mode
    - 1 AES round
  3) Test mode
    - Scan out the round register content

---

*Yang et al., Secure Scan: A Design-for-Test Architecture for Crypto Chips, TCAD’06*

24/05/2018
Differential Attack

- Application of a second vector
  1) Reset
  2) Normal mode
     • 1 AES round
  3) Test mode
     • Scan out the round register content

Secret Key 128 bits

Key Generation

CipherText – 128 bits

ScanIn x

Other FFs

Round

ScanOut

Yang et al., Secure Scan: A Design-for-Test Architecture for Crypto Chips, TCAD’06
24/05/2018
DIFFERENTIAL ATTACK

- Hamming distance

\[ R \text{ xor} R' \]

- Attacker applies pairs of input values until hamming distance equal to specific values => key byte revealed

- On average, 32 trials

\[ 512 \text{ trials to retrieve the whole 128-bit key} \]
SUMMARY

1) Scan attacks

2) A new countermeasure: Scan chain encryption

3) Implementation with block cipher

4) Implementation with stream cipher

5) Conclusion
Solution: test communication encryption

- **Input decryption** prevents sending desired test data
- **Output encryption** prevents reading plain test responses
Solution: test communication encryption

- Input decryption prevents sending desired test data
- Output encryption prevents reading plain test responses
- Test/debug only possible by authorized user knowing the secret key
2 types of symmetric cipher: stream and block ciphers
STREAM CIPHER / BLOCK CIPHER

- **Stream cipher encryption**
  - Keystream XORed bitwise with the plaintext

- **Block cipher encryption**
  - Confusion and diffusion on a block of plaintext

- **Preference for stream ciphers**
  - "Naturally" adapted to serial test communication (JTAG, IEEE 1500, IJTAG)
  - Smaller area footprint compared to block ciphers
  - But ..
**Two-times pad: stream cipher requirement**

- **Two-times pad**: same key and IV re-used => same keystream generated to encrypt different data

⇒ Possible to carry out attacks if requirement is not fit

\[ R_1 \oplus S(IV, Key) \oplus R_2 \oplus S(IV, Key) \]

⇒ Solution: IV generated randomly at each circuit reset

\[ R_1 \oplus S(IV_1, Key) \oplus R_2 \oplus S'(IV_2, Key) \]
Assumption: original circuit embedded a crypto-core with its key management and storing

Scan chain encryption solution shares the key management and storing already implemented
SUMMARY

1) Scan attacks

2) A new countermeasure: Scan chain encryption

3) Implementation with block cipher

4) Implementation with stream cipher

5) Conclusion
**BLOCK CIPHER-BASED SCAN ENCRYPTION**

- Implementation on scan chain with 2 PRESENT block ciphers:
  - Lightweight (1 PRESENT = 2 139 GE)
  - Encryption by 64-bits block size
MODE OF OPERATIONS

- 64 bits encrypted every 32 clock cycles

\[ \Rightarrow \text{\#SFF} = P \times 64 \]

\[ \Rightarrow \text{No test time overhead on each pattern} \]
MODE OF OPERATIONS

- U bits = Unused bits

\[ S_1 \]

\[ R + U = 64 \text{ bits} \]

\[ R = \#SFF \mod 64 \]

\[ \text{U bits added} \]

\[ \Rightarrow \#SFF = P \times 64 + R \]

\[ \Rightarrow \text{Loss of U clock cycles per pattern} \]
SUMMARY

1) Scan attacks

2) A new countermeasure: Scan chain encryption

3) Implementation with block cipher

4) Implementation with stream cipher

5) Conclusion
STREAM CIPHER-BASED SCAN ENCRYPTION

- Implementation on JTAG:
  - 1 TRIVIUM stream cipher (2 016 GE)
  - TRNG to generate random IV
  - New instruction GetIV with a test data register IV

- Mode of operations in 2 phases: initialization and encryption
1) TRNG initialization: reach sufficient entropy to generate random number
INITIALIZATION PHASE

2) Shift IV in the dedicated Test Data Register
INITIALIZATION PHASE

3) Stream cipher setup

- Test Patterns
- Test Responses
- TRNG
- IV
- Key
- Keystream_so
- Keystream_si
- S_{in}
- S_{out}
- Scan chain
- IV
- IDCODE
- BYP
- IR
- TAP controller
- TCK
- TMS
- Off-Chip Encryption
- On-Chip Decryption
- Off-Chip Decryption
INITIALIZATION PHASE

Initialization phase finished => Encryption phase
**Encryption Phase**

- Send *GETIV* instruction

⇒ Shift the content of the IV register out the circuit
**ENCRIPTION PHASE**

- User can encrypt and decrypt test data with the **obtained IV** and the **shared secret key**
TIME FOR THE INITIALIZATION PROCESS

- $T_{TRNG\_init}$ to initialize the TRNG
- 80 clock cycles to shift the $IV$ in the register
- 1 152 clock cycles for the stream cipher setup

<table>
<thead>
<tr>
<th>Original circuit</th>
<th>Triple-DES</th>
<th>Pipelined AES-128</th>
<th>Pipelined AES-256</th>
<th>RSA 1024</th>
<th>LEON3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test time*</td>
<td>687 101</td>
<td>1 944 877</td>
<td>4 559 845</td>
<td>39 405 239</td>
<td>11 612 051</td>
</tr>
<tr>
<td>(clock cycles)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Test time overhead**

<table>
<thead>
<tr>
<th></th>
<th>Block-based solution (%)</th>
<th>Stream-based solution (%)**</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>+0.31</td>
<td>+0.18</td>
</tr>
<tr>
<td></td>
<td>+0.81</td>
<td>+0.06</td>
</tr>
<tr>
<td></td>
<td>+0.006</td>
<td>+0.03</td>
</tr>
<tr>
<td></td>
<td>+0.33</td>
<td>+0.003</td>
</tr>
<tr>
<td></td>
<td>+0.004</td>
<td>+0.01</td>
</tr>
</tbody>
</table>

*: Test time considered for a fault coverage of 100%, except for LEON3 where it reaches 70%
**: test time overhead without the initialization of the TRNG
1) Scan attacks

2) A new countermeasure: Scan chain encryption

3) Implementation with block cipher

4) Implementation with stream cipher

5) Conclusion
## Comparison Between Both Solutions

<table>
<thead>
<tr>
<th></th>
<th>Block cipher-based solution (PRESENT)</th>
<th>Stream cipher-based solution (TRIVIUM)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Security</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Scan attacks</td>
<td>Protected</td>
<td>Protected (two times pad not possible)</td>
</tr>
<tr>
<td>- Malicious core</td>
<td>Protected</td>
<td>Protected</td>
</tr>
<tr>
<td><strong>Cost</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Area</td>
<td>10 658.96 µm²</td>
<td>5 408.52 µm² (+ 31 200 µm² for TRNG)</td>
</tr>
<tr>
<td>- Test time</td>
<td>Depends on the scan length (multiple or not of the block size)</td>
<td>Clock cycles required for the initialization phase</td>
</tr>
<tr>
<td><strong>Integration</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Diagnosis &amp; debug</td>
<td>Still possible in-field</td>
<td></td>
</tr>
<tr>
<td>- Key management</td>
<td>Re-use key management already implemented</td>
<td></td>
</tr>
<tr>
<td>- Integration in test daisy-chain</td>
<td>Possible issue with the padding of test data</td>
<td>No issue</td>
</tr>
</tbody>
</table>
Thank You
ACKNOWLEDGEMENTS

- FUI#20 TEEVA Project

- Partners