Threshold Implementations
(Efficient TI on AES)

Begül Bilgin, Benedikt Gierlichs, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen
# Introduction

## Physical Attacks

<table>
<thead>
<tr>
<th></th>
<th>Active (Fault Attacks)</th>
<th>Passive (Observing Attacks)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-Invasive</td>
<td>Glitching, Temperature Change, Low Voltage, ...</td>
<td>Side Channel Attacks (Timing Analysis, Power Analysis, EM Attacks)</td>
</tr>
<tr>
<td>Semi-Invasive</td>
<td>Light Attacks, Radiation Attacks, ...</td>
<td>Optical Inspection, ...</td>
</tr>
<tr>
<td>Invasive</td>
<td>Laser cutters, Permanent circuit changes, ...</td>
<td>Probing, ...</td>
</tr>
</tbody>
</table>
Introduction
Power Analysis

Exploit information from the correlation between the instantaneous power consumption of the device and the intermediate results of the cryptographic algorithm.

- Simple Power Analysis (SPA)
- Differential Power Analysis (DPA)
  - Difference Of Means (DoM)
  - Correlation Power Analysis (CPA)
  - Templates
Introduction
DPA Countermeasures

- Circuit level
  - WDDL cells

- Algorithmic level
  - Introducing Noise (not provably secure)
    - random delays
    - dummy operations
  - Masking (provably secure)
  - Leakage resilient crypto (limits encryptions per key)
Introduction
Masking

\[ S(inp) = S(mask \oplus inp \oplus mask) \neq S(mask) \oplus S(inp \oplus mask) \]
Introduction

Masking

\[
S(\text{inp}) = S(\text{mask} \oplus \text{inp} \oplus \text{mask}) = S(\text{mask}) \\
S'(\text{inp} \oplus \text{mask}, \text{inp} \oplus \text{mask})
\]

\[
S(x,y,z) = x \oplus yz \\
= (x_1 \oplus x_2) \oplus (y_1 \oplus y_2) \oplus (z_1 \oplus z_2)
\]

\[
S(x_1,y_1,z_1) = x_1 \oplus y_1 z_1 \\
S'(x_1,x_2,y_1,y_2,z_1,z_2) = x_2 \oplus y_1 z_2 \oplus y_2 z_1 \oplus y_2 z_2
\]
Introduction

Masking

$S'(x_1,x_2,y_1,y_2,z_1,z_2) = x_1 \oplus y_1 z_1 \oplus y_1 z_2$

$S'(x_2,x_1,y_2,y_1,z_2,z_1) = x_2 \oplus y_2 z_2 \oplus y_2 z_1$

$S(x,y,z) = x \oplus yz$

$= (x_1 \oplus x_2) \oplus (y_1 \oplus y_2) (z_1 \oplus z_2)$
First-order masking
Introduction

Masking

Second-order masking
Introduction

Masking

✓ Proper randomness
✓ Functions leak independently
✓ Functions should not leak intermediate information depending on both inputs
× Not secure in CMOS because of glitches


**Introduction**

**Glitches**

\[ S(y,z) = yz = (y_1 \oplus y_2) (z_1 \oplus z_2) \]

\[ S(y_1,z_1) = y_1 z_1 \]

\[ S'(y_1,y_2,z_1,z_2) = y_1 z_2 \oplus y_2 z_1 \oplus y_2 z_2 \]

Assume \( y_2 \) arrives late

<table>
<thead>
<tr>
<th>( y_2 )</th>
<th>( z_1 )</th>
<th>( z_2 )</th>
<th># AND</th>
<th># XOR</th>
<th># TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 → 1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1 → 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0 → 1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>1 → 0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>0 → 1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1 → 0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>0 → 1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1 → 0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>
Threshold Implementations

Masking Scheme based on Secret Sharing and Multiparty Computation

Pros:
✓ Security in a circuit with glitches
✓ Efficient in HW
✓ Any HW technology

Cons:
✗ High order non-linear function are challenging

AES (8k), Present (3k), Noekeon, Keccak (30k)
roughly 3 times larger than unshared
Threshold Implementations

\[
S_1(x_1, y_1, z_1, \ldots) \rightarrow (a_1, b_1, c_1, \ldots)
\]

\[
S_2(x_2, y_2, z_2, \ldots) \rightarrow (a_2, b_2, c_2, \ldots)
\]

\[
\vdots
\]

\[
S_s(x_s, y_s, z_s, \ldots) \rightarrow (a_s, b_s, c_s, \ldots)
\]

3 properties
Threshold Implementations

\[ (x_1, y_1, z_1, \ldots) \rightarrow S_1 \rightarrow (a_1, b_1, c_1, \ldots) \]

\[ (x_2, y_2, z_2, \ldots) \rightarrow S_2 \rightarrow (a_2, b_2, c_2, \ldots) \]

\[ \ldots \]

\[ (x_s, y_s, z_s, \ldots) \rightarrow S_s \rightarrow (a_s, b_s, c_s, \ldots) \]

\[ = \]

\[ (x, y, z, \ldots) \]

\[ = \]

\[ (a, b, c, \ldots) \]

Correctness
Threshold Implementations

$S_1(x_1, y_1, z_1, ...)$
$\oplus$

$S_2(x_2, y_2, z_2, ...)$
$\oplus$

$S_s(x_s, y_s, z_s, ...)$
$\oplus$

$= (x, y, z, ...)$

$= (a, b, c, ...)$

Correctness, Non-completeness
Threshold Implementations

\[ S(x, y, z) = x \oplus yz = (x_1 \oplus x_2 \oplus x_3) \oplus (y_1 \oplus y_2 \oplus y_3) \oplus (z_1 \oplus z_2 \oplus z_3) \]

\[ S_1(x_2, x_3, y_2, y_3, z_2, z_3) = x_2 \oplus y_2 z_2 \oplus y_2 z_3 \oplus y_3 z_2 \]
\[ S_2(x_1, x_3, y_1, y_3, z_1, z_3) = x_3 \oplus y_3 z_3 \oplus y_3 z_1 \oplus y_1 z_3 \]
\[ S_3(x_1, x_2, y_1, y_2, z_1, z_2) = x_1 \oplus y_1 z_1 \oplus y_1 z_2 \oplus y_2 z_1 \]
Threshold Implementations

If the input masking is uniform and the circuit is non-complete, then the stochastic functions $S_i$ and $x$ are independent for any $i$.

If the input masking is uniform and the circuit is non-complete, then any single component function $S_i$ does not leak information on $x$.

Need at least $d+1$ shares for a function of degree $d$
Threshold Implementations

\[ (x_1, y_1, z_1, ...) \oplus (x_2, y_2, z_2, ...) \oplus \ldots \oplus (x_s, y_s, z_s, ...) = (x, y, z, ...), \]

\[ (a_1, b_1, c_1, ...) \oplus (a_2, b_2, c_2, ...) \oplus \ldots \oplus (a_s, b_s, c_s, ...) = (a, b, c, ...) \]

Correctness, Non-completeness, Uniformity
Threshold Implementations

Uniformity

A masking $X$ is uniform $\iff \exists$ a constant $p$ s.t. $\forall x$ we have:

- if $X \in \text{Sh}(x)$ then $\Pr(X|x) = p$,
- else $\Pr(X|x)=0$.

If the unshared function is a permutation, the shared function should also be a permutation.

If uniformity can not be achieved during $S_i$ calculation, apply re-masking.
Threshold Implementations
Decomposition

\[ S = G \circ F \]

Separate non-linear functions with registers
Applications

- All 3x3 and 4x4 S-boxes
- PRESENT: uses 4x4 S-box with degree 3
  - 3,3 kGE (1,1 kGE unprotected)
- KECCAK: uses 5x5 S-box with degree 2
  - 32,6 kGE (10,6 kGE unprotected)
- AES: uses 8x8 S-box with degree 7
  - by Moradi et al. and by us
- Authenticated Encryption designs FIDES and PRIMATEs
## Threshold Implementations

### 4x4 S-boxes

<table>
<thead>
<tr>
<th>remark</th>
<th>unshared</th>
</tr>
</thead>
<tbody>
<tr>
<td>affine</td>
<td>1</td>
</tr>
<tr>
<td>quadratic</td>
<td>6</td>
</tr>
<tr>
<td>cubic in $A_{16}$</td>
<td>30</td>
</tr>
<tr>
<td>cubic in $A_{16}$</td>
<td>114</td>
</tr>
<tr>
<td>cubic in $S_{16} \setminus A_{16}$</td>
<td>151</td>
</tr>
</tbody>
</table>
### Threshold Implementations

#### 4x4 S-boxes

<table>
<thead>
<tr>
<th>remark</th>
<th>unshared</th>
<th>3 shares</th>
<th>4 shares</th>
<th>5 shares</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>affine</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>quadratic</td>
<td>6</td>
<td>5</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>cubic in $A_{16}$</td>
<td>30</td>
<td>28</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>cubic in $A_{16}$</td>
<td>114</td>
<td>113</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>cubic in $S_{16} \setminus A_{16}$</td>
<td>151</td>
<td></td>
<td></td>
<td>4</td>
</tr>
</tbody>
</table>

Many S-boxes with good cryptographic properties
Threshold Implementations
4x4 S-boxes

<table>
<thead>
<tr>
<th>remark</th>
<th>unshared</th>
<th>3 shares</th>
<th>4 shares</th>
<th>5 shares</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>affine</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>quadratic</td>
<td>6</td>
<td>5</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>cubic in $A_{16}$</td>
<td>30</td>
<td>28</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>cubic in $A_{16}$</td>
<td>114</td>
<td>113</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>cubic in $S_{16} \setminus A_{16}$</td>
<td>151</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Many S-boxes with good cryptographic properties

GF($2^4$) inversion
Applications

- All 3x3 and 4x4 S-boxes
- PRESENT: uses 4x4 S-box with degree 3
  - 3,3 kGE (1,1 kGE unprotected)
- KECCAK: uses 5x5 S-box with degree 2
  - 32,6 kGE (10,6 kGE unprotected)
- AES: uses 8x8 S-box with degree 7
  - by Moradi et al. and by us
- Authenticated Encryption designs FIDES and PRIMATEs
TI on AES


- All operations on 3 shares
- 5 pipeline stages in S-box
- Tower field GF(2^2)
- Requires extra randomness (48 bits per S-box)
TI on AES


- IDEA: Adjust the number of shares as needed
- RESULT: Smaller area, less clock cycles, less extra randomness
- Data flow as in Moradi et al.
- Linear part: only 2 shares
- S-box: 2 to 5 shares
- Tower field GF(2^4)
TI on AES
S-box
TI on AES

S-box

5 shares
TI on AES

S-box

5 shares, 4 input 3 output shares
TI on AES

S-box

5 shares, 4 input 3 output shares, 2 shares
TI on AES

S-box

5 shares, 4 input 3 output shares, 2 shares, 4 shares
TI on AES

S-box

5 shares, 4 input 3 output shares, 2 shares, 4 shares, 3 shares
TI on AES

S-box

5 shares, 4 input 3 output shares, 2 shares, 4 shares, 3 shares

registers after every nonlinear function
TI on AES

S-box

5 shares, 4 input 3 output shares, 2 shares, 4 shares, 3 shares

registers after every nonlinear function
re-masking to change the number of shares
### TI on AES

#### Implementation Results

<table>
<thead>
<tr>
<th></th>
<th>State Array</th>
<th>Key Array</th>
<th>S-box</th>
<th>Mix Col.</th>
<th>Cont.</th>
<th>MUXes</th>
<th>Other</th>
<th>Total</th>
<th>cycles</th>
<th>rand bits **</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amir et al.</td>
<td>2529</td>
<td>2526</td>
<td>4244</td>
<td>1120</td>
<td>166</td>
<td>376</td>
<td>153</td>
<td>11114/11031</td>
<td>266</td>
<td>48</td>
</tr>
<tr>
<td>This paper</td>
<td>1698</td>
<td>1890</td>
<td>3708</td>
<td>770</td>
<td>221</td>
<td>746</td>
<td>69</td>
<td>9102</td>
<td>246</td>
<td>44</td>
</tr>
<tr>
<td>This paper*</td>
<td>1698</td>
<td>1890</td>
<td>3003</td>
<td>544</td>
<td>221</td>
<td>746</td>
<td>69</td>
<td>8171</td>
<td>246</td>
<td>44</td>
</tr>
</tbody>
</table>

* compile Ultra  
** per S-box

- Based on plain Canright S-box (233 GE)
- Based on plain Amir’s AES (2.4 GE)
- Keeping Hierarchy
TI on AES
Practical Security Evaluation

• Goals:
  1. Verify resistance against first order attacks
  2. Evaluate resistance against HO attacks

• Univariate attacks $\rightarrow$ shares are processed in parallel

• Adversary friendly conditions
  1. PRNG not active during TI-AES $\rightarrow$ less noise
  2. Well alignment
  3. Adversary knows the implementation (masks unknown)
TI on AES
Practical Security Evaluation

- PRNG off, first order CPA, HD model at S-box output
- Highest peak 3 cycles later, input MC
TI on AES
Practical Security Evaluation

- PRNG off, first order correlation collision attack
TI on AES
Practical Security Evaluation

- PRNG on, first order CPA / correlation collision attack
- 10 million traces
TI on AES
Practical Security Evaluation

- PRNG on, second order CPA
- HD model at S-box output
TI on AES
Practical Security Evaluation

- PRNG on, second order correlation collision attack
TI on AES
Practical Security Evaluation

- **Goal 1: verify resistance against first order attacks**
  - Evaluation limited by number of traces
  - 10 million traces

- **Goal 2: evaluate resistance against HO attacks**
  - Most trace-efficient second order attack requires 600k tr
  - Second Order attacks: Number of traces scales quadratically in the noise standard deviation (we had little noise)
Conclusion

- TI is provably secure against first order DPA
- TI can be efficient
- Room for improvement:
  - Solutions to uniformity problems
  - Security against higher order DPA
- Consider countermeasures during design process
Thank You!