Comparison of Side-Channel countermeasures

Sonia Belaïd, Vincent Grosso, Romain Poussier, François-Xavier Standaert.

Journées C2, 27th March, 2014
Outline

Background

Results

SPRF
Outline

Background

Results

SPRF
The adversary uses public information that could be computed by an oracle who knows the secret.
Physical model

The adversary can use additional information due to computation of secret operations.
Side Channel Attack

- A probability for each subkey $k_i^*$, according to the leakages $l$, and public information $p$.

\[ Pr[K_i = k_i^* | L = l, P = p] \]

- Recover the full key $K = (k_0, \ldots, k_{15})$, according to the previous probability (some enumeration could be done).

- A Side Channel Attack is evaluated according to the enumeration required to recover the key in function of the data complexity.
Masking (a.k.a. secret sharing)

For unprotected device basic computations depend on the secret.
Masking (a.k.a. secret sharing)

When masking is implemented basic computations are independent of the secret.
Masking (a.k.a. secret sharing)

\[ p_i \rightarrow S_i \rightarrow S_i(p_i \oplus k_i) \]

\[ p_i \rightarrow C_i^1 \rightarrow x_i^1 \]
\[ m_i \rightarrow C_i^2 \rightarrow x_i^2 \]

The number of measurements required for an attack grows up exponentially in the number of shares with noise as basis.
Re-keying (a.k.a. leakage resilient)

- Limit the number of use of a key ($n$).
- The number of observations is limited.
- The information leaked on secret is limited.
PRG

\[
\begin{array}{c}
k_1 \rightarrow E \\
\end{array}
\]
PRG

\[ \text{PRG} \]

Diagram:

- Input: \( p_1 \)
- Input: \( k_1 \)
- Input: \( c_1 \)
- Output: \( E \)
PRG

\[ E(k_1, p_1, p_2, c_1, c_2, p_{n-1}, c_{n-1}) \]
PRG

\[ p_1 p_2 \cdots p_{n-1} \]

\[ k_1 \]

\[ E \]

\[ c_1 c_2 \cdots c_{n-1} \]
$$PRG$$

\[ p_1 \ p_2 \ldots \ p_{n-1} \]

\[ E \]

\[ C_1 \ C_2 \ldots \ C_{n-1} \]

\[ k_1 \quad E \quad k_2 \]
PRG

\[
\begin{align*}
&\text{Input:} \quad p_1, p_2, \ldots, p_{n-1} \\
&\text{Output:} \quad c_1, c_2, \ldots, c_{n-1}
\end{align*}
\]
The construction limits the number of measurements for 1 key.
Efficient \( \frac{n-1}{n} \) useful evaluation.
But 2 parties need to be synchronized (stateful).
PRF

\[ E^{k_0}(x) \]

\[ E^{k_1}(x) \]

\[ \ldots \]

\[ E^{k_{16}}(x) \]
The 2 parties had shared different input $p_i$ that are indexed by a part of the input of the PRF $x_j$. 

PRF
PRF

\[ E^k \]

\[ E^{k^0} p_{x_1} \]

\[ E^{k^1} p_{x_2} \]
PRF

\[ E_{k^0}(p_{x0}) \]

\[ E_{k^1}(p_{x1}) \]

\[ k \]

\[ k^0 \]

\[ k^1 \]

\[ k^{15} \]

\[ p_{x1} \]

\[ p_{x2} \]

\[ \ldots \]
PRF

\[ E_{k^0}(x) \]

\[ E_{k^1}(x) \]

\[ E_{k^{15}}(x) \]

\[ p_{x_1} \]

\[ p_{x_2} \]

\[ p_{x_{16}} \]

\[ \ldots \]
PRF

\[ k \rightarrow \begin{array}{c}
E \\
k^0 \\
p_{x_1}
\end{array}
\]

\[ k^1 \rightarrow \begin{array}{c}
E \\
p_{x_2}
\end{array}
\]

\[ k^{15} \rightarrow \begin{array}{c}
E \\
p_{x_{16}}
\end{array}
\]

\[ k^{16} \rightarrow \begin{array}{c}
\vdots
\end{array}
\]
PRF

\[ PRF_k(x_1) = E_k^0(p_{x_1}) \]

\[ PRF_k(x_2) = E_k^1(p_{x_2}) \]

\[ \ldots \]

\[ PRF_k(x_{16}) = E_k^{15}(p_{x_{16}}) \]
PRF

\[
\begin{align*}
&F_k(x) \\
&k^{16} \\
&k^1 \\
&k^0 \\
&p_{x_1} \\
&p_{x_2} \\
&\ldots
\end{align*}
\]
PRF

\[
\begin{align*}
F_k(x) &= F_k(x) \\
F_k(x) &= E_{k_1}(p_{x_1}) \\
F_k(x) &= E_{k_0}(p_{x_2}) \\
F_k(x) &= E_{k_{15}}(p_{x_{16}}) \\
F_k(x) &= E_{k_{16}}(p) \\
\end{align*}
\]
The 2 parties don’t need to be synchronized (stateless).
But only the data complexity is limited not the number of measurements.
Each node had at most $n$ sons.
Expensive: depth of the tree call for one PRF evaluation.
Simulated leaking implementations

For software, the operations are done sequentially. At each time, the trace depends only on one share.
Simulated leaking implementations

For hardware, the operations are done in parallel. The leakages on different shares are at the time in the trace.
Outline

Background

Results

SPRF
Security graph [VCGS13]

- x-axis corresponds to the number of measurements.
- y-axis corresponds to the rank of the key (log\(_2\) scale).
- E=AES.
- Template attacks, with perfect leakage function.
- Several experiments for a number of measurements.
- Maximal number of measurements for a fixed enumeration.
Hardware case, with an SNR of 0.2.
For different security level (i.e. the enumeration to perform) we can extract different number of measurements, before re-keying:-)
The masking does not improve the bounds, with respect to the cost of masking.
In that case the SNR is $\infty$. Indeed, the noise can be removed by performing again and again the same measurement.
No bound could be reached :-(

**PRF**
The masking increase the number of measurements required to eliminate the noise.
The parallelize of hardware implementation could be used to realize noise key dependent [MSJ12].

Use the same byte of plaintext, in the tree construction \( p_{x_1} = (x_1, \ldots, x_1) \).

\[
L = \sum_i S(x_1 \oplus k_i).
\]

Even if an adversary recover the key she still needs to perform an enumeration to recover the order for 16 subkey \( 2^{44} \).
Hardware PRF + tweak

The data complexity is limited to 256.
Hardware PRF + tweak

The physical noise can be averaged, but the key dependent noise is still present (improve the bound).
Hardware PRF + tweak

A bound could be reached :-)
Outline

Background

Results

SPRF
Shuffling
Shuffling

![Graph showing shuffling with time on the x-axis and resource on the y-axis, with a single data point marked as 'u'.]
Shuffling
Shuffling

![Graph showing shuffling results with time on the x-axis and resource on the y-axis. The graph has points marked at specific times with resource values.]
Shuffling
Shuffling
Shuffling

resource

0 1 2 3 4 5 6 7

A B C D E F

time

p

0 1 2 3 4 5 6 7 8 9 A B C D E F

u u u u u u u u u u u u u u u

p

Shuffling

```
<table>
<thead>
<tr>
<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>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<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>
</tr>
</thead>
<tbody>
<tr>
<td>p</td>
<td>p</td>
<td></td>
<td></td>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<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>
</tr>
</thead>
<tbody>
<tr>
<td>u</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```
Shuffling
PRF shuffling case

- The shuffling emulates the hardware.
- Some leakage on the permutation used could be additionally obtained by the adversary.
The data complexity is limited to 256. But averaging is not the best option to attack shuffling, hence we consider number of measurements.
The leakage on the permutation give some additional information to reduce the enumeration part.
SPRF + tweak

A bound could be reached :-)
Conclusion

- For PRG bounds are easy to reach.
- For PRF no bound is reachable.
  - Masking increase the number of measurements.
  - Tweak in hardware allows to reach bound.
  - Shuffling + tweak allow to reach bound in software.
References I
