More Efficient Private Circuits II
Through Threshold Implementations

Thomas De Cnudde
Svetla Nikova
Investigating a novel approach towards a hardware implementation resisting combined SCA and FAs
Countermeasures for SCA and FA are generally researched separately

SCA Countermeasures  FA Countermeasures
Countermeasures for SCA and FA are generally researched separately

SCA Countermeasures

FA Countermeasures

Masking
Countermeasures for SCA and FA are generally researched separately

SCA Countermeasures

- Masking

FA Countermeasures

- Hiding
Countermeasures for SCA and FA are generally researched separately

SCA Countermeasures: Masking, Private Circuits

FA Countermeasures: Hiding
Countermeasures for SCA and FA are generally researched separately

SCA Countermeasures

- Masking
  - Private Circuits
  - Threshold Implementations

- Hiding

FA Countermeasures
Countermeasures for SCA and FA are generally researched separately

SCA Countermeasures

- Masking
  - Private Circuits
  - Threshold
  - Implementations
- Hiding

FA Countermeasures
Countermeasures for SCA and FA are generally researched separately

**SCA Countermeasures**
- Masking
  - Private Circuits
  - Threshold Implementations
  - ...
- Hiding

**FA Countermeasures**
- Temporal or Spatial redundancy
Countermeasures for SCA and FA are generally researched separately

SCA Countermeasures

- Masking
  - Private Circuits
  - Threshold Implementations
  ... 
- Hiding

FA Countermeasures

- Temporal or Spatial redundancy
- Error correction/detection
Countermeasures for SCA and FA are generally researched separately

**SCA Countermeasures**
- Masking
  - Private Circuits
  - Threshold
  - Implementations
  - ...
- Hiding

**FA Countermeasures**
- Temporal or Spatial redundancy
- Error correction/detection
- Infective computing
Countermeasures for SCA and FA are generally researched separately

SCA Countermeasures

- Masking
- Private Circuits
- Threshold Implementations
- Hiding

FA Countermeasures

- Temporal or Spatial redundancy
- Error correction/detection
- Infective computing
- Sensors
Private Circuits II provides resistance against combined SCA and FA

SCA Countermeasures

- Masking
  - Private Circuits
  - Threshold Implementations
    ...
- Hiding

FA Countermeasures

- Temporal or Spatial redundancy
- Error correction/detection
- Infective computing
- Sensors
Applying Private Circuits II requires a series of transformation

PRESENT
Applying Private Circuits II requires a series of transformation

PRESENT

obtaining SCA resistance

Private Circuits
Applying Private Circuits II requires a series of transformation

PRESENT

obtaining SCA resistance

Private Circuits

obtaining combined SCA and FA resistance

Private Circuits II
Private Circuits and Threshold Implementations are closely related
Private Circuits and Threshold Implementations are closely related.

Private Circuits ↔ Threshold Implementations

Glitches not allowed ↔ Glitches allowed
Combined SCA and FA resistance for the PRESENT block cipher

PRESENT

Private Circuits

Threshold Implementations

Private Circuits II
Combined SCA and FA resistance for the PRESENT block cipher
Combined SCA and FA resistance for the PRESENT block cipher

Which approach is more efficient in HW?
PC and TI are both boolean masking schemes

Input Encoding

\[ I_1 = \text{Input} + R_1 + R_2 \]
\[ I_2 = R_1 \]
\[ I_3 = R_2 \]
PC and TI are both boolean masking schemes

**Input Encoding**

\[ \begin{align*}
I_1 &= \text{Input} + R_1 + R_2 \\
I_2 &= R_1 \\
I_3 &= R_2
\end{align*} \]
PC and TI are both boolean masking schemes

Input Encoding

\[ I_1 = \text{Input} + R_1 + R_2 \]
\[ I_2 = R_1 \]
\[ I_3 = R_2 \]

Output Decoding

\[ O_1 + O_2 + O_3 = \text{Output} \]
PC and TI are both boolean masking schemes

Input Encoding
\[ I_1 = \text{Input} + R_1 + R_2 \]
\[ I_2 = R_1 \]
\[ I_3 = R_2 \]

Output Decoding
\[ O_1 + O_2 + O_3 = \text{Output} \]

Linear operations are performed on individual shares
PC and TI differ in the nonlinear operations
PC and TI differ in the nonlinear operations

Private Circuits:
1) Generate $R_{1,2} \ R_{1,3} \ R_{2,3}$
2) Compute $R_{2,1} = R_{1,2} + a_1b_2$
   $R_{3,1} = R_{1,3} + a_1b_3$
3) Compute $c_1 = a_1b_1 + R_{1,2} + R_{1,3}$
   $c_2 = a_2b_2 + R_{2,1} + a_2b_1 + R_{2,3}$
   $c_3 = a_3b_2 + R_{3,1} + a_3b_1 + R_{2,1} + a_2b_1$
PC and TI differ in the nonlinear operations

Private Circuits:
1) Generate $R_{1,2} R_{1,3} R_{2,3}$
2) Compute $R_{2,1} = R_{1,2} + a_1 b_2$
   $R_{3,1} = R_{1,3} + a_1 b_3$
3) Compute $c_1 = a_1 b_1 + R_{1,2} + R_{1,3}$
   $c_2 = a_2 b_2 + R_{2,1} + a_2 b_1 + R_{2,3}$
   $c_3 = a_3 b_2 + R_{3,1} + a_3 b_1 + R_{2,1} + a_2 b_1$
PC and TI differ in the nonlinear operations
PC and TI differ in the nonlinear operations

Threshold Implementations: 

\[ c_1 = a_2b_2 + a_1b_2 + a_2b_1 \]
\[ c_2 = a_3b_3 + a_3b_2 + a_2b_3 \]
\[ c_3 = a_1b_1 + a_1b_3 + a_3b_1 \]
PC and TI differ in the nonlinear operations

Threshold Implementations:

\[
c_1 = a_2b_2 + a_1b_2 + a_2b_1 \\
c_2 = a_3b_3 + a_3b_2 + a_2b_3 \\
c_3 = a_1b_1 + a_1b_3 + a_3b_1
\]
Implementing PRESENT S-box with Private Circuits

<table>
<thead>
<tr>
<th>x</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>S(x)</td>
<td>C</td>
<td>5</td>
<td>6</td>
<td>B</td>
<td>9</td>
<td>0</td>
<td>A</td>
<td>D</td>
<td>3</td>
<td>E</td>
<td>F</td>
<td>8</td>
<td>4</td>
<td>7</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>
Implementing PRESENT S-box with Private Circuits

<table>
<thead>
<tr>
<th>x</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>S(x)</td>
<td>C</td>
<td>5</td>
<td>6</td>
<td>B</td>
<td>9</td>
<td>0</td>
<td>A</td>
<td>D</td>
<td>3</td>
<td>E</td>
<td>F</td>
<td>8</td>
<td>4</td>
<td>7</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>G(x)</td>
<td>7</td>
<td>E</td>
<td>9</td>
<td>2</td>
<td>B</td>
<td>0</td>
<td>4</td>
<td>D</td>
<td>5</td>
<td>C</td>
<td>A</td>
<td>1</td>
<td>8</td>
<td>3</td>
<td>6</td>
<td>F</td>
</tr>
<tr>
<td>F(x)</td>
<td>0</td>
<td>8</td>
<td>B</td>
<td>7</td>
<td>A</td>
<td>3</td>
<td>1</td>
<td>C</td>
<td>4</td>
<td>6</td>
<td>F</td>
<td>9</td>
<td>E</td>
<td>D</td>
<td>5</td>
<td>2</td>
</tr>
</tbody>
</table>
Implementing PRESENT S-box with Private Circuits

<table>
<thead>
<tr>
<th>x</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>S(x)</td>
<td>C</td>
<td>5</td>
<td>6</td>
<td>B</td>
<td>9</td>
<td>0</td>
<td>A</td>
<td>D</td>
<td>3</td>
<td>E</td>
<td>F</td>
<td>8</td>
<td>4</td>
<td>7</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>G(x)</td>
<td>7</td>
<td>E</td>
<td>9</td>
<td>2</td>
<td>B</td>
<td>0</td>
<td>4</td>
<td>D</td>
<td>5</td>
<td>C</td>
<td>A</td>
<td>1</td>
<td>8</td>
<td>3</td>
<td>6</td>
<td>F</td>
</tr>
</tbody>
</table>

23 AND gadgets
23 XOR gadgets
Implementing PRESENT S-box with Private Circuits

<table>
<thead>
<tr>
<th>x</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>S(x)</td>
<td>C</td>
<td>5</td>
<td>6</td>
<td>B</td>
<td>9</td>
<td>0</td>
<td>A</td>
<td>D</td>
<td>3</td>
<td>E</td>
<td>F</td>
<td>8</td>
<td>4</td>
<td>7</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>G(x)</td>
<td>7</td>
<td>E</td>
<td>9</td>
<td>2</td>
<td>B</td>
<td>0</td>
<td>4</td>
<td>D</td>
<td>5</td>
<td>C</td>
<td>A</td>
<td>1</td>
<td>8</td>
<td>3</td>
<td>6</td>
<td>F</td>
</tr>
<tr>
<td>F(x)</td>
<td>0</td>
<td>8</td>
<td>B</td>
<td>7</td>
<td>A</td>
<td>3</td>
<td>1</td>
<td>C</td>
<td>4</td>
<td>6</td>
<td>F</td>
<td>9</td>
<td>E</td>
<td>D</td>
<td>5</td>
<td>2</td>
</tr>
</tbody>
</table>

23 AND gadgets
23 XOR gadgets
9 AND gadgets
19 XOR gadgets
Implementing PRESENT S-box with Threshold Implementations

(Poschmann, 2011)
PC and TI achieve equivalent security with 25 Million traces
PC and TI achieve equivalent security with 25 Million traces

PC

1st Order

TI

2nd Order
Threshold Implementations is less costly in all aspects

<table>
<thead>
<tr>
<th></th>
<th>PC-I</th>
<th>TI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Slices</td>
<td>107</td>
<td>29</td>
</tr>
<tr>
<td>Number of Slice Flip Flops</td>
<td>166</td>
<td>48</td>
</tr>
<tr>
<td>Number of 4 input LUTs</td>
<td>96</td>
<td>57</td>
</tr>
<tr>
<td>Consumed Random Bits</td>
<td>28</td>
<td>0</td>
</tr>
<tr>
<td>Number of Clock Cycles</td>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>
PRESENT-TI achieves its security with 100 Million traces
PRESENT-TI achieves its security with 100 Million traces
Combined SCA and FA resistance for the PRESENT block cipher
Combined SCA and FA resistance for the PRESENT block cipher

PRESENT

Private Circuits

Threshold Implementations

Private Circuits II
Combined SCA and FA resistance for the PRESENT block cipher
Private Circuits II comes in two variants of FA resistance

1) Resisting any number of reset-only wire faults

2) Resisting t arbitrary wire faults
Private Circuits II comes in two variants of FA resistance

1) Resisting any number of reset-only wire faults

2) Resisting $t$ arbitrary wire faults $\quad t = 1$
Tamper resistance against any number of reset-only wire faults

SCA Resistant Circuit
Tamper resistance against any number of reset-only wire faults

SCA Resistant Circuit

Manchester Encoding

0 = (0,1)
1 = (1,0)
Tamper resistance against any number of reset-only wire faults

SCA Resistant Circuit

→

Manchester Encoding

0 = (0,1)  
1 = (1,0)

Gadgets Encoding and Error Cascading

propagates invalid 00
Tamper resistance against any number of reset-only wire faults

SCA Resistant Circuit

→

Manchester Encoding

0 = (0,1)  
1 = (1,0)

Gadgets Encoding and Error Cascading

propagates invalid 00
Tamper resistance against any number of reset-only wire faults

SCA Resistant Circuit

→

Manchester Encoding

0 = (0,1)
1 = (1,0)

→

Gadgets Encoding and Error Cascading

propagates invalid 00

→

Output Decoding

(0,1) = 0
(1,0) = 1
Tamper resistance against one arbitrary wire faults

SCA Resistant Circuit
Tamper resistance against one arbitrary wire faults

SCA Resistant Circuit

\[ 0 = (0,0) \]
\[ 1 = (1,1) \]
Tamper resistance against one arbitrary wire faults

SCA Resistant Circuit

Repetition Encoding

0 = (0,0)
1 = (1,1)

Gadgets Encoding and Error Cascading

propagates invalid 01
Tamper resistance against one arbitrary wire faults

SCA Resistant Circuit

\[ 0 = (0,0) \]
\[ 1 = (1,1) \]

Repetition Encoding

Gadgets Encoding and Error Cascading

OR of ANDs form

propagates invalid 01
Tamper resistance against one arbitrary wire faults

SCA Resistant Circuit

Repetition Encoding

0 = (0,0)
1 = (1,1)

Gadgets Encoding and Error Cascading

propagates invalid 01

OR of ANDs form

NOT gate is reversible

fault at output propagates to the input
Tamper resistance against one arbitrary wire faults

SCA Resistant Circuit

Repetition Encoding
\[0 = (0,0)\]
\[1 = (1,1)\]

Gadgets Encoding and Error Cascading

Output Decoding
\[(0,0) = 0\]
\[(1,1) = 1\]

OR of ANDs form

NOT gate is reversible
fault at output propagates to the input
For SCA resistance only the data dependent values need to be masked.

```plaintext
generateRoundKeys()
for i = 1 to 31 do
    addRoundKey(STATE, K_i)
    sBoxLayer(STATE)
    pLayer(STATE)
end for
addRoundKey(STATE, K_{32})
```
With FA, control signals can be the target of Fault Injection

Fault on ready signal can reveal all intermediate results
With FA, control signals can be the target of Fault Injection

Fault on ready signal reveals all intermediate results
All possible signals need to be encoded with PC II
All possible signals need to be encoded with PC II

Adders
All possible signals need to be encoded with PC II

Adders
Comparators
All possible signals need to be encoded with PC II

Adders
Comparators
Multiplexers
PC II effectively handles the injected fault on the ready signal

Fault on ready signal reveals no information on the intermediate results
Applying PC II results in a significant increase in area

<table>
<thead>
<tr>
<th></th>
<th>TI</th>
<th>TI + PC-II Reset-Only</th>
<th>TI + PC-II General Attack (t = 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Slices</td>
<td>696</td>
<td>6125</td>
<td>6125</td>
</tr>
<tr>
<td>Number of Slice Flip Flops</td>
<td>647</td>
<td>1292</td>
<td>1292</td>
</tr>
<tr>
<td>Number of 4 input LUTs</td>
<td>1356</td>
<td>10341</td>
<td>10341</td>
</tr>
<tr>
<td>Consumed Random Bits</td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Number of Clock Cycles</td>
<td></td>
<td></td>
<td>578</td>
</tr>
</tbody>
</table>
Applying PC II results in a significant increase in area

<table>
<thead>
<tr>
<th></th>
<th>TI</th>
<th>TI + PC-II Reset-Only</th>
<th>TI + PC-II General Attack (t = 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Slices</td>
<td>696</td>
<td>6125</td>
<td>6125</td>
</tr>
<tr>
<td>Number of Slice Flip Flops</td>
<td>647</td>
<td>1292</td>
<td>1292</td>
</tr>
<tr>
<td>Number of 4 input LUTs</td>
<td>1356</td>
<td>10341</td>
<td>10341</td>
</tr>
<tr>
<td>Consumed Random Bits</td>
<td></td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Number of Clock Cycles</td>
<td></td>
<td>578</td>
<td></td>
</tr>
</tbody>
</table>

Result of use of LUTs vs atomic gates
Applying PC II results in a significant increase in area

<table>
<thead>
<tr>
<th></th>
<th>TI</th>
<th>TI + PC-II Reset-Only</th>
<th>TI + PC-II General Attack (t = 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Slices</td>
<td>696</td>
<td>6125</td>
<td>6125</td>
</tr>
<tr>
<td>Number of Slice Flip Flops</td>
<td>647</td>
<td>1292</td>
<td>1292</td>
</tr>
<tr>
<td>Number of 4 input LUTs</td>
<td>1356</td>
<td>10341</td>
<td>10341</td>
</tr>
<tr>
<td>Consumed Random Bits</td>
<td></td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Number of Clock Cycles</td>
<td></td>
<td>578</td>
<td></td>
</tr>
</tbody>
</table>

Result of use of LUTs 4 input function vs atomic gates

Can be reduced when care is applied
Applying PC II results in a significant increase in area

Future work can improve the area cost for our FPGA implementations

1. Packing gates in LUT while satisfying the OR of AND structure
Applying PC II results in a significant increase in area

Future work can improve the area cost for our FPGA implementations

1. Packing gates in LUT while satisfying the OR of AND structure

2. Move implementations to larger FPGAs and launch combined attacks
Applying PC II results in a significant increase in area

Future work can improve the area cost for our FPGA implementations

1. Packing gates in LUT while satisfying the OR of AND structure

2. Move implementations to larger FPGAs and launch combined attacks

3. Circuits with randomness consumption
Thank you

Questions ?
Error Cascading Stage is nonlinear
Error Cascading Stage is nonlinear

\[
S_{1, EC, 1} = (s_{1, 1} \overline{s_{1, 0}} s_{2, 1} s_{2, 0}) \oplus (s_{1, 1} \overline{s_{1, 0}} s_{2, 1} s_{2, 0}) \\
S_{1, EC, 0} = (\overline{s_{1, 1}} s_{1, 0} \overline{s_{2, 1}} s_{2, 0}) \oplus (\overline{s_{1, 1}} s_{1, 0} s_{2, 1} \overline{s_{2, 0}}) \\
S_{2, EC, 1} = (\overline{s_{1, 1}} s_{1, 0} s_{2, 1} \overline{s_{2, 0}}) \oplus (s_{1, 1} \overline{s_{1, 0}} s_{2, 1} \overline{s_{2, 0}}) \\
S_{2, EC, 0} = (\overline{s_{1, 1}} s_{1, 0} \overline{s_{2, 1}} s_{2, 0}) \oplus (s_{1, 1} \overline{s_{1, 0}} s_{2, 1} s_{2, 0})
\]
Error Cascading Stage is nonlinear

Non-completeness is broken!

\[
\begin{align*}
S_{1,E,C,1} &= (s_{1,1}s_{1,0}^{\top}s_{2,1}s_{2,0}) \oplus (s_{1,1}^{\top}s_{1,0}s_{2,1}^{\top}s_{2,0}) \\
S_{1,E,C,0} &= (s_{1,1}^{\top}s_{1,0}s_{2,1}s_{2,0}) \oplus (s_{1,1}s_{1,0}s_{2,1}^{\top}s_{2,0}) \\
S_{2,E,C,1} &= (s_{1,1}s_{1,0}s_{2,1}^{\top}s_{2,0}) \oplus (s_{1,1}s_{1,0}s_{2,1}s_{2,0}) \\
S_{2,E,C,0} &= (s_{1,1}s_{1,0}s_{2,1}s_{2,0}) \oplus (s_{1,1}s_{1,0}s_{2,1}s_{2,0})
\end{align*}
\]