Glitch-Resistant Masking Schemes as Countermeasure Against Fault Sensitivity Analysis

Victor Arribas, Thomas De Cnudde, Danilo Šijačić
IoT security
IoT security
Outline

- FSA
- Threshold Implementations
- The power of SCA glitch-resistance
- Experiments
- Results
Outline

- FSA
  - Threshold Implementations
  - The power of SCA glitch-resistance
  - Experiments
  - Results
FSA

Actively triggered passive attack

Critical path depends on the data

\[
T_A < T_B \Rightarrow T_C = \begin{cases} 
T_A + T_{AND} & \text{if } A = 0 \\
T_B + T_{AND} & \text{if } A = 1 
\end{cases}
\]

Source: [LSG+10]

XOR data-independent

FSA

Critical timing delay $\Rightarrow$ maximum timing delay

Source: [LSG+10]

FSA

Fault sensitivity $\Rightarrow$ Critical condition

FSA

Fault sensitivity ⇒ Critical condition

FSA

Fault sensitivity $\Rightarrow$ Critical condition

Clock glitch $\Rightarrow$ Our metric: propagation delay

FSA

Attack Implementation $\Rightarrow$ Two phases

1. Profiling

2. Key recovery

CPA
Outline

- FSA
- Threshold Implementations
- The power of SCA glitch-resistance
- Experiments
- Results
Threshold Implementations

Side-Channel Analysis (SCA) countermeasure

Provable security with minimal assumptions on the HW

Q.E.D.

Security in the presence of glitches
Threshold Implementations

Boolean masking scheme

Secret sharing and multi party computation techniques

- Correctness

- Non-Completeness

- Uniformity
Threshold Implementations

Threshold Implementations (TI) [NRR06]

\[ x_3 = x + x_1 + x_2 \]
Outline

- FSA
- Threshold Implementations
- The power of SCA glitch-resistance
- Experiments
- Results
Glitch-Resistance

\[ FS_1(x_{1,2}, y_1) \geq FS_2(x_{2,3}, y_2) \geq FS_3(x_{1,3}, y_1) \]
Glitch-Resistance

T₁

T₂

T₃
Glitch-Resistance
Glitch-Resistance

Share 1:

\[ T_A = \begin{cases} T_{y_1}^1 + T_{AND} & \text{(if } y_1 = '0') \\ T_{x_1}^1 + T_{AND} & \text{(if } y_1 = '1') \end{cases} \]

\[ T_B = \begin{cases} T_{y_2}^1 + T_{AND} & \text{(if } y_2 = '0') \\ T_{x_1}^2 + T_{AND} & \text{(if } y_2 = '1') \end{cases} \]

\[ T_C = \begin{cases} T_{y_1}^2 + T_{AND} & \text{(if } y_1 = '0') \\ T_{x_2}^1 + T_{AND} & \text{(if } y_1 = '1') \end{cases} \]
Outline

- FSA
- Threshold Implementations
- The power of SCA glitch-resistance
- Experiments
- Results
Experiments

ASIC

CASCADE [SBY+18]

FPGA

Xilinx tools

Targets: PRESENT and Keccak Sboxes

Experiments

Profiling: initial value $\Rightarrow 2^{2\cdot N}$

PRESENT:
ASIC $\Rightarrow 1011$
FPGA $\Rightarrow 0000$
ASIC $\Rightarrow 0x798$
FPGA $\Rightarrow 0x821$

NOTE: correlation differences
Experiments

Profiling: One input reference profile

Pick the best correlation profile
Experiments

Test circuit

\[ F S_g = HW(SBox^{-1}(CT \oplus K_g)) \]
Experiments

Test circuit

Profiling $\Rightarrow$ known sharing

Key recovery $\Rightarrow$ unshared ptxt-ciph
Experiments

Unrealistically strong adversary

- Detailed profiling
- Straightforward circuit
- Whole state of inputs covered
- Highest correlation initial value
- Unrealistic metric
Outline

- FSA
- Threshold Implementations
- The power of SCA glitch-resistance
- Experiments
- Results
Results

Poschmann et al. implementation of PRESENT [PMK+11]

Results

Poschmann et al. implementation of PRESENT [PMK+11]

Results

PRESENT:
Key = 12
Results

Keccak “Changing of the Guards” [Daemen17]

19 bits $\Rightarrow$ 15 bits Sbox

Uniform

Results

Keccak “Changing of the Guards”

15 bits ⇒ 15 bits Sbox

NOT Uniform

Results

Keccak:
Key = 29
Conclusions

- Glitch-resistant masking schemes provide FSA protection
- This protection is ensured by Non-completeness
- Unrealistically powerfull attacker
- Tests over simulations in ASIC and FPGA
Thank you!