Towards an Open Approach to Secure Cryptographic Implementations

François-Xavier Standaert
UCLouvain, ICTEAM, Crypto Group (Belgium)
EUROCRYPT 2019, Darmstadt, Germany
Transparency (as a measure of maturity)

- Block ciphers & symmetric encryption
Transparency (as a measure of maturity)

- Secure cryptographic 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)
4. Leakage-resistant AE *designs* (& implementations)
5. Conclusions (& the need of open evaluations)
Acknowledgments

C. Archambeau  B. Gierlichs  T. Malkin  K. Pietrzak
J. Balasch  C. Glowacz  S. Mangard  R. Poussier
G. Barthe  D. Goudarzi  D. Masny  E. Prouff
S. Belaïd  B. Grégoire  C. Massart  F. Regazzoni
D. Bellizia  V. Grosso  P. Méaux  M. Renauld
F. Berti  S. Guilley  M. Medwed  O. Reparaz
O. Bronchain  T. Güneysu  C. Momin  M. Rivain
G. Cassiers  Chun Guo  A. Moradi  T. Schneider
C. Dobraunig  Qian Guo  M. Naya-Plasencia  P. -Y. Strub
A. Duc  G. Herold  A. Olshevsky  N. Veyrat-Charvillon
F. Dupressoir  A. Journault  Y. Oren  S. Vivek
F. Durvaux  D. Kamel  E. Oswald  Weijia Wang
S. Duval  G. Leander  C. Paglialonga  C. Whitnall
S. Dziembowski  L. Lerman  O. Pereira  Yu Yu
S. Faust  G. Leurent  T. Peters  M. Yung
P.-A. Fouque  I. Levi  C. Petit

IV
Acknowledgments & cautionary note

- Mixing (very) different abstraction levels
  - Hopefully in a consistent manner *(be forgiving if not)*
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)

4. Leakage-resistant AE **designs** (& implementations)

5. Conclusions (& the need of open evaluations)
AES Rijndael: $y = AES_K(x)$
Leaking AES: $y = \text{AES}_K(x) \rightarrow L$
Leakage function definition

• Leakage 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: \( \text{SNR}_i = \frac{\text{var}(\delta^i_x)}{\text{var}(N^i)} \)
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: \( \text{SNR}^i = \frac{\text{var}(\delta^i_x)}{\text{var}(N^i)} \)

• 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>$\lambda$ (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>

$\log(32) = 5$ 

$\lambda = 25,000 \left(1 + \frac{1}{\text{SNR}}\right)$
Basic facts (I)

• 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>( \frac{\log(32)}{1 + \frac{1}{\text{SNR}}} )</td>
<td>( \frac{25,000}{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>( \frac{\log(128)}{1 + \frac{1}{\text{SNR}}} )</td>
<td>( \frac{350}{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 \textit{offline leakage oracle} $L(\cdot, K)$:

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

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

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

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

$\Rightarrow$ Distinguishing games without anything fresh and secret in the challenge are trivial to win
Basic facts (II)

• Key recovery attacks may not easily exploit all leakage samples (since $\mathcal{A}$ 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
• 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>$\approx \left(\frac{14}{100}\right)^d$</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>$\log(128) = 7$</td>
<td>$\frac{14}{100}^d$</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):

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

\[\Rightarrow\text{Typical success probability (e.g., for key recovery):}\]

\[\Pr[A_{KR}(x_1, L(x_1, K), \ldots, x_q, L(x_q, K)) \rightarrow K|K ← $] \approx 2^{-128+q⋅λ(r)}\]

- There are two main types of attacks (jargon)
  - \textbf{SPA}: \(q\) is a small constant (e.g., thanks to re-keying)
  - \textbf{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)

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

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

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

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

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

- Message Comparison (MC) attacks (with fresh challenge)
  \[
  \Pr \left[ A_{MC}^{L(x)} \left( x_0, x_1, L(x_b, K) \right) = b \mid K, b \leftarrow \$ \right] \approx 2^{-128 + D(L(x_0,K);L(x_1,K))}
  \]
  - 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(x), K} \left( x_0, x_1, L(x_b, K) \right) = b \mid K, b \leftarrow \$ \right] \approx 1 \text{ anyway}
1. Side-channel \textbf{(crypt)analysis}: attacks taxonomy

2. Masking \textbf{countermeasure}: security vs. cost

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

4. Leakage-resistant AE \textbf{designs} (& implementations)

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

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

- Additive noise $\approx \text{cost} \times 2 \Rightarrow \text{security} \times 2$
- $\Rightarrow$ not a good (crypto) security parameter
- $\approx$ same holds for all hardware countermeasures
Masking (≈ noise amplification)

- 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]

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

serial implementation.

- Noisy leakage security [PR13]
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]

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

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

Masking (reduction)

[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
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

\[
\begin{bmatrix}
  a_1 b_1 & a_1 b_2 & a_1 b_3 \\
  a_2 b_1 & a_2 b_2 & a_2 b_3 \\
  a_3 b_1 & a_3 b_2 & a_3 b_3
\end{bmatrix}
\]

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

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

partial products refreshing
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 refreshment compression
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 **refreshing** compression

\( a_1 b_1 \oplus a_1 b_2 \oplus a_1 b_3 = a_1 b \) 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 \quad \text{refreshing} \quad \text{compression}

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

\[ \Rightarrow \text{Quadratic overheads & randomness} \]
• (Many published optimizations [R+15,Be+16,GM18])
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]

![Graph showing various security levels and their relationship to SNR](image-url)
Case study: ARM Cortex M4 [JS17]

Graph 1: Logarithmic scale for security

Graph 2: Cycles per byte vs. number of shares

Security

Performance
Case study: ARM Cortex M4 [JS17]

security

performance

The top graph shows the relationship between log_{10}(SNR) and log_{10}(MI) for different security orders and bit sizes.

The bottom graph illustrates the performance in cycles per byte as a function of the 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)
• 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
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)

4. Leakage-resistant AE designs (& implementations)

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

- $A$ cannot ask a decryption query on $(N, AD, C)$ after $C$ is returned by an $(N, AD, .)$ encryption query.
Authenticated Encryption (AEAD)

- Why not extending [RS06]'s *all in one* definition?

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

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

\[ E_K(.,.,.,.) \rightarrow N,AD,M \leftarrow N,AD,C \rightarrow D_K(.,.,.,.) \]

\[ C \rightarrow fresh\ C^* \rightarrow M \]
Ciphertext Integrity with Leakage

- CIL1: leakage in encryption only [Be+18]
• CIL2: leakage in encryption and decryption [BPPS17]
• Natural extensions (no definitional challenges) with many applications (e.g., secure bootloading)
Chosen Ciphertext Security

\[ N, AD, M \]
\[ C \]
\[ N, AD, C \]
\[ M \]
\[ N^*, AD, M_0, M_1 \]
\[ C^* = E_K(N^*, AD, M_b) \]
\[ N, AD, M \]
\[ C \]
\[ N, AD, C \]
\[ M \]
\[ (K, b) \]
\[ E_K(\ldots, \ldots) \]
\[ D_K(\ldots, \ldots) \]
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 $L_{dec}^*$ (applications: IP protection, ...)

$$C^* = E_K(N^*, AD, M_b), L_{enc}^*$$
• [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
• [MR04] (and [NS09,BG10,...]): indistinguishability with $L_{enc^*}$ 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)
4. Leakage-resistant AE **designs** (& implementations)
5. 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 can ensure leakage-resistance and (nonce) misuse-resistance (excluding trivial / fully leak-free solutions)
For confidentiality, no meaningful encryption scheme can ensure leakage-resistance and (nonce) misuse-resistance (excluding trivial / fully 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 can ensure leakage-resistance and (nonce) misuse-resistance (excluding trivial / fully 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 can ensure leakage-resistance and (nonce) misuse-resistance (excluding trivial / fully 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

• Strongest (*leak.-resist.*) def.: AEML=CIML2+CCAmL2+MR
Summarizing

- For confidentiality, no meaningful encryption scheme can ensure leakage-resistance and (nonce) misuse-resistance (excluding trivial / fully 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 (\textit{leak.-resist.}) def.: AEML=CIML2+CCAmL2+MR

- Weaker variants can be meaningful: for instance AEmL=CIML2+CCAmL2 [Be+19], CPA\textsubscript{1} [DM19], ...
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)

4. Leakage-resistant AE designs (& implementations)

5. Conclusions (& the need of open evaluations)
• Forgeries can exploit two attack paths
  • a DPA against the long-term key $K$
  • a DPA against the tag verification $\tau = \tau_c$?
    • By monitoring the comparison with random tags
• Leakage on the dotted parts can be unbounded
Forgeries can exploit two attack paths
  - a DPA against the long-term key $K$
  - a DPA against the tag verification $\tau = \tau_c$?
    - By monitoring the comparison with random tags

Leakage on the dotted parts can be unbounded

Can we « minimize the attack » surface?
First tweak: LR tag verification

- Natural option: inverse-based tag verification
  - Only performs comparisons with a public $h$
- So leakage can be unbounded for this part too
  - Beneficial: good leakage assumptions hard to find
Natural option: inverse-based tag verification
- Only performs comparisons with a public $h$
- So leakage can be unbounded for this part too
- Beneficial: good leakage assumptions hard to find

How to generalize this to an AE scheme?
• First ignoring confidentiality with leakage

• Many parts of the design can leak in full
  • Strong motivation for composite definitions: allow using the weakest possible assumptions for integrity and confidentiality (which are not the same)
• First ignoring confidentiality with leakage

• Many parts of the design can leak in full
  • Strong motivation for composite definitions: allow using the weakest possible assumptions for integrity and confidentiality (which are not the same)

• How to add confidentiality guarantees?
Engineering approach to CCAL security

- Requires protecting all the BC blocks against DPA
- And to deal with the (unavoidable) MC attacks
• Requires protecting all the BC blocks against DPA
• And to deal with the (unavoidable) MC attacks
• Typically leads to (very) expensive implementations
A CCAmL2 encryption scheme

- Most BC executions can be protected against SPA only (+ two DPA-secure BC calls and security against MC attacks)
full fledged scheme

- Formally, modeled as leak-free
- Graceful degradation seems possible

KR (DPA) security of two BC executions
Security reductions (simplified)

full fledged scheme

CCAm12 assumptions [Be+19]

unbounded leakage model

KR (SPA) & MC security of one PRG iteration

KR (DPA) security of two BC executions

• Formally, modeled as leak-free
• Graceful degradation seems hard
Security reductions (simplified)

**full fledged scheme**

- Formally, modeled as leak-free
- Graceful degradation seems hard

**CCAmL2 assumptions’ [GPPS19]**

**KR (SPA) & MC security of one PRG iteration**

**KR (DPA) security of two BC executions**
Example of full-fledged scheme

- **S1P:** 1-pass (online), CIML2, CCAmL1 [GPPS19]

- Encourages « leveled implementations »
  - Strongly protected TBC: high-order masking
  - Weakly protected permutation: low-latency
- For such implementations, two different primitives are not an issue (since implementations are different)
Example of full-fledged scheme

- **S1P**: 1-pass (online), CIML2, CCAmL1 [GPPS19]

- Encourages « leveled implementations »
  - Strongly protected TBC: high-order masking
  - Weakly protected permutation: low-latency
  - For such implementations, two different primitives are not an issue (since implementations are different)

- (+ beyond birthday w.r.t. TBC key, multi-user security)
Example of full-fledged scheme

- **S1P: 1-pass** (online), CIML2, CCAmL1 [GPPS19]

- **Performance gains of leveled implementations**

![Diagram of a 100-byte message and cycles per byte (32-bit ARM) vs. number of shares](image)
Example of full-fledged scheme

- **S1P**: 1-pass (online), CIML2, CCAmL1 [GPPS19]

- Performance gains of leveled implementations

![Diagram of S1P scheme with cycle per byte vs number of shares for 1K-byte message]
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)
4. Leakage-resistant AE **designs** (& implementations)
5. 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
A theory to guide practice?

5.1

• 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

• Not black & white notions, e.g., $\text{CIML2}$ is obtained
  • With inv. verification in the unbounded leakage model
  • With direct verification if it is secure against DPA
    • What is best in practice still has to be evaluated
A theory to guide practice?

- 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, e.g., CIML2 is obtained
  - With inv. verification in the unbounded leakage model
  - With direct verification if it is secure against DPA
    - What is best in practice still has to be evaluated

⇒ Hope: strong assumptions in the proofs/analyzes indicate where implementers must put most efforts
• We have good ingredients ⇒ how to mix them?
Open problems

- We have good ingredients ⇒ how to mix them?
- Classification of existing AE schemes
  - E.g., NIST lightweight competition candidates
- Links between the different security notions
- Graceful degradations (for CIML2, CCAmL2)
- Proofs under weaker physical assumptions
- Application to signatures/PKE?
Open problems

• We have good ingredients ⇒ how to mix them?
• Classification of existing AE schemes
  • E.g., NIST lightweight competition candidates
• Links between the different security notions
• Graceful degradations (for CIML2, CCAML2)
• Proofs under weaker physical assumptions
• Application to signatures/PKE?
• Cipher designs / key-homomorphic primitives
• Masking (physical defaults, composition, ...)

5.2
Open problems

- We have good ingredients ⇒ how to mix them?
- Classification of existing AE schemes
  - E.g., NIST lightweight competition candidates
- 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
Open problems

• We have good ingredients ⇒ how to mix them?
• Classification of existing AE schemes
  • E.g., NIST lightweight competition candidates
• Links between the different security notions
• Graceful degradations (for CIML2, CCAmL2)
• Proofs under weaker physical assumptions
• Application to signatures/PKE?
• Cipher designs / key-homomorphic 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

success probability
Evaluation challenge

standard practice

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

bounds

> $2^{30}$

= $2^{40}$?

= $2^{80}$?
Evaluation challenge

standard practice

open design & evaluation

evidence-based evaluations on reduced versions

proof-based evaluations [DFS15,GS18]
THANKS

http://perso.uclouvain.be/fstandae/
SUPPLEMENTARY SLIDES
Scalability & composability

$t$-probing security [ISW03]
any $t$-tuple of shares in the protected circuit is independent of any sensitive variable
Scalability & composability

**Problem:**

The cost of testing probing security increases (very) fast with circuit size and the # of shares ($\exists$ many tuples) [Ba+15].

**t-probing security** [ISW03]

Any $t$-tuple of shares in the protected circuit is independent of any sensitive variable.

**Problem:**

The cost of testing probing security increases (very) fast with circuit size and the # of shares ($\exists$ many tuples) [Ba+15].
Scalability & composability

\[ q_1 + q_2 \leq q \]

- \( q_1 \) internal probes
- \( q_2 \) output probes

Problem: the cost of testing probing security increases (very) fast with circuit size and the # of shares (\( \exists \) many tuples) [Ba+15]

**q-(Strong) Non Interference** [Ba+16]: a circuit gadget (e.g., \( f_1 \)) is (Strongly) Non-Interferent if any set of \( q_1 + q_2 \) probes can be simulated with at most \( q_1 + q_2 \) (only \( q_1 \)) shares of each input.

\[
D(\text{input shares} || \text{probes}) \approx D(\text{input shares} || \text{simulation})
\]
Separation result (simplified)

• Why CI+CCA (while in black box: CI+CPA = PI+CCA)?
• Why CI+CCA (while in black box: CI+CPA = PI+CCA)?

• Let AEAD be CIML2 & CCAML2 with $L_{enc}$ and $L_{dec}$
  • Define AEAD’ such that
    • $L_{enc}'(K, M) = L_{enc}(K, M) + L_{enc}(K', M')$
    • $L_{dec}'(K, C) = L_{dec}(K, C) + L_{enc}(K', M')$
  • AEAD’ is still CIML2 but not CCAML2 anymore
  • Attack: use the $L_{dec}'$ query to leak about $M'$ and then use $M'$ as challenge plaintext
Separation result (simplified)

• Why CI+CCA (while in black box: CI+CPA = PI+CCA)?

• Let AEAD be CIML2 & CCAmL2 with $L_{enc}$ and $L_{dec}$
  • Define AEAD’ such that
    • $L_{enc}'(K, M) = L_{enc}(K, M) + L_{enc}(K', M')$
    • $L_{dec}'(K, C) = L_{dec}(K, C) + L_{enc}(K', M')$
  • AEAD’ is still CIML2 but not CCAmL2 anymore
  • Attack: use the $L_{dec}'$ query to leak about $M'$ and then use $M'$ as challenge plaintext

• First apparition of a recurring issue: somewhat artificial attack related to the difficulty to model $L$
Analyses must deal with the **information** and the **computational** complexity of the leakage function.
• Analyses must deal with the **information** and the **computational** complexity of the leakage function
  
  • Information: obvious (full leakage ⇒ no secrecy)
  • Computation: avoid « precomputation attacks » [DP08]
Analyses must deal with the **information** and the **computational** complexity of the leakage function.

- **Information**: obvious (full leakage \(\Rightarrow\) no secrecy)
- **Computation**: avoid « precomputation attacks » [DP08]

(Second apparition of a recurring issue: somewhat artificial attack related to the difficulty to model \(L\))
Analyses must deal with the information and the computational complexity of the leakage function:

- Information: obvious (full leakage ⇒ no secrecy)
- Computation: avoid « precomputation attacks » [DP08]

Background: the shape of $L$ is unknown:

- We don’t even know its complexity class, e.g.,
  - Solving Maxwell’s equations for an AES circuit takes days
  - But a physical circuit provides an instantaneous answer
Physical assumptions (symmetric SOTA)

- Information restriction

  - HILL pseudoentropy [DP08]

  - Seed-preserving PRG [YPSM10] (≈ hard to invert leakages [DKL09])
Physical assumptions (symmetric SOTA)

- Information restriction + computation restriction

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>HILL pseudoentropy [DP08]</td>
<td>😞</td>
</tr>
<tr>
<td>only computation leaks [MR04] + alternating structure [DP08]</td>
<td>😞</td>
</tr>
<tr>
<td>seed-preserving PRG [YPSM10] (≈ hard to invert leakages [DKL09,D+10])</td>
<td>☺</td>
</tr>
<tr>
<td>oracle-free leakage function [YPMS10]</td>
<td>😞</td>
</tr>
</tbody>
</table>
Physical assumptions (symmetric SOTA)

- Information restriction + computation restriction

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>HILL pseudoentropy [DP08]</td>
<td>😞</td>
</tr>
<tr>
<td>only computation leaks [MR04] + alternating structure [DP08]</td>
<td>😞</td>
</tr>
<tr>
<td>Simulatibility [SPY13]</td>
<td>😞</td>
</tr>
<tr>
<td>Seed-preserving PRG [YPSM10]</td>
<td>😊</td>
</tr>
<tr>
<td>Oracle-free leakage function [YPMS10]</td>
<td>😞</td>
</tr>
</tbody>
</table>
Physical assumptions (symmetric SOTA)

- Information restriction + computation restriction
- Safer strategy: try proofs with both combinations
  - Weaker physical assumption but idealized analysis
  - Stronger physical assumption in the standard model

<table>
<thead>
<tr>
<th>Simulatability [SPY13]</th>
<th>😞</th>
</tr>
</thead>
<tbody>
<tr>
<td>∅</td>
<td>☺</td>
</tr>
</tbody>
</table>

- Seed-preserving PRG [YPSM10] (*hard to invert leakages [DKL09,D+10]*)
- Oracle-free leakage function [YPMS10]
Simulatable leakage (I)

- \( L \) is hard to model \( \Rightarrow \) just don’t model it
- Give public I/O access to device & setup
• $L$ is hard to model $\Rightarrow$ just don’t model it
• Give public I/O access to device & setup

- Assume $L(x, K)$ can be simulated
  - Using the same hardware as the target
  - But without knowing the secret key $K$
• $L(x, K)$ can be simulated without knowledge of $K$

• E.g., FPGA implementation, 128-bit architecture

Simulatable leakages (II)

$x \rightarrow$ rounds (with fresh & unknown data) $\rightarrow y = BC_K(x)$
Simulatable leakages (II)

- $L(x, K)$ can be simulated without knowledge of $K$
  - E.g., FPGA implementation, 128-bit architecture

Simulated traces should be consistent with $x$ and $y$

$x$ rounds (with fresh & unknown data) $y = BC_K(x)$
Simulatable leakages (II)

- \( L(x, K) \) can be simulated without knowledge of \( K \)
- E.g., FPGA implementation, 128-bit architecture

Simulated traces should be consistent with \( x \) and \( y \)

- Simple proposal [SPY13]: « split & concatenate »

\[
S(x, K, y) = L^{1/2} (x, K^*, y^*) || L^{2/2} (x^*, K^*, y)
\]
The Longo et al. distinguisher [Lo+14]

- Intra-trace correlation: $\rho(L^t_i, L^{1:t_{2500}})$, real traces
The Longo et al. distinguisher [Lo+14]

- Intra-trace correlation: \( \rho(L^{t_i}, L^{1:t_{2500}}) \), real traces

- Same, with simulated traces \( L^{1/2}(x, K^*, y^*) \parallel L^{2/2}(x^*, K^*, y) \)
• Intra-trace correlation: $\rho(L^t_i, L^{1:t_{2500}})$, real traces

![Cross-correlation plot]

• Same, with simulated traces $L^{1/2}(x, K^*, y^*) \| L^{2/2}(x^*, K^*, y)$

![Cross-correlation plot]

• Fixing this requires modeling the physics of $L$ (but the goal of simulatability was to avoid such modeling)
Another (new) approach

- Just look (exhaustively) for a key $K^*$ such that
  \[ BC_{K^*}(x) = \tilde{y} \rightarrow l_{\tilde{y}} \] and $e = |l_y - l_{\tilde{y}}|$ is small
Another (new) approach

• Just look (exhaustively) for a key $K^*$ such that $BC_{K^*}(x) = \tilde{y} \rightarrow l_{\tilde{y}}$ and $e = |l_y - l_{\tilde{y}}|$ is small

- Add key-dep. noise s.t. $\forall x, K, K^*, E(l_\Delta) \neq E(l_{\Delta^*})$
• Assume a perfect (additive) leakage model
• Remove the contribution $l_y$ for each $x, K$ pair

**real distributions**

- $l_{\Delta}$
- $\mu$

**simulated distributions**

- $l_{\Delta^*} + (l_y - l_{\tilde{y}})$
- $\mu - e$
- $\mu + e$
• Assume a perfect (additive) leakage model
• Remove the contribution $l_y$ for each $x, K$ pair

\[ l_\Delta \]

\[ l_\Delta^* + (l_y - \tilde{l}_y) \]

• The difference btw. variances decreases in $e^2$
• Assume a perfect (additive) leakage model
• Remove the contribution $l_y$ for each $x, K$ pair

real distributions

$\Delta$

$\mu$

simulated distributions

$\Delta^* + (l_y - \tilde{l}_y)$

$\mu - e \quad \mu + e$

• The difference btw. variances decreases in $e^2$
• Error $e$ decreases linearly in sim. complexity $C$
  • Depending on the leakage function (experiments needed)
• For key-dep. noise variance $\sigma_\Delta^2$ and simulator error $\varepsilon$, the simulation is distinguished in $\geq \frac{(\sigma_\Delta^2)^2}{\varepsilon^2}$ traces
Distinguishing complexity

- For key-dep. noise variance $\sigma_\Delta^2$ and simulator error $e$, the simulation is distinguished in $\geq \frac{(\sigma_\Delta^2)^2}{e^2}$ traces.

- For example (experimental data):
  - Key-dep. noise variance $\sigma_\Delta^2 = 32$ (ignoring $+\sigma_n^2$)
  - Simulator complexity $C = 2^c = 2^{32}$
  - Simulator error $e = 2^{-27}$
  - Distinguishing complexity $\geq 2^{64}$
For key-dep. noise variance $\sigma^2_{\Delta}$ and simulator error $e$, the simulation is distinguished in $\geq \frac{(\sigma^2_{\Delta})^2}{e^2}$ traces.

For example (experimental data):
- Key-dep. noise variance $\sigma^2_{\Delta} = 32$ (ignoring $+\sigma^2_n$)
- Simulator complexity $C = 2^c = 2^{40}$
- Simulator error $e = 2^{-34}$
- Distinguishing complexity $\geq 2^{78}$
For key-dep. noise variance $\sigma^2_\Delta$ and simulator error $e$, the simulation is distinguished in $\geq \frac{(\sigma^2_\Delta)^2}{e^2}$ traces.

For example (experimental data)

- Key-dep. noise variance $\sigma^2_\Delta = 32$ (ignoring $+\sigma^2_n$)
- Simulator complexity $C = 2^c = 2^{40}$
- Simulator error $e = 2^{-34}$
- Distinguishing complexity $\geq 2^{78}$

$\implies$ Reductions: $k$-bit key gives $(k-c)$-bit confidentiality
• For key-dep. noise variance $\sigma_\Delta^2$ and simulator error $e$, the simulation is distinguished in $\geq \frac{(\sigma_\Delta^2)^2}{e^2}$ traces.

• For example (experimental data):
  • Key-dep. noise variance $\sigma_\Delta^2 = 32$ (ignoring $+\sigma_n^2$)
  • Simulator complexity $C = 2^c = 2^{40}$
  • Simulator error $e = 2^{-34}$
  • Distinguishing complexity $\geq 2^{78}$

$\Rightarrow$ Reductions: $k$-bit key gives $(k-c)$-bit confidentiality

• (Imperfect model: $l_\Delta + e'$ and $l_\Delta^* + e + e'$ get closer)
  
  real

  simulation
• [DP08] require $H^\text{HILL}(K|L) > k - \lambda$
• Roughly: $\forall l, \exists$ a set of $2^{k-\lambda}$ keys s.t. $BC_{K^*}(x) \rightarrow \tilde{l} \overset{c}{\approx} l$
Simulatability vs. pseudoentropy

- [DP08] require $H^{\text{HILL}}(K | L > k - \lambda$
- Roughly: $\forall l, \exists$ a set of $2^{k-\lambda}$ keys s.t. $BC_{K^*}(x) \rightarrow \tilde{l} \approx l$
- (Without the comp. limit., SPA samples become SC samples)
Simulatability vs. pseudoentropy

- [DP08] require $H^{\text{HILL}}(K|L) > k - \lambda$
- Roughly: $\forall l$, $\exists$ a set of $2^{k-\lambda}$ keys s.t. $BC_{K^*}(x) \rightarrow \tilde{l} \approx l$
- *(Different parts of the leakages raise different challenges)*
Simulatability vs. pseudoentropy

- [DP08] require $H^{HILL}(K|L) > k - \lambda$
- Roughly: $\forall l, \exists$ a set of $2^{k-\lambda}$ keys s.t. $BC_{K^*}(x) \rightarrow \tilde{l} \approx l$
- (Different parts of the leakages raise different challenges)

$\Rightarrow$ a fraction $\frac{1}{2^{\lambda}}$ of the keys can be used to simulate

- Theory [FH15]: $q$-SIM and $H^{HILL}$ disconnected
- Cryptanalysis: $q$-SIM $\leq H^{HILL} \ll$ CCAmL(for 1 block)
Simulatable leakage definition

- More formally, \((k, A)\) has \(q\)-simulatable leakages of \(\exists\) a simulator \(S^L(\cdot)\) such that the bit \(b\) in the following game is hard to guess

| Game \(q\)-sim\((A, k, S^L(\cdot), b)\) with \(K, K^*\) uniformly random |
|---------------------------------|---------------------|---------------------|
| \(q\) queries | response if \(b = 0\) | response if \(b = 1\) |
| \(\text{enc}(x)\) | \(y = BC_k(x), L(x, K)\) | \(y = BC_k(x), S^L(\cdot)(x, K^*, y)\) |
| 1 query | response if \(b = 0\) | response if \(b = 1\) |
| \(\text{gen}(z, x)\) | \(S^L(\cdot)(z, x, K)\) | \(S^L(\cdot)(z, x, K^*)\) |

- (Not exactly real vs. simulated due to the gen query)
\[ \rightarrow L(x, K_0, K_1) \parallel L(x, K_1, K_2) \]
Proof intuition

\[ \rightarrow L(x, K_0, K_1) \parallel L(x, K_1, K_2) \]

\[ \rightarrow S(x, K_0^*, K_1) \parallel L(x, K_1, K_2) \]
Proof intuition

\[ \rightarrow L(x, K_0^*, K_1) \| L(x, K_1^*, K_2) \]

\[ \rightarrow S(x, K_0^*, K_1^*) \| L(x, K_1^*, K_2) \]
Proof intuition

\[ \rightarrow L(x, K_0, K_1) \parallel L(x, K_1, K_2) \]

\[ \rightarrow S(x, K_0^*, K_1) \parallel L(x, K_1, K_2) \]

\[ \rightarrow S(x, K_0^*, K_1^*) \parallel L(x, K_1, K_2) \]

\[ \rightarrow S(x, K_0^*, K_1^*) \parallel S(x, K_1, K_2) \]


