Towards an Open Approach to Side-Channel Resistant Authenticated Encryption

François-Xavier Standaert

UCLouvain, ICTEAM, Crypto Group (Belgium)

ASHES 2019, London, UK
• Block ciphers & symmetric encryption
• Secure cryptographic implementations
1. Side-channel *(crypto*)analysis: attacks taxonomy
2. Masking *countermeasure*: security vs. cost
3. Security *definitions* (authenticated encryption)
   a. Nonce-respecting setting (i.e., AEL)
   b. Nonce-misuse setting (i.e., AEml)

1. Leakage-resistant AE *designs* (& implementations)
   • Level 0: no mode-level leakage-resistance
   • Level 1: re-keyed modes (including sponges)
   • Level 2: level 1 + strengthened init./final.
   • Level 3: level 2 + two-passes

2. Conclusions (& the need of open evaluations)
Acknowledgments & cautionary note

• Mixing (very) different abstraction levels
  • Hopefully in a consistent manner (*be forgiving if not*)
1. Side-channel (crypt)analysis: attacks taxonomy

2. Masking countermeasure: security vs. cost

3. Security definitions (authenticated encryption)
   a. Nonce-respecting setting (i.e., AEL)
   b. Nonce-misuse setting (i.e., AEmL)

1. Leakage-resistant AE designs (& implementations)
   • Level 0: no mode-level leakage-resistance
   • Level 1: re-keyed modes (including sponges)
   • Level 2: level 1 + strengthened init./final.
   • Level 3: level 2 + two-passes

2. Conclusions (& the need of open evaluations)
AES Rijndael: $y = AES_K(x)$
Leaking AES: $y = \text{AES}_K(x) \rightarrow L$
• Leakages are vectors: \( \mathbf{L} = (L^1, L^2, ..., L^t) \)
• Made of many samples (\( t \approx 10^3-10^6 \))
Leakage function definition

- Leakages are vectors: $L = (L^1, L^2, ..., L^t)$
- Made of many samples ($t \approx 10^3-10^6$)
- Leakages are noisy: $L(x, K) \approx \delta(x, K) + N$
- Signal-to-Noise Ratio: $\text{SNR}^i = \frac{\text{var}(\delta^i_x)}{\text{var}(N^i)}$

1.2
Leakage function definition

- Leakages are vectors: $L = (L^1, L^2, \ldots, L^t)$
  - Made of many samples ($t \approx 10^3$-$10^6$)

- Leakages are noisy: $L(x, K) \approx \delta(x, K) + N$

- Signal-to-Noise Ratio: $\text{SNR}^i = \frac{\text{var}(\delta^i_x)}{\text{var}(N^i)}$

- The shape of $\delta$ & $N$ is technology-dependent
  - Their exact representation is unknown
• Computing less means leaking less
  • E.g., unprotected **32-bit** implem. (**HW** leakages)

<table>
<thead>
<tr>
<th># rounds</th>
<th># ops. / round</th>
<th># samples / op.</th>
<th>MI (bits) / sample</th>
<th>λ (bits) / trace</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>100</td>
<td>5</td>
<td>[\log(32) = 5]</td>
<td>[\frac{25,000}{1 + \frac{1}{\text{SNR}}}]</td>
</tr>
</tbody>
</table>
• Computing less means leaking less
  • E.g., unprotected 32-bit implem. (HW leakages)

<table>
<thead>
<tr>
<th># rounds</th>
<th># ops. / round</th>
<th># samples / op.</th>
<th>MI (bits) / sample</th>
<th>λ (bits) / trace</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>100</td>
<td>5</td>
<td>(\log(32) = 5) (\frac{1}{1 + \frac{1}{\text{SNR}}})</td>
<td>(25,000) (\frac{1}{1 + \frac{1}{\text{SNR}}})</td>
</tr>
</tbody>
</table>

• Unprotected 128-bit implem. (HW leakages)

<table>
<thead>
<tr>
<th># rounds</th>
<th># ops. / round</th>
<th># samples / op.</th>
<th>MI (bits) / sample</th>
<th>λ (bits) / trace</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>1</td>
<td>5</td>
<td>(\log(128) = 7) (\frac{1}{1 + \frac{1}{\text{SNR}}})</td>
<td>(350) (\frac{1}{1 + \frac{1}{\text{SNR}}})</td>
</tr>
</tbody>
</table>
• Games that give the adversary the ability to compare the leakages of two identical device states are in general trivial to win. For example, given a keyed offline leakage oracle \( L(., K) \):

\[
\Pr \left[ A_{\text{SC}}^{L(., K)} (x_0, x_1, L(x_b, K)) = b | K, b \leftarrow $ \right] \approx 1
\]

• Just compare \( L(x_b, K) \) with \( L(x_0, K) \) and \( L(x_1, K) \)
• (SC stands for « state comparison » attack)
Consequence (for theoretical analysis)

- Games that give the adversary the ability to compare the leakages of two identical device states are in general trivial to win. For example, given a keyed offline leakage oracle $L(., K)$:

$$\Pr \left[ A_{SC}^{L(., K)}(x_0, x_1, L(x_b, K)) = b \mid K, b \leftarrow \$ \right] \approx 1$$

- Just compare $L(x_b, K)$ with $L(x_0, K)$ and $L(x_1, K)$
- (SC stands for « state comparison » attack)

⇒ Distinguishing games without anything fresh and secret in the challenge are trivial to win
• Key recovery attacks may not easily exploit all leakage samples (since $A$ needs to guess the state), leading to reduced « effective » $\lambda$’s, e.g.,

<table>
<thead>
<tr>
<th>exploited # rounds</th>
<th># ops. / round</th>
<th># samples / op.</th>
<th>MI (bits) / sample</th>
<th>eff. $\lambda$ (bits) / trace &amp; subkey</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>$\approx 2$ (indep.)</td>
<td>$\log(128) = 7$</td>
<td>$\frac{14}{100}$</td>
</tr>
</tbody>
</table>

• One key byte recovered in $\approx \frac{128}{0.14} \approx 1000$ traces
Basic facts (II)

- Key recovery attacks may not easily exploit all leakage samples (since $A$ needs to guess the state), leading to reduced « effective » $\lambda$’s, e.g.,

<table>
<thead>
<tr>
<th>exploited # rounds</th>
<th># ops. / round</th>
<th># samples / op.</th>
<th>MI (bits) / sample</th>
<th>eff. $\lambda$ (bits) / trace &amp; subkey</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>$\approx 2 \text{ (indep.)}$</td>
<td>$\log(128) = 7$</td>
<td>$\frac{14}{100}^d$</td>
</tr>
</tbody>
</table>

- With the masking countermeasures (see next)
• Key recovery attacks may not easily exploit all leakage samples (since $A$ needs to guess the state), leading to reduced « effective » $\lambda$’s, e.g.,

<table>
<thead>
<tr>
<th>exploited # rounds</th>
<th># ops. / round</th>
<th># samples / op.</th>
<th>MI (bits) / sample</th>
<th>eff. $\lambda$ (bits) / trace &amp; subkey</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>$\approx 2$ (indep.)</td>
<td>$\log(128) = 7$ (\frac{100}{100})</td>
<td>$\approx \left(\frac{14}{100}\right)^d$</td>
</tr>
</tbody>
</table>

• With the masking countermeasures (see next)

• *(128-bit example, 32-bit case significantly harder)*
Basic facts (III)

• \((q, r)\)-bounded SCAs are « continuous » attacks
• with \(q\) different message blocks per key
• and each measurement repeated \(r\) times

⇒ Typical success probability (e.g., for key recovery):

\[
\Pr \left[ A_{KR} \left( x_1, L(x_1, K), \ldots, x_q, L(x_q, K) \right) \rightarrow K \mid K ← \$ \right] \approx 2^{-128+q \cdot \lambda(r)}
\]
Basic facts (III)

• \((q, r)\)-bounded SCAs are « continuous » attacks
  • with \(q\) different message blocks per key
  • and each measurement repeated \(r\) times

⇒ Typical success probability (e.g., for key recovery):

\[
\Pr \left[ A_{KR} \left( x_1, L(x_1, K), \ldots, x_q, L(x_q, K) \right) \rightarrow K | K \leftarrow $ \right] \approx 2^{-128+q\cdot\lambda(r)}
\]

• There are two main types of attacks (jargon)
  • **SPA**: \(q\) is a small constant (e.g., thanks to re-keying)
  • **DPA**: \(q\) can be large & is adversarially chosen
Basic facts (III)

- \((q, r)\)-bounded SCAs are « continuous » attacks
  - with \(q\) different message blocks per key
  - and each measurement repeated \(r\) times

\[ \Pr \left[ A_{KR} \left( x_1, L(x_1, K), \ldots, x_q, L(x_q, K) \right) \rightarrow K | K \leftarrow $ \right] \approx 2^{-128 + q \cdot \lambda(r)} \]

- There are two main types of attacks (jargon)
  - SPA: \(q\) is a small constant (e.g., thanks to re-keying)
  - DPA: \(q\) can be large & is adversarially chosen

- Larger \(r\)’s can improve the SNR (average the noise)
• Key Recovery (KR) attacks (with known/chosen $x_i$’s)

$$\Pr \left[ A_{KR} \left( x_1, L(x_1, K), \ldots, x_q, L(x_q, K) \right) \rightarrow K | K \leftarrow $ \right] \approx 2^{-128+q \cdot \lambda(r)}$$

• May require large amounts of leakage vectors to succeed
Key Recovery (KR) attacks (with known/chosen $x_i$’s)

$$\Pr \left[ A_{KR} \left( x_1, L(x_1, K), \ldots, x_q, L(x_q, K) \right) \rightarrow K | K \leftarrow \$ \right] \approx 2^{-128+q \cdot \lambda(r)}$$

- May require large amounts of leakage vectors to succeed
- Or have bounded success probability in case of SPA
Summarizing (taxonomy of attacks) 1.7

- Key Recovery (KR) attacks (with known/chosen $x_i$’s)
  $\Pr[A_{KR}(x_1, L(x_1, K), \ldots, x_q, L(x_q, K)) \rightarrow K | K \leftarrow $) $\approx 2^{-128+q \cdot \lambda(r)}$
  - May require large amounts of leakage vectors to succeed
  - Or have bounded success probability in case of SPA

- State Comparison (SC) attacks (with keyed oracle)
  - $\Pr[A_{SC}^{L(K)}(x_0, x_1, L(x_b, K)) = b | K, b \leftarrow $) $\approx 1$ anyway
Summarizing (taxonomy of attacks)

- **Key Recovery (KR) attacks** (with known/chosen $x_i$’s)
  \[
  \Pr\left[A_{KR}\left(x_1, L(x_1, K), \ldots, x_q, L(x_q, K)\right) \to K | K \leftarrow \$ight] \approx 2^{-128+q \cdot \lambda(r)}
  \]
  - May require large amounts of leakage vectors to succeed
  - Or have bounded success probability in case of SPA

- **Message Comparison (MC) attacks** (with fresh challenge)
  \[
  \Pr\left[A_{MC}^{L(\cdot, \cdot)}(x_0, x_1, L(x_b, K)) = b | K, b \leftarrow \$ight] \approx 2^{-128+D(L(x_0,K);L(x_1,K))}
  \]
  - Significantly simpler than KR - but not trivial for all $x_0, x_1$ (!)
  - Depends on similarity of the message blocks’ leakages

- **State Comparison (SC) attacks** (with keyed oracle)
  - \[
  \Pr\left[A_{SC}^{L(\cdot,K)}(x_0, x_1, L(x_b, K)) = b | K, b \leftarrow \$\right] \approx 1 \text{ anyway}
  \]
Outline

1. Side-channel (crypt)analysis: attacks taxonomy
2. Masking countermeasure: security vs. cost
3. Security definitions (authenticated encryption)
   a. Nonce-respecting setting (i.e., AEL)
   b. Nonce-misuse setting (i.e., AEmL)

1. Leakage-resistant AE designs (& implementations)
   • Level 0: no mode-level leakage-resistance
   • Level 1: re-keyed modes (including sponges)
   • Level 2: level 1 + strengthened init./final.
   • Level 3: level 2 + two-passes

2. Conclusions (& the need of open evaluations)
Noise (hardware) is not enough

\[ Y = 0 \quad Y = 1 \]
Noise (hardware) is not enough
Noise (hardware) is not enough

- Additive noise \( \approx \) cost \( \times 2 \) \( \Rightarrow \) security \( \times 2 \) 
  \( \Rightarrow \) not a good (crypto) security parameter
- \( \approx \) same holds for all hardware countermeasures
Example: Boolean encoding

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

With \( y_1, y_2, \ldots, y_{d-2}, y_{d-1} \leftarrow \{0,1\}^n \)
• Private circuits / probing security [ISW03]

\[ y = y_1 \oplus y_2 \oplus \cdots \oplus y_{d-1} \oplus y_d \]
• Private circuits / probing security [ISW03]

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

• \( d - 1 \) probes do not reveal anything on \( y \)
• Private circuits / probing security [ISW03]

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

• But \( d \) probes completely reveal \( y \)
Masking (concrete view)

- Private circuits / probing security [ISW03]

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

- Noisy leakage security [PR13]

serial implementation.
Masking (concrete view)

- Private circuits / probing security [ISW03]

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

- Bounded information \( \text{MI}(Y; L) < \text{MI}(Y_i; L_{Y_i})^d \)
• Private circuits / probing security [ISW03]

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

Masking (reduction)

• Bounded information \( \text{MI}(Y; L) \leq \text{MI}(Y_i; L_{Y_i})^d \)

noise + independence

[DDF14]
• Linear operations: \( f(a) = f(a_1) \oplus f(a_2) \oplus \cdots \oplus f(a_d) \)
Masked operations [ISW03]

- **Linear operations:** \[ f(a) = f(a_1) \oplus f(a_2) \oplus \cdots \oplus f(a_d) \]

- **Multiplications:** \[ c = a \times b \] in three steps
• Linear operations: \( f(a) = f(a_1) \oplus f(a_2) \oplus \cdots \oplus f(a_d) \)

• Multiplications: \( c = a \times b \) in three steps

\[
\begin{bmatrix}
a_1b_1 & a_1b_2 & a_1b_3 \\
a_2b_1 & a_2b_2 & a_2b_3 \\
a_3b_1 & a_3b_2 & a_3b_3
\end{bmatrix}
\]

partial products
Masked operations [ISW03]

- **Linear operations:** $f(a) = f(a_1) \oplus f(a_2) \oplus \cdots \oplus f(a_d)$

- **Multiplications:** $c = a \times b$ in three steps

\[
\begin{bmatrix}
a_1b_1 & a_1b_2 & a_1b_3 \\
a_2b_1 & a_2b_2 & a_2b_3 \\
a_3b_1 & a_3b_2 & a_3b_3
\end{bmatrix}
+ \begin{bmatrix}
0 & r_1 & r_2 \\
-r_1 & 0 & r_3 \\
-r_2 & -r_3 & 0
\end{bmatrix}
\]

partial products refreshing
Masked operations [ISW03]

- **Linear operations**: \( f(a) = f(a_1) \oplus f(a_2) \oplus \ldots \oplus f(a_d) \)

- **Multiplications**: \( c = a \times b \) in three steps

\[
\begin{bmatrix}
  a_1 b_1 & a_1 b_2 & a_1 b_3 \\
  a_2 b_1 & a_2 b_2 & a_2 b_3 \\
  a_3 b_1 & a_3 b_2 & a_3 b_3 \\
\end{bmatrix}
\begin{bmatrix}
  0 & r_1 & r_2 \\
  -r_1 & 0 & r_3 \\
  -r_2 & -r_3 & 0 \\
\end{bmatrix}
\Rightarrow
\begin{bmatrix}
  c_1 \\
  c_2 \\
  c_3 \\
\end{bmatrix}
\]

Partial products

- Refreshing
- Compression
Masked operations [ISW03]

- **Linear operations**: \( f(a) = f(a_1) \oplus f(a_2) \oplus \ldots \oplus f(a_d) \)

- **Multiplications**: \( c = a \times b \) in three steps
  
  \[
  \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} +
  \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 \hspace{1cm} \text{refreshing} \hspace{1cm} \text{compression}

  \[ a_1b_1 \oplus a_1b_2 \oplus a_1b_3 = a_1b \] leaks on \( b \)
Masked operations [ISW03]

- **Linear operations**: \( f(a) = f(a_1) \oplus f(a_2) \oplus \cdots \oplus f(a_d) \)

- **Multiplications**: \( c = a \times b \) in three steps

\[
\begin{bmatrix}
  a_1 b_1 & a_1 b_2 & a_1 b_3 \\
  a_2 b_1 & a_2 b_2 & a_2 b_3 \\
  a_3 b_1 & a_3 b_2 & a_3 b_3
\end{bmatrix} + \begin{bmatrix}
  0 & r_1 & r_2 \\
  -r_1 & 0 & r_3 \\
  -r_2 & -r_3 & 0
\end{bmatrix} \Rightarrow \begin{bmatrix}
  c_1 \\
  c_2 \\
  c_3
\end{bmatrix}
\]

- Partial products compression
- Refreshing

\[ a_1 b_1 \oplus a_1 b_2 \oplus a_1 b_3 = a_1 b \text{ leaks on } b \]

\( \Rightarrow \) Quadratic overheads & randomness

- (Many published optimizations [R+15,Be+16,GM18])
Statistical intuition (2 shares)

- Leakage mean vector for $Y = 0, 1 = [0.5, 0.5]$
• Leakage mean value for $Y = 0, 1 = 1$
Case study: ARM Cortex M4 [JS17]
Case study: ARM Cortex M4 [JS17]

security

performance

2.7
Case study: ARM Cortex M4 [JS17]

security

- 31st-order security
- 15th-order security
- 7th-order security
- $2^{128}$-bit security
- $2^{64}$-bit security
- measured SNR

performance

\[ \log_{10}(\text{MI}) \]

\[ \log_{10}(\text{SNR}) \]

\[ \log_{10}(\text{cycles per byte}) \]

number of shares
• Sounds easy but implementation is complex
• Sounds easy but implementation is complex
  • **Independence issue**: physical defaults (e.g., glitches) can re-combine shares (e.g., [MPG05,NRS11,F+18])
  • Security against horizontal attacks require more *noise/randomness* as $d$ increases [BCPZ16,CS19]
  • Scalability/*composition* are challenging [Ba+15,Ba+16]
• Sounds easy but implementation is complex
  • *Independence issue*: physical defaults (e.g., glitches) can re-combine shares (e.g., [MPG05,NRS11,F+18])
  • Security against horizontal attacks require more *noise/randomness* as $d$ increases [BCPZ16,CS19]
  • Scalability/*composition* are challenging [Ba+15,Ba+16]

⇒ High security against DPA can be reached but
  • It implies large performance overheads
    • E.g., industry currently uses 2-4 shares (?)
  • It « only » protects the key (plaintexts are not shared)
Summarizing

- Sounds easy but implementation is complex
  - Independence issue: physical defaults (e.g., glitches) can re-combine shares (e.g., [MPG05,NRS11,F+18])
  - Security against horizontal attacks require more noise/randomness as $d$ increases [BCPZ16,CS19]
  - Scalability/composition are challenging [Ba+15,Ba+16]

⇒ High security against DPA can be reached but
  - It implies large performance overheads
    - E.g., industry currently uses 2-4 shares (?)
    - It « only » protects the key (plaintexts are not shared)

- SPA security expected to be (much) cheaper
Outline

1. Side-channel (crypt)analysis: attacks taxonomy
2. Masking countermeasure: security vs. cost
3. Security definitions (authenticated encryption)
   a. Nonce-respecting setting (i.e., AEL)
   b. Nonce-misuse setting (i.e., AEmL)

1. Leakage-resistant AE designs (& implementations)
   • Level 0: no mode-level leakage-resistance
   • Level 1: re-keyed modes (including sponges)
   • Level 2: level 1 + strengthened init./final.
   • Level 3: level 2 + two-passes

2. Conclusions (& the need of open evaluations)
Why not extending [RS06]’s all in one definition?

\[ E_K(\ldots) \]  
\[ D_K(\ldots) \]  
\[ N, AD, M \]  
\[ N, AD, C \]  
\[ \bot (\ldots) \]  
\[ \bot (\ldots) \]  
\[ C \]  
\[ M \]  
\[ A \]  
\[ \$\ldots) \]

A cannot ask a decryption query on \((N, AD, C)\) after \(C\) is returned by an \((N, AD, \ldots)\) encryption query.
• Why not extending [RS06]’s \textit{all in one} definition?

• \(A\) cannot ask a decryption query on \((N, AD, C)\) after \(C\) is returned by an \((N, AD, .)\) encryption query

• Problem: the leakage of ideal objects (which do not have implementations) seems difficult to define
Ciphertext Integrity

\[ E_K(\ldots) \quad \text{and} \quad D_K(\ldots) \]

\[ N, AD, M \quad \text{and} \quad N, AD, C \]

Ciphertext $C$ is decrypted to $fresh\ C^*$ using $D_K(\ldots)$.

Diagram showing the process of encryption and decryption with inputs $N, AD, M$ and $N, AD, C$ leading to the output $fresh\ C^*$. The diagram also includes a note on security considerations.
• CIL1: leakage in encryption only [Be+18]
Ciphertext Integrity with Leakage

- CIL2: leakage in encryption and decryption [BPPS17]
- Natural extensions (no definitional challenges) with many applications (e.g., secure bootloadding)
Chosen Ciphertext Security

\[ C^* = E_K(N^*, AD, M_b) \]

\[ \text{b} \]

\[ (K, b) \]

\[ E_K(\ldots, \ldots) \]

\[ D_K(\ldots, \ldots) \]
CCA Security with Leakage [GPPS18]

3.3

- **CCAL1**: leakage in encryption
CCA Security with Leakage [GPPS18]

- CCAL2: leakage in encryption and decryption
CCA Security with Leakage [GPPS18]

- + challenge $Ldec^*$ (applications: IP protection, ...)

$N, AD, M$

$C, Lenc$

$N, AD, C$

$M, Ldec$

$N^*, AD, M_0, M_1$

$C^* = E_K(N^*, AD, M_b), Lenc^*$

$E_K(., ., ., .)$

$D_K(., ., ., .)$

$(K, b)$
• [MR04] (and [NS09,BG10,...]): indistinguishability with $Lenc^*$ is hard (one bit breaks it with $p = 1$)
  • So it is quite tempting to ignore it
  • Which can make sense (e.g., if you tolerate « local attacks » but not « global » security degradations)
    • Leakage-resilience vs. leakage-resistance
The challenge leakage controversy (I)

- [MR04] (and [NS09,BG10,...]): indistinguishability with \( Lenc^* \) is hard (one bit breaks it with \( p = 1 \))
  - So it is quite tempting to ignore it
  - Which can make sense (e.g., if you tolerate « local attacks » but not « global » security degradations)
    - Leakage-resilience vs. leakage-resistance

- Ignoring challenge leakages means that an implementation leaking messages in full is OK
  - This is not what we want in general / theory
  - It can have big impact (e.g., TLS [CHV03],[AP13], ...)
    - Different attacks but they show plaintext leakage matters
• If we do not make it part of the definition it will never be a goal for cryptographers & engineers
  • Cryptographers: minimize the message manipulation
  • Engineers: minimize message leakage, e.g., with special encodings (which is not much studied yet)
• If we do not make it part of the definition it will never be a goal for cryptographers & engineers
  • Cryptographers: minimize the message manipulation
  • Engineers: minimize message leakage, e.g., with special encodings (which is not much studied yet)

• We need to understand what can be achieved
  • Even if results are not ideal (e.g., no negl. Adv.)
The challenge leakage controversy (II)

• If we do not make it part of the definition it will never be a goal for cryptographers & engineers
  • Cryptographers: minimize the message manipulation
  • Engineers: minimize message leakage, e.g., with special encodings (which is not much studied yet)

• We need to understand what can be achieved
  • Even if results are not ideal (e.g., no negl.Adv.)

• Technically: more greyscale view than [MR04]
  • Challenge leakages allow Message Comparison (MC) attacks which are not always trivial, e.g.,
    • Remote timing attacks: scalar leakages (vs. vectors)
    • Proxy re-encryption: messages are not chosen
An motivating example

- Tree-based leakage-resilient PRF [GGM84, FPS12]
An motivating example

- Tree-based leakage-resilient PRF [GGM84,FPS12]

- Leads to simple MC attacks
  - Message encrypted bit per bit ⇒ no algorithmic noise
  - Constant block cipher inputs « all zeros » and « all ones » easy to distinguish with HWs [B12]

- (Yet is quite good against KR)
Outline

1. Side-channel (crypt)analysis: attacks taxonomy
2. Masking countermeasure: security vs. cost
3. Security definitions (authenticated encryption)
   a. Nonce-respecting setting (i.e., AEL)
   b. Nonce-misuse setting (i.e., AEmL)

1. Leakage-resistant AE designs (& implementations)
   • Level 0: no mode-level leakage-resistance
   • Level 1: re-keyed modes (including sponges)
   • Level 2: level 1 + strengthened init./final.
   • Level 3: level 2 + two-passes

2. Conclusions (& the need of open evaluations)
• Black box: only identical \((N, M)\) pairs should be at risk
• Typically achieved by having a 2-pass mode (e.g., SIV)
With leakage: a SC attack against $M_1 = \{x_1, x_2, x_3, x_4\}$ and $M_2 = \{x_1, x_2, x_3, x_4^*\}$ leaks that they first blocks are equal
• Fresh challenge nonce circumvent this impossibility
• Intuition: leaves mostly MC attacks and DPAs
• For confidentiality, no meaningful encryption scheme seem to ensure leakage-resistance and (nonce) misuse-resistance (excluding trivial / leak-free solutions)
• For confidentiality, no meaningful encryption scheme seem to ensure leakage-resistance and (nonce) misuse-resistance (excluding trivial / leak-free solutions).

• Natural combinations include:
  a. Misuse-resilience/leakage-resistance: CCAmL [GPPS18]
  b. Misuse-resistance/leakage-resilience: CCAML [BMOS17]
• For confidentiality, no meaningful encryption scheme seem to ensure leakage-resistance and (nonce) misuse-resistance (excluding trivial / leak-free solutions)

• Natural combinations include:
  a. Misuse-resilience/leakage-resistance: CCAmL [GPPS18]
  b. Misuse-resistance/leakage-resilience: CCAMI [BMOS17]

• ≈ a choice between the need for applications to limit the leakage or for implementers to control nonces
• For confidentiality, no meaningful encryption scheme seem to ensure leakage-resistance and (nonce) misuse-resistance (excluding trivial / leak-free solutions).

• Natural combinations include:
  a. Misuse-resilience/leakage-resistance: CCAmL [GPPS18]
  b. Misuse-resistance/leakage-resilience: CCAMl [BMOS17]

• ≈ a choice between the need for applications to limit the leakage or for implementers to control nonces.

• Strongest def.: AEML=CIML2+CCAmL2+CCAMl2
For confidentiality, no meaningful encryption scheme seem to ensure leakage-resistance and (nonce) misuse-resistance (excluding trivial / leak-free solutions).

Natural combinations include:

a. Misuse-resilience/leakage-resistance: CCAmL [GPPS18]
b. Misuse-resistance/leakage-resilience: CCAMl [BMOS17]

≈ a choice between the need for applications to limit the leakage or for implementers to control nonces.

Strongest def.: AEmL=CIML2+CCAmL2+CCAMl2

Weaker variants can be meaningful: for instance AEmL=CIML2+CCAmL2 [Be+19], CPAI1 [DM19], ...
Outline

1. Side-channel (crypt)analysis: attacks taxonomy
2. Masking countermeasure: security vs. cost
3. Security definitions (authenticated encryption)
   a. Nonce-respecting setting (i.e., AEL)
   b. Nonce-misuse setting (i.e., AEmL)

1. Leakage-resistant AE designs (& implementations)
   • Level 0: no mode-level leakage-resistance
   • Level 1: re-keyed modes (including sponges)
   • Level 2: level 1 + strengthened init./final.
   • Level 3: level 2 + two-passes

2. Conclusions (& the need of open evaluations)
• Identify main steps, e.g., inner keyed sponge
• Identify main steps, e.g., inner keyed sponge

• Choose the target for confidentiality & integrity
• Reduce the mode to (weak) assumptions (tightly)

only computation leaks

leak-free components bounded leakage

strong unpredictability with leakage

simulatable leakages hard-to-invert leakages

oracle-free leakages [...]

4.2
4.3 Practical evaluation (I)

- Translate assumptions into necessary design goals

<table>
<thead>
<tr>
<th></th>
<th>init./final.</th>
<th>bulk comp.</th>
<th>tag verif.</th>
</tr>
</thead>
<tbody>
<tr>
<td>conf.</td>
<td>DPA (key recovery)</td>
<td>DPA (key recovery)</td>
<td>SPA (key recovery)</td>
</tr>
<tr>
<td>int.</td>
<td>DPA (key recovery)</td>
<td>DPA (key recovery)</td>
<td>SPA (key recovery)</td>
</tr>
</tbody>
</table>

- **Set the target security level** ($2^m$ leakages, $2^t$ time)
- **Evaluate implementation cost & performances**
### Practical evaluation (II)

- **Approximate performance overheads**

<table>
<thead>
<tr>
<th></th>
<th>init./final.</th>
<th>bulk comp.</th>
<th>tag verif.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>conf.</strong></td>
<td>x 5 – 10 – &gt;100</td>
<td>x 5 – 10 – &gt;100 x 1 – 5</td>
<td>Ø</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x 1 – 5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>⬤</td>
<td></td>
</tr>
<tr>
<td><strong>int.</strong></td>
<td>x 5 – 10 – &gt;100</td>
<td>x 5 – 10 – &gt;100 x 1 – 5</td>
<td>x 5 – 10 – &gt;100 x 1 – 5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x 1</td>
<td></td>
</tr>
</tbody>
</table>

- **DPA security**: high-order masking, shuffling, ...
- **SPA security**: parallel implementations, noise, ...
1. Side-channel (crypt)analysis: attacks taxonomy
2. Masking countermeasure: security vs. cost
3. Security definitions (authenticated encryption)
   a. Nonce-respecting setting (i.e., AEL)
   b. Nonce-misuse setting (i.e., AEmL)

1. Leakage-resistant AE designs (& implementations)
   • Level 0: no mode-level leakage-resistance
   • Level 1: re-keyed modes (including sponges)
   • Level 2: level 1 + strengthened init./final.
   • Level 3: level 2 + two-passes

2. Conclusions (& the need of open evaluations)
• Target: CCAL1, CIL1 (L in enc only, no misuse)

• Needs DPA resistance for all $E_K$ blocks
  • Primitive/implementtion SCA security only
• Target: CCAL1, CIL1 (L in enc only, no misuse)

• Needs DPA resistance for all $E_K$ blocks
  • Primitive/implementation SCA security only

• Others: SKINNY-AEAD, SUNDAE-GIFT, OCB-AES, ...
Outline

1. Side-channel (crypt)analysis: attacks taxonomy
2. Masking countermeasure: security vs. cost
3. Security definitions (authenticated encryption)
   a. Nonce-respecting setting (i.e., AEL)
   b. Nonce-misuse setting (i.e., AEml)

1. Leakage-resistant AE designs (& implementations)
   • Level 0: no mode-level leakage-resistance
   • Level 1: re-keyed modes (including sponges)
   • Level 2: level 1 + strengthened init./final.
   • Level 3: level 2 + two-passes

2. Conclusions (& the need of open evaluations)
- **Target**: CCAL1, CIL1 (L in enc only, no misuse)

- Bulk computation only requires SPA security
  - Light green: no averaging is possible (fresh states)
  - Calling for so-called “levelled” implementations
    - Energy gains thanks to 2 different implementations
• Target: CCAmL1, CIML1 (L in enc only, misuse)

• DPA security needed everywhere with nonce misuse (idem with decryption leakages)
• Target: CCAmL1, CIML1 (L in enc only, misuse)

• DPA security needed everywhere with nonce misuse (idem with decryption leakages)

• Others: Gimli, Ketje, Oribatida, ...
  • (Roughly applies to all inner-keyed sponges)
Outline

1. Side-channel (crypt)analysis: attacks taxonomy
2. Masking countermeasure: security vs. cost
3. Security definitions (authenticated encryption)
   a. Nonce-respecting setting (i.e., AEL)
   b. Nonce-misuse setting (i.e., AEmL)

1. Leakage-resistant AE designs (& implementations)
   • Level 0: no mode-level leakage-resistance
   • Level 1: re-keyed modes (including sponges)
   • Level 2: level 1 + strengthened init./final.
   • Level 3: level 2 + two-passes

2. Conclusions (& the need of open evaluations)
Ascon [DEMS19] (confidentiality)

• Target: CCAL1 (L in enc only, no misuse)

• Similar to inner-keyed sponges
Ascon [DEMS19] (confidentiality)

- Target: CCAmL1 (L in enc only, misuse-resilience)

- Strengthened init./final. steps maintain the SPA resistance requirement for the bulk computation with nonce misuse and encryption leakages
Ascon [DEMS19] (confidentiality)

- Target: CCAmL2 (L in enc/dec, misuse-resilience)

- Limited confidentiality with decryption leakages
- Dark orange/green: message decrypted before verification ⇒ the same state can be repeatedly measured, allowing SPA with averaged leakage
• Target: CIL1 (L in enc only, no misuse)

• Bulk computation leakage can be unbounded
• Shows interest of composite definitions!

Ascon [DEMS19] (integrity)
• Target: CIML1 (L in enc only, misuse-resistance)

Ascon [DEMS19] (integrity)

• Same feature (unbounded leakages for the bulk)
• Target: CIML2 (L in enc/dec, misuse-resistance)

• Tag verification must be protected against DPA
• Shows key recovery security is not enough!
• Target: CIML2 (L in enc/dec, misuse-resistance)

• Tag verification must be protected against DPA

• Shows key recovery security is not enough!

• Others: ACE, GIBBON, Spix, WAGE, ...
• CCAL1, CCAmL1

≈ further exploiting the leveled implementation concept

• Similar to ASCON (but smaller masked state)
Spook [B+19] (confidentiality)

- **CCAmL2**

  $P|\|N$  
  $E_K$  
  $N$  
  $P$

  $M_1$  $C_1$  $M_2$  $C_2$  $M_3$  $C_3$

  $\approx$ further exploiting the leveled implementation concept
  - **Similar to ASCON** (but smaller masked state)
• CIL1, CIML1

≈ further exploiting the leveled implementation concept

• Similar to ASCON (but smaller masked state)
Spook [B+19] (integrity)

- **CIML2** (L in enc/dec, misuse-resistance)

- Tag verification tolerates unbounded leakages
- (Inverse-free DPA resistant tag verif. also possible)

- Others: TBC-only variant (TET)
1. Side-channel (crypt)analysis: attacks taxonomy

2. Masking countermeasure: security vs. cost

3. Security definitions (authenticated encryption)
   a. Nonce-respecting setting (i.e., AEL)
   b. Nonce-misuse setting (i.e., AEmL)

1. Leakage-resistant AE designs (& implementations)
   • Level 0: no mode-level leakage-resistance
   • Level 1: re-keyed modes (including sponges)
   • Level 2: level 1 + strengthened init./final.
   • Level 3: level 2 + two-passes

2. Conclusions (& the need of open evaluations)
• CCAmL2 (L in enc/dec, misuse-resilience)

• 2 pass ⇒ confidentiality in dec. if DPA-resistant verif.
• **CCAmL2 (L in enc/dec, misuse-resilience)**

\[ P||N \xrightarrow{E_K} N \xrightarrow{\pi} M_1 \xrightarrow{C_1} M_2 \xrightarrow{C_2} M_3 \xrightarrow{C_3} \]

• **Tag verification with unbounded leakages**
• ∃ a tradeoff between mode-level and implementation leakage-resistance
• As the security target and level increase, mode-level leakage-resistance gains more interest
- ∃ a tradeoff between mode-level and implementation leakage-resistance
- As the security target and level increase, mode-level leakage-resistance gains more interest
- Performance gains of levelled implementations
• ∃ a tradeoff between mode-level and implementation leakage-resistance
• As the security target and level increase, mode-level leakage-resistance gains more interest

• Performance gains of levelled implementations
1. Side-channel *(crypt)*analysis: attacks taxonomy
2. Masking *countermeasure*: security vs. cost
3. Security *definitions* (authenticated encryption)
   a. Nonce-respecting setting (i.e., AEL)
   b. Nonce-misuse setting (i.e., AEmL)

1. Leakage-resistant AE *designs* (& implementations)
   • Level 0: no mode-level leakage-resistance
   • Level 1: re-keyed modes (including sponges)
   • Level 2: level 1 + strengthened init./final.
   • Level 3: level 2 + two-passes

2. Conclusions (& the need of open evaluations)
• Overall, $\exists$ a wide zoo of definitions including
  • Leakage-resilience vs. leakage-resistance
  • Misuse-resilience vs. misuse-resistance
  • Leakage in encryption and decryption
  • For integrity and confidentiality
Overall, there is a wide zoo of definitions including:

- Leakage-resilience vs. leakage-resistance
- Misuse-resilience vs. misuse-resistance
- Leakage in encryption and decryption
- For integrity and confidentiality

Not black & white notions: all security notions can be reached using more demanding physical assumptions:

- Best solutions to reach each target have to be evaluated
  - Which requires (tight) bounds and concrete (primitive-dependent) security evaluations
A theory to guide practice?

5.1

- Overall, ∃ a wide zoo of definitions including
  - Leakage-resilience vs. leakage-resistance
  - Misuse-resilience vs. misuse-resistance
  - Leakage in encryption and decryption
  - For integrity and confidentiality

- Not black & white notions: all security notions can be reached using more demanding physical assumptions
  - Best solutions to reach each target have to be evaluated
    - Which requires (tight) bounds and concrete (primitive-dependent) security evaluations

⇒ Hope: strong assumptions in the proofs/analyzes indicate where implementers must put most efforts
Open problems

• We have good ingredients ⇒ how to mix them?
• Evaluation of AE schemes for various security targets
• Links between the different security notions
• Graceful degradations (for CIML2, CCAmL2)
• Proofs under weaker physical assumptions
• Application to signatures/PKE?
• Cipher designs / key-homomomorphic primitives
• Masking (physical defaults, composition, ...)
• Improved confidentiality for 1-block messages
• Prototype (open source) implementations
• Anything leading to simple(r) hardware guidelines...
Evaluation challenge

standard practice

evidence-based evaluations
(assumptions tested per device!)

bounds

2^{10} 2^{20} 2^{30}

computations

measurements

success probability

5.3
Evaluation challenge

standard practice

evidence-based evaluations
(*assumptions tested per device!*)

\[
2^{128} \quad 2^{64} \quad 2^{30} \quad 2^{10} \quad 2^0
\]

success probability

bound
Evaluation challenge

standard practice

open design & evaluation

evidence-based evaluations on reduced versions

proof-based evaluations [DFS15,GS18]

bounds

computed success probability

tighter bounds
THANKS

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