Parallel Implementations of Masking Schemes and the Bounded Moment Leakage Model

G. Barthe, F. Dupressoir, S. Faust, B. Grégoire, F.-X. Standaert, P.-Y. Strub
IMDEA (Spain), Univ. Surrey (UK), Univ. Bochum (Germany), INRIA Sophia-Antipolis (France), UCL (Belgium), Ecole Polytechnique (France)

EUROCRYPT 2017, Paris, France
Outline

• Introduction / motivation
  • Side-channel attacks and noise
  • Masking and leakage models

• Bounded moment model
  • Masking intuition & BMM definition
  • Probing security ⇒ BM security

• Parallel multiplication (& refreshing)

• BM security ⇓ probing security
  • Inner product masking (with “non-mixing” leakages)
  • Continuous security & refreshing gadgets

• Conclusions
Outline

• Introduction / motivation
  • Side-channel attacks and noise
  • Masking and leakage models

• Bounded moment model
  • Masking intuition & BMM definition
  • Probing security ⇒ BM security

• Parallel multiplication (& refreshing)

• BM security ≠ probing security
  • Inner product masking (with “non-mixing” leakages)
  • Continuous security & refreshing gadgets

• Conclusions
Side-channel attacks

- ≈ physical attacks that decreases security exponentially in the # of measurements
Noise (hardware countermeasures)
Noise (hardware countermeasures)
Additive noise $\approx \text{cost} \times 2 \Rightarrow \text{security} \times 2 \Rightarrow$ not a good (crypto) security parameter
• Introduction / motivation
  • Side-channel attacks and noise
  • Masking and leakage models

• Bounded moment model
  • Masking intuition & BMM definition
  • Probing security $\Rightarrow$ BM security

• Parallel multiplication (& refreshing)
• BM security $\not\Rightarrow$ probing security
  • Inner product masking (with “non-mixing” leakages)
  • Continuous security & refreshing gadgets

• Conclusions
• Example: Boolean encoding

\[ y = y_1 \oplus y_2 \oplus \cdots \oplus y_{d-1} \oplus y_d \]

• With \( y_1, y_2, \ldots, y_{d-2}, y_{d-1} \leftarrow \{0,1\}^n \)
- Probing security (Ishai, Sahai, Wagner 2003)

\[ y = y_1 \oplus y_2 \oplus \cdots \oplus y_{d-1} \oplus y_d \]
Probing security (Ishai, Sahai, Wagner 2003)

\[ y = y_1 \oplus y_2 \oplus \cdots \oplus y_{d-1} \oplus y_d \]

\[ d - 1 \] probes do not reveal anything on \( y \)
• Probing security (Ishai, Sahai, Wagner 2003)

\[ y = y_1 \oplus y_2 \oplus \cdots \oplus y_{d-1} \oplus y_d \]

• But \( d \) probes completely reveal \( y \)
• Probing security (Ishai, Sahai, Wagner 2003)

\[ y = y_1 \oplus y_2 \oplus \cdots \oplus y_{d-1} \oplus y_d \]

- Bounded information leakage $\text{MI}(Y_i; L)^d$

(a) serial implementation.
• Probing security (Ishai, Sahai, Wagner 2003)

\[ y = y_1 \oplus y_2 \oplus \cdots \oplus y_{d-1} \oplus y_d \]

• Noisy leakage security (Prouff, Rivain 2013)
• Probing security (Ishai, Sahai, Wagner 2003)

\[ y = y_1 \oplus y_2 \oplus \ldots \oplus y_{d-1} \oplus y_d \]

• Noisy leakage security (Prouff, Rivain 2013)

(Duc, Dziembowski, Faust 2014)
1. What happens with parallel implementations?
   - For example: one probe reveals the shares’ sum
1. What happens with parallel implementations?
   - For example: one probe reveals the shares’ sum

2. How to test physical independence? (consolidating)
1. What happens with parallel implementations?
   - For example: one probe reveals the shares’ sum

2. How to test physical independence? (consolidating)

- W/O directly working in the noisy leakage model
Outline

- Introduction / motivation
  - Side-channel attacks and noise
  - Masking and leakage models

- Bounded moment model
  - Masking intuition & BMM definition
  - Probing security $\Rightarrow$ BM security

- Parallel multiplication (& refreshing)

- BM security $\not\Rightarrow$ probing security
  - Inner product masking (with “non-mixing” leakages)
  - Continuous security & refreshing gadgets

- Conclusions
Masking statistical intuition

- 2-share / 1-bit example, **serial** implementation

\[ L_1 = y_1 + n_1 \]
\[ L_2 = y_2 + n_2 \]

(a) \( Y = 0 \), serial.
(b) \( Y = 1 \), serial.
• 2-share / 1-bit example, parallel implementation

\[ L_1 = y_1 + n_1 \]
\[ L_2 = y_2 + n_2 \]
\[ L = y_1 + y_2 + n \]
Definition (informal). An implementation is secure at order $o$ in the bounded moment model if all mixed statistical moments of order up to $o$ of its leakage vectors are independent of any sensitive variable manipulated.
Outline

- Introduction / motivation
  - Side-channel attacks and noise
  - Masking and leakage models

- Bounded moment model
  - Masking intuition & BMM definition
  - Probing security ⇒ BM security

- Parallel multiplication (& refreshing)
  - BM security ≠ probing security
    - Inner product masking (with “non-mixing” leakages)
    - Continuous security & refreshing gadgets

- Conclusions
Abstract reduction (answer to Q1)

- **Theorem (informal).** A parallel implementation is secure at order $o$ in the BMM if its serialization is secure at order $o$ in the probing model where
  - $\text{Adv}_{pr}$ can (typically) probe $o = d - 1$ wires
  - $\text{Adv}_{bm}$ can observe any $L = \sum_{i=1}^{d} \alpha_i \cdot y_i$
Abstract reduction

• **Theorem** (informal). A parallel implementation is secure at order $o$ in the BMM if its serialization is secure at order $o$ in the probing model where
  
  - $\text{Adv}_{pr}$ can (typically) probe $o = d - 1$ wires
  - $\text{Adv}_{bm}$ can observe any $L = \sum_{i=1}^{d} \alpha_i \cdot y_i$

• Intuition: summing the shares (in $\mathbb{R}$) does not break the independent leakage assumption
• **Theorem** (informal). A parallel implementation is secure at order $o$ in the BMM if its serialization is secure at order $o$ in the probing model where

- $\text{Adv}_{pr}$ can (typically) probe $o = d - 1$ wires
- $\text{Adv}_{bm}$ can observe any $L = \sum_{i=1}^{d} \alpha_i \cdot y_i$

• Intuition: summing the shares (in $\mathbb{R}$) does not break the independent leakage assumption

• Main $\neq$ between probing and BM security
  - $\text{Adv}_{bm}$ can sum over *all* the shares!
  - BM security is weaker (moments vs. distributions)
Concrete consequence

- If physically independent leakages, BM security extends to actual measurements (e.g., $d = 3$)

![Graphs showing sample traces and their orders](image)

(a) Sample trace
(b) 1st-order
(c) 2nd-order
(d) 3rd-order
• If physically independent leakages, BM security extends to actual measurements (e.g., $d = 3$)

• If not, leakages are not independent
Introduction / motivation
- Side-channel attacks and noise
- Masking and leakage models

Bounded moment model
- Masking intuition & BMM definition
- Probing security $\Rightarrow$ BM security

Parallel multiplication (& refreshing)
- BM security $\nRightarrow$ probing security
- Inner product masking (with “non-mixing” leakages)
- Continuous security & refreshing gadgets

Conclusions
Serial multiplication

- ISW 2003: multiplication $c = a \times b$

$$\begin{bmatrix} a_1b_1 & a_1b_2 & a_1b_3 \\ a_2b_1 & a_2b_2 & a_2b_3 \\ a_3b_1 & a_3b_2 & a_3b_3 \end{bmatrix} \oplus \begin{bmatrix} 0 & r_1 & r_2 \\ -r_1 & 0 & r_3 \\ -r_2 & -r_3 & 0 \end{bmatrix} \Rightarrow \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

partial products refresh compress
Serial multiplication

- ISW 2003: multiplication $c = a \times b$

\[
\begin{bmatrix}
a_1b_1 & a_1b_2 & a_1b_3 \\
a_2b_1 & a_2b_2 & a_2b_3 \\
a_3b_1 & a_3b_2 & a_3b_3
\end{bmatrix} \oplus \begin{bmatrix}
0 & r_1 & r_2 \\
-r_1 & 0 & r_3 \\
-r_2 & -r_3 & 0
\end{bmatrix} \Rightarrow \begin{bmatrix}
c_1 \\
c_2 \\
c_3
\end{bmatrix}
\]

- partial products
- refresh

- AES S-box ($n = 8$) implementation
  - $a = a_1 \oplus a_2 \oplus \cdots \oplus a_d$ (e.g., $d = 8$)
  - Each register stores an $a_i$ (i.e., a $GF(2^8)$ element)
  - Memory $\propto n \cdot d$, Time: $\propto d^2$ $GF(2^8)$ mult.
  - AES S-box $\approx 3$ multiplications (& 4 squarings)
Parallel multiplication

- Main tweak: interleave & regularize

\[
\begin{bmatrix}
  a_1 b_1 \\
a_2 b_2 \\
a_3 b_3
\end{bmatrix} \oplus \begin{bmatrix} r_1 \end{bmatrix} \oplus \begin{bmatrix}
  a_1 b_3 & a_3 b_1 \\
a_2 b_1 & a_1 b_2 \\
a_3 b_2 & a_2 b_3
\end{bmatrix} \oplus \begin{bmatrix} r_3 \end{bmatrix} \Rightarrow \begin{bmatrix} c_1 \\
c_2 \\
c_3
\end{bmatrix}
\]
Parallel multiplication

- Main tweak: interleave & regularize

\[
\begin{bmatrix}
  a_1 b_1 \\
  a_2 b_2 \\
  a_3 b_3
\end{bmatrix}
\oplus
\begin{bmatrix}
  r_1 \\
  r_2 \\
  r_3
\end{bmatrix}
\oplus
\begin{bmatrix}
  a_1 b_3 & a_3 b_1 \\
  a_2 b_1 & a_1 b_2 \\
  a_3 b_2 & a_2 b_3
\end{bmatrix}
\oplus
\begin{bmatrix}
  r_3 \\
  r_1 \\
  r_2
\end{bmatrix}
\Rightarrow
\begin{bmatrix}
  c_1 \\
  c_2 \\
  c_3
\end{bmatrix}
\]

- AES S-box \((n = 8)\) implementation
  - \(a = a_1 \oplus a_2 \oplus \cdots \oplus a_d\) (e.g., \(d = 8\))
  - Each register stores \(n\) \(a_i\)'s (i.e., \(GF(2)\) elements)
  - Memory \(\propto n \cdot d\), Time: \(\propto d\) \(GF(2)\) mult. (i.e., ANDs)
  - AES bitslice S-box \(\approx 32\) AND gates (& 83 XORs)
Parallel multiplication

• Main tweak: interleave & regularize

\[
\begin{bmatrix}
    a_1 b_1 \\
    a_2 b_2 \\
    a_3 b_3
\end{bmatrix}
\oplus
\begin{bmatrix}
    r_1 \\
    r_2 \\
    r_3
\end{bmatrix}
\oplus
\begin{bmatrix}
    a_1 b_3 & a_3 b_1 \\
    a_2 b_1 & a_1 b_2 \\
    a_3 b_2 & a_2 b_3
\end{bmatrix}
\oplus
\begin{bmatrix}
    r_3 \\
    r_1 \\
    r_2
\end{bmatrix}
\Rightarrow
\begin{bmatrix}
    c_1 \\
    c_2 \\
    c_3
\end{bmatrix}
\]

• AES S-box \((n = 8)\) implementation
  • \(a = a_1 \oplus a_2 \oplus \cdots \oplus a_d\) (e.g., \(d = 8\))
  • Each register stores \(n\) \(a_i\)'s (i.e., GF(2) elements)
  • Memory \(\propto n \cdot d\), Time: \(\propto d\) GF(2) mult. (i.e., ANDs)
  • AES bitslice S-box \(\approx 32\) AND gates (& 83 XORs)

\(\Rightarrow\) Performance gains with large \(d\)'s (8, 16, 32)
Security analysis

• We analyzed the SNI security of the gadgets ≈ composable probing security (Barthe et al. 2016)
• We analyzed the SNI security of the gadgets
  \approx \text{composable probing security (Barthe et al. 2016)}

• Iterating \left\lfloor \frac{(d - 1)}{3} \right\rfloor \text{ refresh is SNI for } d < 12
Security analysis

- We analyzed the SNI security of the gadgets \( \approx \) composable probing security (Barthe et al. 2016)
- Iterating \( \lceil \frac{(d - 1)}{3} \rceil \) refresh is SNI for \( d < 12 \)
- Multiplication is more tricky...

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>( d )</th>
<th>( (d-1))-SNI</th>
<th>( \text{rand} ) (our alg.)</th>
<th>( \text{rand} ) (ISW 2003)</th>
</tr>
</thead>
<tbody>
<tr>
<td>multiplication</td>
<td>3</td>
<td>✓</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>( d \geq 4 )</td>
<td>×</td>
<td>( d \frac{(d-1)}{4} )</td>
<td>( d \frac{(d-1)}{2} )</td>
</tr>
<tr>
<td>refresh o multiplication</td>
<td>4</td>
<td>✓</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>✓</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>✓</td>
<td>18</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>✓</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>✓</td>
<td>24</td>
<td>28</td>
</tr>
</tbody>
</table>
Outline

• Introduction / motivation
  • Side-channel attacks and noise
  • Masking and leakage models

• Bounded moment model
  • Masking intuition & BMM definition
  • Probing security ⇒ BM security

• Parallel multiplication (& refreshing)

• BM security ≠ probing security
  • Inner product masking (with “non-mixing” leakages)
  • Continuous security & refreshing gadgets

• Conclusions
Specialized encodings

• Probing security is stronger than BM security
  • (And also stronger than noisy leakage security)
• Is it sometimes “too strong”?
  • i.e., breaks designs that are secure against DPA
Specialized encodings

• Probing security is stronger than BM security
  • (And also stronger than noisy leakage security)
• Is it sometimes “too strong”?  
  • i.e., breaks designs that are secure against DPA

• Example: Boolean encoding (2 shares)

\[ y = y_1 \oplus y_2 \]
Specialized encodings

- Probing security is stronger than BM security
  - (And also stronger than noisy leakage security)
- Is it sometimes “too strong”?
  - i.e., breaks designs that are secure against DPA
- Example: Boolean encoding (2 shares)

\[ y = y_1 \oplus y_2 \]

- IP masking in \( \text{GF}(2^8) \) with “non-mixing” leakages

\[ y = \sum_{i=1}^{2} p_i \times s_i \]

- \( p_2 = 1 \)
- \( p_2 = 5 \)
- \( p_2 = 7 \)
• Introduction / motivation
  • Side-channel attacks and noise
  • Masking and leakage models
• Bounded moment model
  • Masking intuition & BMM definition
  • Probing security ⇒ BM security
• Parallel multiplication (& refreshing)
• BM security ⇉ probing security
  • Inner product masking (with “non-mixing” leakages)
  • Continuous security & refreshing gadgets
• Conclusions
Continuous security

- So far we discussed “one-shot” probing attacks
Continuous security

• So far we discussed “one-shot” probing attacks

• Yet, side-channel attacks are usually continuous
• i.e, accumulate information from multiple executions
Continuous security

• So far we discussed “one-shot” probing attacks

• Yet, side-channel attacks are usually continuous
  • i.e, accumulate information from multiple executions

• Typical issue: refreshing by add a share of 0
  • Frequently used in practice
  • Yet insecure in the continuous probing model
  • What does it mean concretely?
  • i.e., can we (sometimes) use such a refreshing?
Continuous probing attack

- Target: refresh\( (a) = a \oplus r \oplus \text{rot}(r) \)

  step 1

\[ a_1^{(1)} \]
\[ a_2^{(1)} \]
\[ a_3^{(1)} \]
\[ a_4^{(1)} \]

Accumulated knowledge: \( \emptyset \)
Continuous probing attack

- Target: refresh\((a) = a \oplus r \oplus \text{rot}(r)\)

step 1

\[
\begin{align*}
\text{Accumulated knowledge: } & \emptyset \\
A_1^{(1)} & \\
A_2^{(1)} & \\
A_3^{(1)} & \\
A_4^{(1)} &
\end{align*}
\]
Continuous probing attack

- Target: $\text{refresh}(a) = a \oplus r \oplus \text{rot}(r)$

step 1

Accumulated knowledge: $a_1^{(1)}$
Continuous probing attack

- Target: \( \text{refresh}(a) = a \oplus r \oplus \text{rot}(r) \)

\[
\begin{array}{cccc}
\text{step 1} & \text{step 2} \\
\hline
a_1^{(1)} & r_1^{(2)} & r_4^{(2)} & a_1^{(2)} \\
a_2^{(1)} & r_2^{(2)} & r_1^{(2)} & a_2^{(2)} \\
a_3^{(1)} & r_3^{(2)} & r_2^{(2)} & a_3^{(2)} \\
a_4^{(1)} & r_4^{(2)} & r_3^{(2)} & a_4^{(2)} \\
\end{array}
\]

Accumulated knowledge: \( a_1^{(1)} \)
Continuous probing attack

- Target: \( \text{refresh}(a) = a \oplus r \oplus \text{rot}(r) \)

<table>
<thead>
<tr>
<th></th>
<th>step 1</th>
<th>step 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a_1^{(1)} )</td>
<td>( r_1^{(2)} )</td>
<td>( r_4^{(2)} )</td>
</tr>
<tr>
<td>( a_2^{(1)} )</td>
<td>( r_2^{(2)} )</td>
<td>( r_2^{(2)} )</td>
</tr>
<tr>
<td>( a_3^{(1)} )</td>
<td>( r_3^{(2)} )</td>
<td>( r_2^{(2)} )</td>
</tr>
<tr>
<td>( a_4^{(1)} )</td>
<td>( r_4^{(2)} )</td>
<td>( r_3^{(2)} )</td>
</tr>
</tbody>
</table>

Accumulated knowledge: \( a_1^{(1)} \)
Continuous probing attack

- Target: \( \text{refresh}(a) = a \bigoplus r \bigoplus \text{rot}(r) \)

<table>
<thead>
<tr>
<th></th>
<th>step 1</th>
<th>step 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a_1 ) (^{(1)})</td>
<td>( r_1 ) (^{(2)})</td>
<td>( a_1 ) (^{(2)})</td>
</tr>
<tr>
<td>( a_2 ) (^{(1)})</td>
<td>( r_2 ) (^{(2)})</td>
<td>( a_2 ) (^{(2)})</td>
</tr>
<tr>
<td>( a_3 ) (^{(1)})</td>
<td>( r_3 ) (^{(2)})</td>
<td>( a_3 ) (^{(2)})</td>
</tr>
<tr>
<td>( a_4 ) (^{(1)})</td>
<td>( r_4 ) (^{(2)})</td>
<td>( a_4 ) (^{(2)})</td>
</tr>
</tbody>
</table>

Accumulated knowledge: \( a_1 \) \(^{(1)}\)
Continuous probing attack

- Target: \( \text{refresh}(a) = a \oplus r \oplus \text{rot}(r) \)

<table>
<thead>
<tr>
<th>step 1</th>
<th>step 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a_1^{(1)} )</td>
<td>( r_1^{(2)} )</td>
</tr>
<tr>
<td>( a_2^{(1)} )</td>
<td>( r_2^{(2)} )</td>
</tr>
<tr>
<td>( a_3^{(1)} )</td>
<td>( r_3^{(2)} )</td>
</tr>
<tr>
<td>( a_4^{(1)} )</td>
<td>( r_4^{(2)} )</td>
</tr>
</tbody>
</table>

Accumulated knowledge: \( a_1^{(2)} \oplus a_2^{(2)} \)
Continuous probing attack

- Target: \( \text{refresh}(a) = a \oplus r \oplus \text{rot}(r) \)

\[
\begin{align*}
\text{step 1} & & \text{step 2} & & \text{step 3} \\
& a_1^{(1)} & & r_1^{(2)} & & r_4^{(2)} & & a_1^{(2)} & & r_1^{(3)} & & r_4^{(3)} & & a_1^{(3)} \\
& a_2^{(1)} & & r_2^{(2)} & & r_2^{(2)} & & a_2^{(2)} & & r_2^{(3)} & & r_1^{(3)} & & a_2^{(3)} \\
& a_3^{(1)} & & r_3^{(2)} & & r_2^{(2)} & & a_3^{(2)} & & r_3^{(3)} & & r_2^{(3)} & & a_3^{(3)} \\
& a_4^{(1)} & & r_4^{(2)} & & r_3^{(2)} & & a_4^{(2)} & & r_4^{(3)} & & r_3^{(3)} & & a_4^{(3)}
\end{align*}
\]

Accumulated knowledge: \( a_1^{(2)} \oplus a_2^{(2)} \)
Continuous probing attack

- Target: \( \text{refresh}(a) = a \oplus r \oplus \text{rot}(r) \)

<table>
<thead>
<tr>
<th>Step 1</th>
<th>Step 2</th>
<th>Step 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a_1^{(1)})</td>
<td>(r_1^{(2)})</td>
<td>(r_1^{(3)})</td>
</tr>
<tr>
<td>(a_2^{(1)})</td>
<td>(r_2^{(2)})</td>
<td>(r_2^{(3)})</td>
</tr>
<tr>
<td>(a_3^{(1)})</td>
<td>(r_3^{(2)})</td>
<td>(r_3^{(3)})</td>
</tr>
<tr>
<td>(a_4^{(1)})</td>
<td>(r_4^{(2)})</td>
<td>(r_4^{(3)})</td>
</tr>
</tbody>
</table>

Accumulated knowledge: \(a_1^{(2)} \oplus a_2^{(2)}\)
Continuous probing attack

- Target: $\text{refresh}(a) = a \oplus r \oplus \text{rot}(r)$

<table>
<thead>
<tr>
<th></th>
<th>step 1</th>
<th>step 2</th>
<th>step 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a^{(1)}_1$</td>
<td>$r^{(2)}_1$</td>
<td>$r^{(2)}_4$</td>
<td>$a^{(2)}_1$</td>
</tr>
<tr>
<td>$a^{(1)}_2$</td>
<td>$r^{(2)}_2$</td>
<td>$r^{(2)}_1$</td>
<td>$a^{(2)}_2$</td>
</tr>
<tr>
<td>$a^{(1)}_3$</td>
<td>$r^{(2)}_3$</td>
<td>$r^{(2)}_2$</td>
<td>$a^{(2)}_3$</td>
</tr>
<tr>
<td>$a^{(1)}_4$</td>
<td>$r^{(2)}_4$</td>
<td>$r^{(2)}_3$</td>
<td>$a^{(2)}_4$</td>
</tr>
</tbody>
</table>

Accumulated knowledge: $a^{(2)}_1 \oplus a^{(2)}_2$
Continuous probing attack

- **Target:** $\text{refresh}(a) = a \oplus r \oplus \text{rot}(r)$

<table>
<thead>
<tr>
<th></th>
<th>Step 1</th>
<th>Step 2</th>
<th>Step 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1^{(1)}$</td>
<td>$r_1^{(2)}$</td>
<td>$r_1^{(2)}$</td>
<td>$r_1^{(3)}$</td>
</tr>
<tr>
<td>$a_2^{(1)}$</td>
<td>$r_2^{(2)}$</td>
<td>$r_2^{(2)}$</td>
<td>$r_2^{(3)}$</td>
</tr>
<tr>
<td>$a_3^{(1)}$</td>
<td>$r_3^{(2)}$</td>
<td>$r_3^{(2)}$</td>
<td>$r_3^{(3)}$</td>
</tr>
<tr>
<td>$a_4^{(1)}$</td>
<td>$r_4^{(2)}$</td>
<td>$r_4^{(2)}$</td>
<td>$r_4^{(3)}$</td>
</tr>
</tbody>
</table>

Accumulated knowledge: $a_1^{(3)} \oplus a_2^{(3)} \oplus a_3^{(3)}$
Continuous probing attack

- Target: \( \text{refresh}(a) = a \oplus r \oplus \text{rot}(r) \)

<table>
<thead>
<tr>
<th>Step 1</th>
<th>Step 2</th>
<th>Step 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a_1^{(1)} )</td>
<td>( r_1^{(2)} )</td>
<td>( a_1^{(2)} )</td>
</tr>
<tr>
<td>( a_2^{(1)} )</td>
<td>( r_2^{(2)} )</td>
<td>( a_2^{(2)} )</td>
</tr>
<tr>
<td>( a_3^{(1)} )</td>
<td>( r_3^{(2)} )</td>
<td>( a_3^{(2)} )</td>
</tr>
<tr>
<td>( a_4^{(1)} )</td>
<td>( r_4^{(2)} )</td>
<td>( a_4^{(2)} )</td>
</tr>
</tbody>
</table>

\[ \Rightarrow \text{After } d \text{ iterations, } a \text{ is learned in full by Adv}_{pr} \]
### Continuous probing attack

- **Target:** \( \text{refresh}(a) = a \oplus r \oplus \text{rot}(r) \)

<table>
<thead>
<tr>
<th>step 1</th>
<th>step 2</th>
<th>step 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a_1^{(1)} )</td>
<td>( r_1^{(2)} ) ( r_4^{(2)} ) ( a_1^{(2)} )</td>
<td>( r_1^{(3)} ) ( r_4^{(3)} ) ( a_1^{(3)} )</td>
</tr>
<tr>
<td>( a_2^{(1)} )</td>
<td>( r_2^{(2)} ) ( r_1^{(2)} ) ( a_2^{(2)} )</td>
<td>( r_2^{(3)} ) ( r_1^{(3)} ) ( a_2^{(3)} )</td>
</tr>
<tr>
<td>( a_3^{(1)} )</td>
<td>( r_3^{(2)} ) ( r_2^{(2)} ) ( a_3^{(2)} )</td>
<td>( r_3^{(3)} ) ( r_2^{(3)} ) ( a_3^{(3)} )</td>
</tr>
<tr>
<td>( a_4^{(1)} )</td>
<td>( r_4^{(2)} ) ( r_3^{(2)} ) ( a_4^{(2)} )</td>
<td>( r_4^{(3)} ) ( r_3^{(3)} ) ( a_4^{(3)} )</td>
</tr>
</tbody>
</table>

⇒ After \( d \) iterations, \( a \) is learned in full by \( \text{Adv}_{pr} \)

- **Not possible in the BMM. Intuition:** *adaptation does not help* since \( \text{Adv}_{bm} \) can anyway sum over all shares!
Continuous probing attack

- **Target:** \( \text{refresh}(a) = a \oplus r \oplus \text{rot}(r) \)

<table>
<thead>
<tr>
<th>Step 1</th>
<th>Step 2</th>
<th>Step 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a_1^{(1)} )</td>
<td>( r_1^{(2)} )</td>
<td>( r_1^{(3)} )</td>
</tr>
<tr>
<td>( a_2^{(1)} )</td>
<td>( r_2^{(2)} )</td>
<td>( r_2^{(3)} )</td>
</tr>
<tr>
<td>( a_3^{(1)} )</td>
<td>( r_3^{(2)} )</td>
<td>( r_3^{(3)} )</td>
</tr>
<tr>
<td>( a_4^{(1)} )</td>
<td>( r_4^{(2)} )</td>
<td>( r_4^{(3)} )</td>
</tr>
</tbody>
</table>

⇒ After \( d \) iterations, \( a \) is learned in full by \( \text{Adv}_{pr} \)

- **Impact:** \( \text{refresh}(\ . \ ) \) can be used to refresh the key of a key homomorphic primitive (⇒ fully linear overheads)
• Introduction / motivation
  • Side-channel attacks and noise
  • Masking and leakage models

• Bounded moment model
  • Masking intuition & BMM definition
  • Probing security ⇒ BM security

• Parallel multiplication (& refreshing)

• BM security ⇉ probing security
  • Inner product masking (with “non-mixing” leakages)
  • Continuous security & refreshing gadgets

• Conclusions
Conclusions

- Probing security is relevant to parallel implem.
Conclusions

- Probing security is relevant to parallel implem.
- BMM suggests a principled path to security eval.

\[ \text{probing security} \xrightarrow{\text{[DDF14]}} \text{noisy leakages security} \]
\[ + \text{noise, } \star \]

\[ + \text{noise} \]
\[ + \text{noise & ???} \]

\[ \text{bounded moment security} \]
Conclusions

- Probing security is relevant to parallel implem.
- BMM suggests a principled path to security eval.

Parallel implem. are appealing for masking
  - Leverage the memory needed to store shares
Conclusions

• Probing security is relevant to parallel implem.
• BMM suggests a principled path to security eval.
• Parallel implem. are appealing for masking
• Leverage the memory needed to store shares
• Cont. probing security sometimes “too strong”
THANKS

http://perso.uclouvain.be/fstandae/