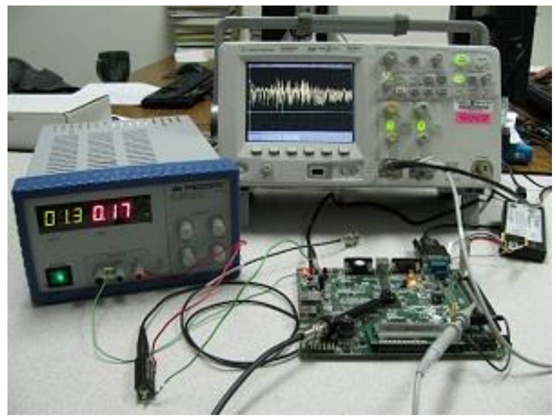
Towards an Open Approach to Secure Cryptographic Implementations



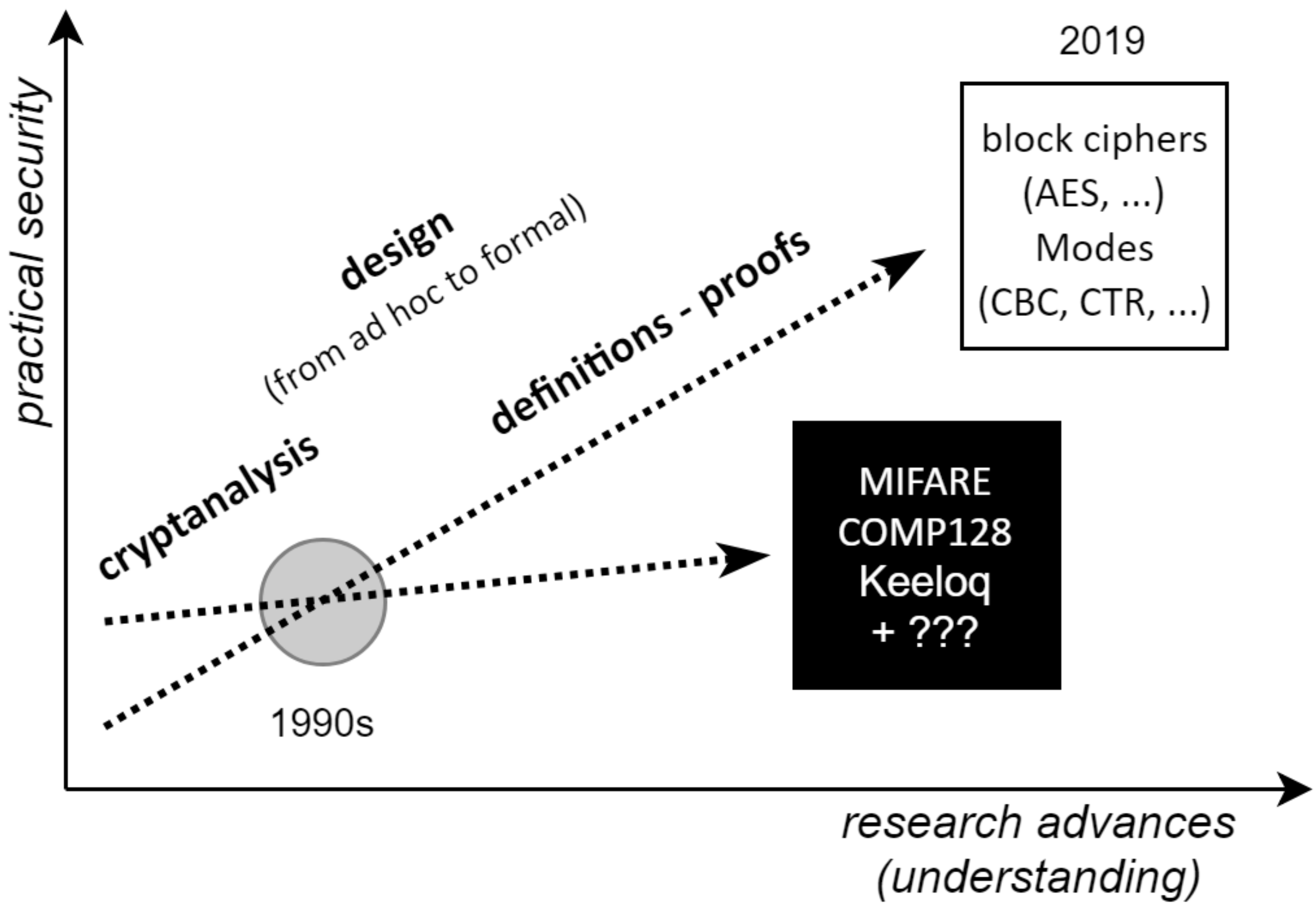
François-Xavier Standaert

UCLouvain, ICTEAM, Crypto Group (Belgium)

EUROCRYPT 2019, Darmstadt, Germany

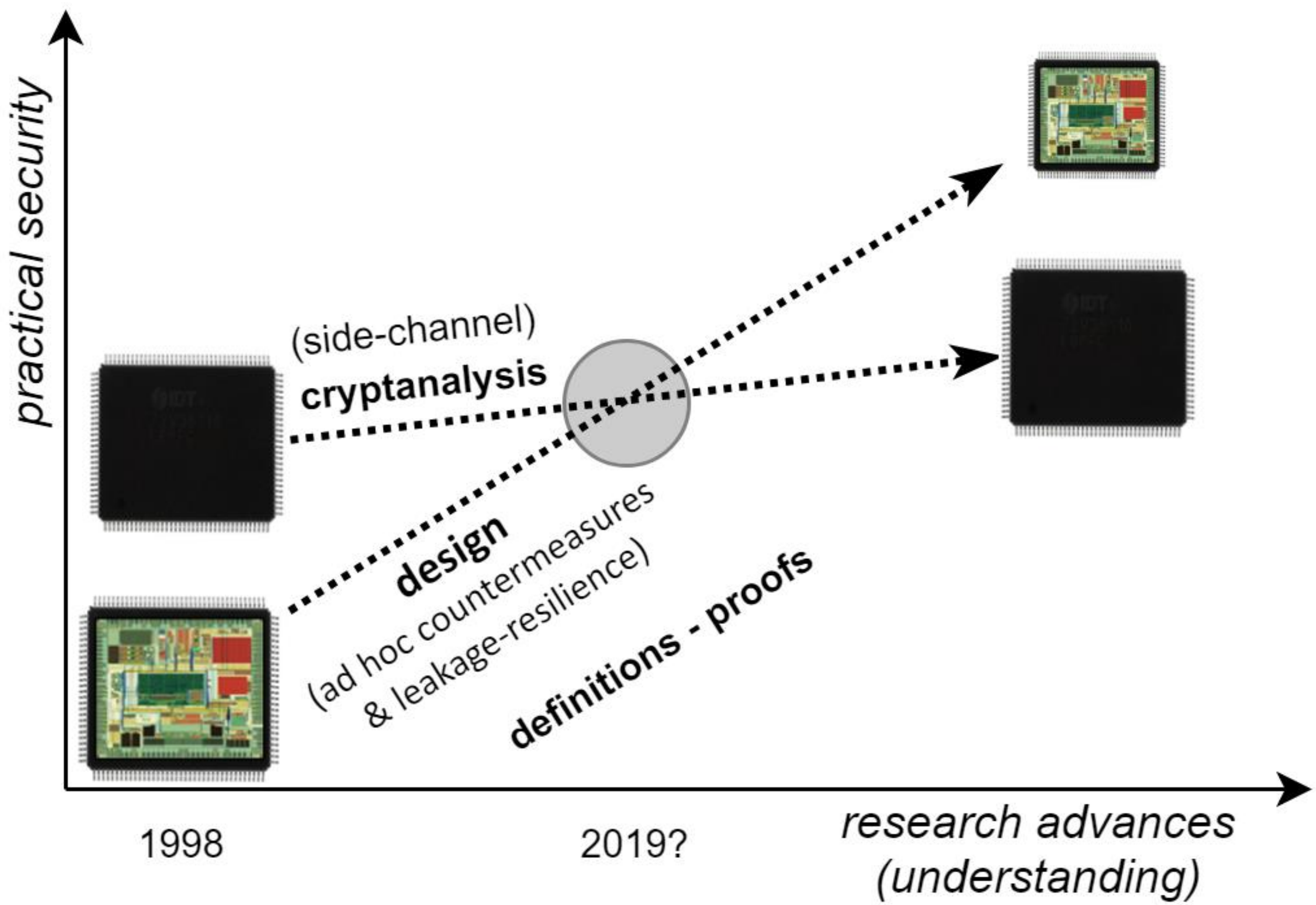
Transparency (as a measure of maturity)

- Block ciphers & symmetric encryption



Transparency (as a measure of maturity) II

- Secure cryptographic implementations



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)

Acknowledgments

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

Acknowledgments & cautionary note IV

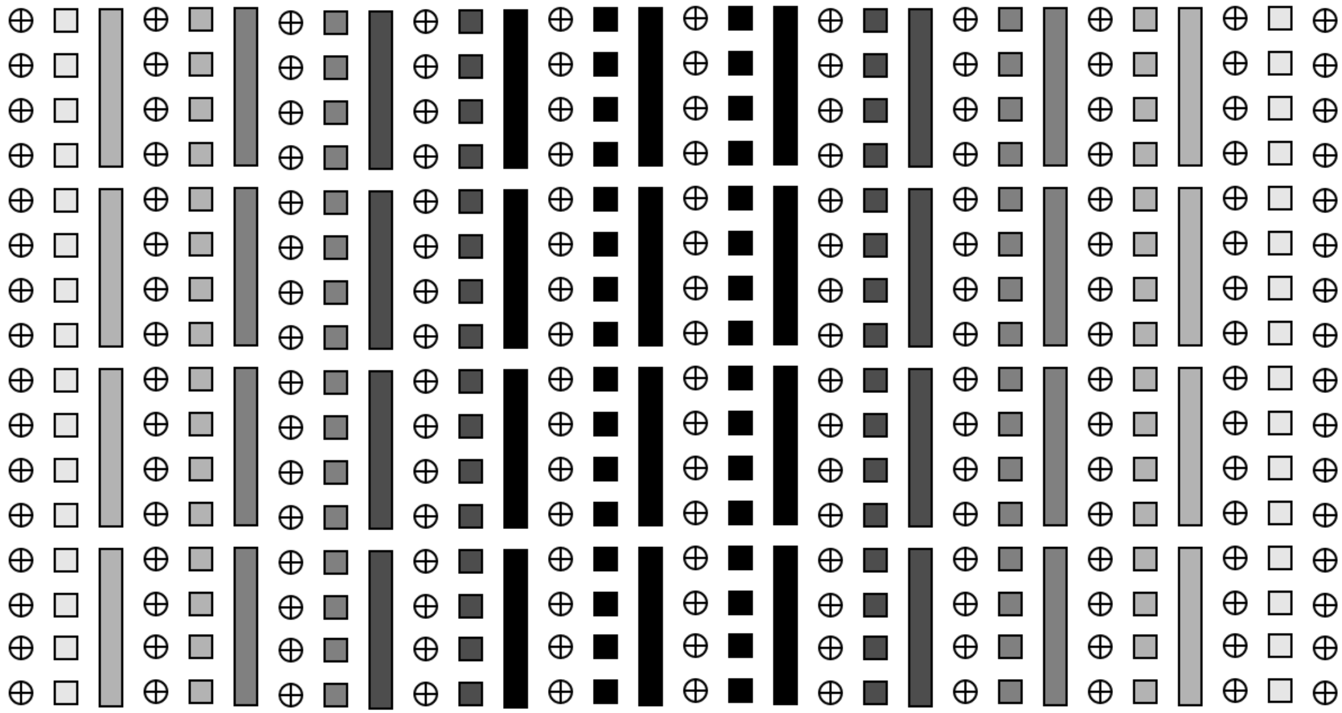
C. Archambeau
J. Balasch
G. Barthe
S. Belaïd
D. Bellizia
F. Berti
O. Bronchain
G. Cassiers
C. Dobraunig
A. Duc
F. Dupressoir
F. Durvaux
S. Duval
S. Dziembowski
S. Faust
P.-A. Fouque
B. Gierlichs
C. Glowacz
D. Goudarzi
B. Grégoire
V. Grosso
S. Guilley
T. Güneysu
Chun Guo
Qian Guo
G. Herold
A. Journault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

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

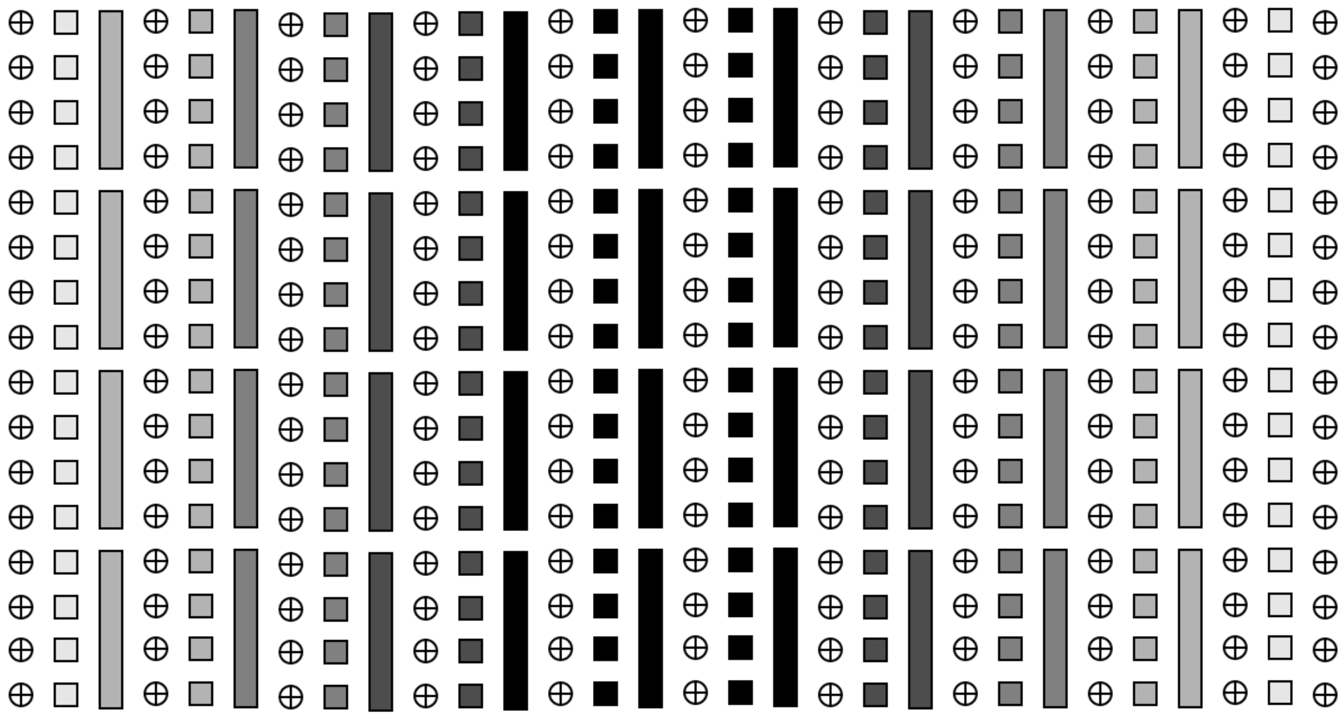
Outline

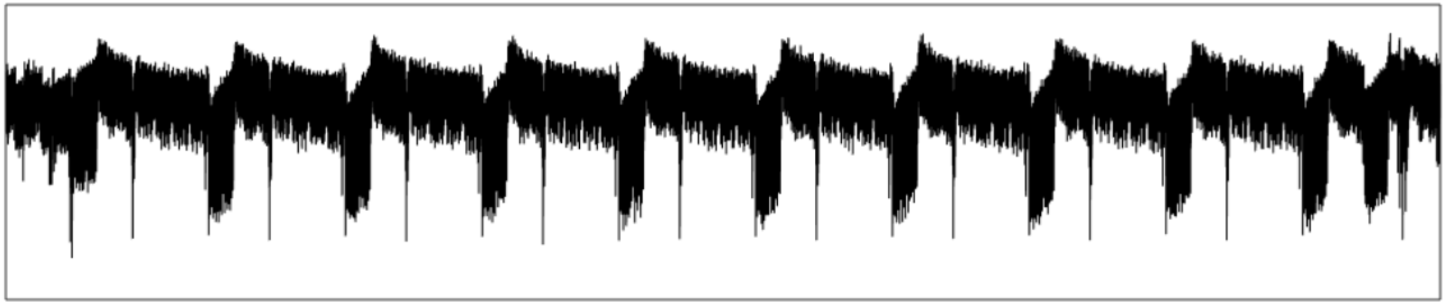
1. Side-channel **(crypt)analysis**: attacks taxonomy
2. Masking **countermeasure**: security vs. cost
3. Security **definitions** (authenticated encryption)
   a. Nonce-respecting setting (i.e., AEL)
   b. Nonce-misuse setting (i.e., AEmL)
4. Leakage-resistant AE **designs** (& implementations)
5. Conclusions (& the need of open evaluations)

AES Rijndael: $y = \mathrm{AES}_K(x)$



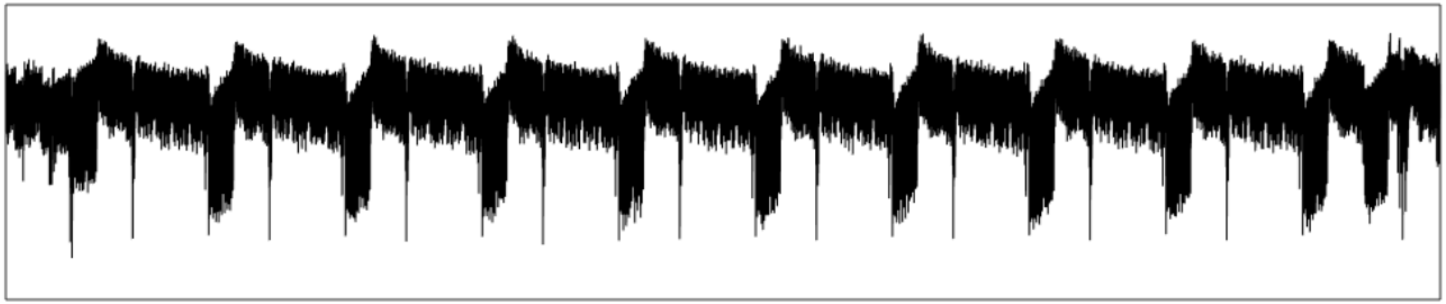
Leaking AES: $y = \mathrm{AES}_K(x) \rightarrow \boldsymbol{L}$





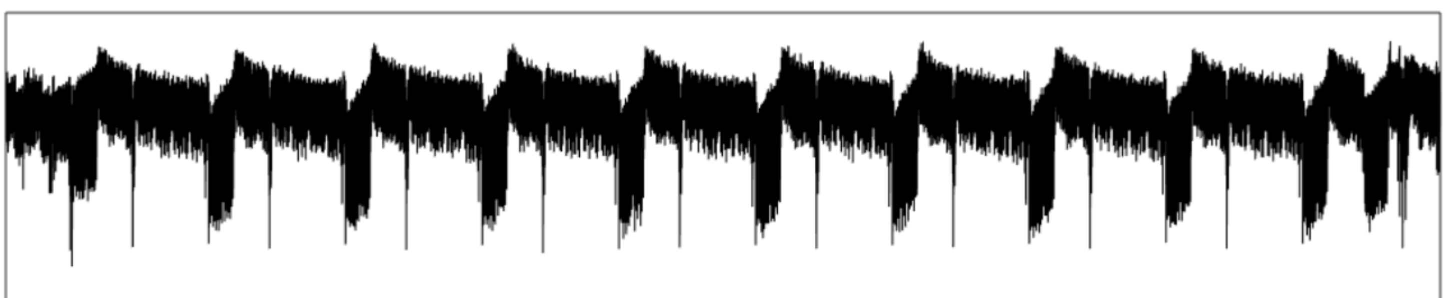
Leakage function definition

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



Leakage function definition

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

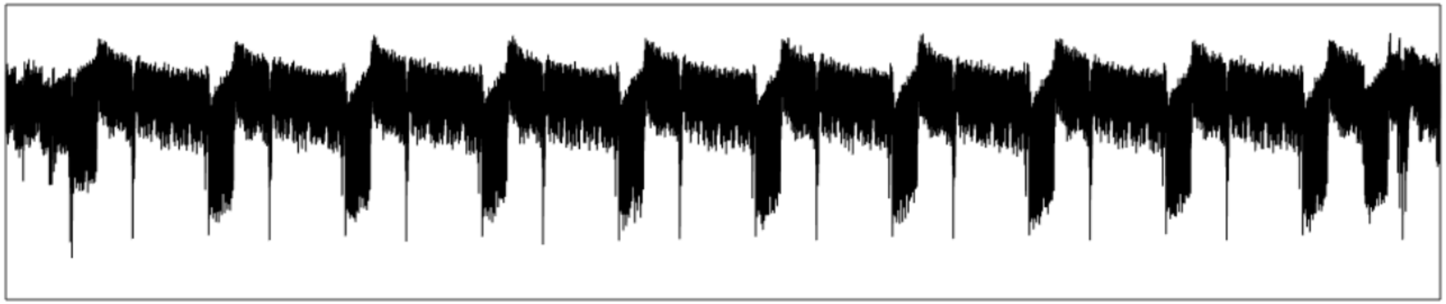


Leakage function definition

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

- Leakages are noisy: $\boldsymbol{L}(x, K) \approx \boldsymbol{\delta}(x, K) + \boldsymbol{N}$
  - Signal-to-Noise Ratio: $\mathrm{SNR}^i = \frac{\mathrm{var}(\delta_x^i)}{\mathrm{var}(N^i)}$

- The shape of $\boldsymbol{\delta}$ & $\boldsymbol{N}$ is technology-dependent
  - Their exact representation is unknown



Basic facts (I)

- Computing less means leaking less
  - E.g., unprotected **32-bit** implem. (**HW** leakages)

# rounds	# ops. / round	# samples / op.	MI (bits) / sample	λ (bits) / trace
10	100	5	$\dfrac{\mathbf{log(32) = 5}}{\left(1 + \frac{1}{\mathrm{SNR}}\right)}$	$\dfrac{\mathbf{25{,}000}}{\left(1 + \frac{1}{\mathrm{SNR}}\right)}$

Basic facts (I)

- Computing less means leaking less
  - E.g., unprotected **32-bit** implem. (**HW** leakages)

# rounds	# ops. / round	# samples / op.	MI (bits) / sample	λ (bits) / trace
10	100	5	$\frac{\textbf{log(32) = 5}}{\left(1+\frac{1}{\mathrm{SNR}}\right)}$	$\frac{\textbf{25,000}}{\left(1+\frac{1}{\mathrm{SNR}}\right)}$

  - Unprotected **128-bit** implem. (**HW** leakages)

# rounds	# ops. / round	# samples / op.	MI (bits) / sample	λ (bits) / trace
10	1	5	$\frac{\textbf{log(128) = 7}}{\left(1+\frac{1}{\mathrm{SNR}}\right)}$	$\frac{\textbf{350}}{\left(1+\frac{1}{\mathrm{SNR}}\right)}$

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* $\boldsymbol{L}(.,K)$:

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

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

Consequence (for theoretical analysis) 1.4

- 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* $\boldsymbol{L}(.,K)$:

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

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

⇒ 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 A needs to guess the state), leading to reduced « effective » λ's, e.g.,

exploited # rounds	# ops. / round	# samples / op.	MI (bits) / sample	eff. λ (bits) / trace & subkey
1	1	$\approx$ **2 (indep.)**	$\frac{\text{log(128) = 7}}{\mathbf{100}}$	$\frac{14}{100}$

- 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 » λ's, e.g.,

exploited # rounds	# ops. / round	# samples / op.	MI (bits) / sample	eff. λ (bits) / trace & subkey
1	1	$\approx$ **2 (indep.)**	$\frac{\text{log(128) = 7}}{\mathbf{100}}$	$\approx \left(\frac{14}{100}\right)^{\boldsymbol{d}}$

- With the masking countermeasures (see next)

Basic facts (II)

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

exploited # rounds	# ops. / round	# samples / op.	MI (bits) / sample	eff. λ (bits) / trace & subkey
1	1	$\approx$ **2 (indep.)**	$\frac{\log(128) = 7}{\mathbf{100}}$	$\approx \left(\frac{14}{100}\right)^{\boldsymbol{d}}$

- 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):

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

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):

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

- 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

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

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

- 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)

Summarizing (taxonomy of attacks)

- Key Recovery (KR) attacks (with known/chosen x_i's)

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

  - May require large amounts of leakage vectors to succeed

Summarizing (taxonomy of attacks) 1.7

- Key Recovery (KR) attacks (with known/chosen x_i's)

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

  - 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)

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

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

Summarizing (taxonomy of attacks)

- Key Recovery (KR) attacks (with known/chosen x_i's)

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

  - May require large amounts of leakage vectors to succeed
    - Or have bounded success probability in case of SPA

- Message Comparison (MC) attacks (with fresh challenge)

$$\Pr\left[A_{\text{MC}}^{\boldsymbol{L}(.,.)}\left(x_0, x_1, \boldsymbol{L}(x_b, K)\right) = b | K, b \leftarrow \$\right] \approx 2^{-128+\mathrm{D}\left(\boldsymbol{L}(x_0,K);\boldsymbol{L}(x_1,K)\right)}$$

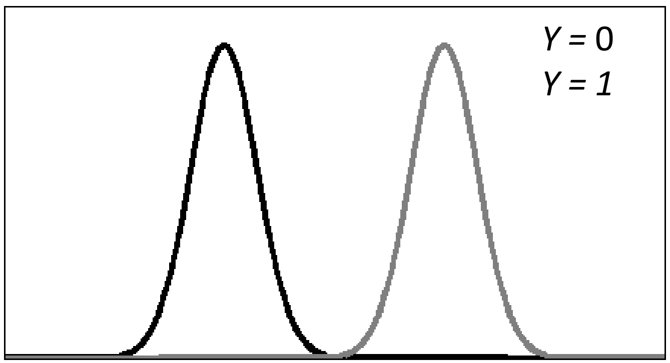
  - Significanly simpler than KR - but not trivial for all x_0, x_1 (!)
  - Depends on similarity of the message blocks' leakages

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

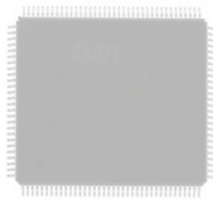
Outline

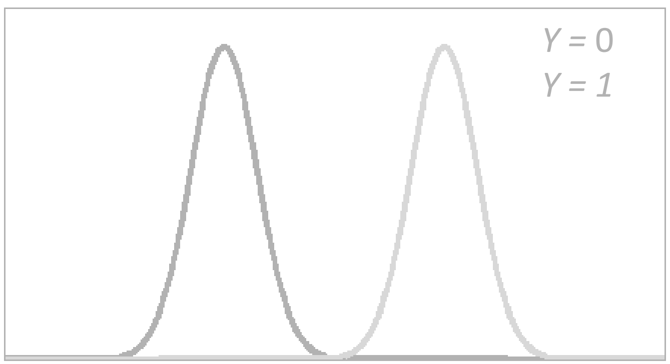
1. Side-channel **(crypt)analysis**: attacks taxonomy
2. Masking **countermeasure**: security vs. cost
3. Security **definitions** (authenticated encryption)
   a. Nonce-respecting setting (i.e., AEL)
   b. Nonce-misuse setting (i.e., AEmL)
4. Leakage-resistant AE **designs** (& implementations)
5. Conclusions (& the need of open evaluations)

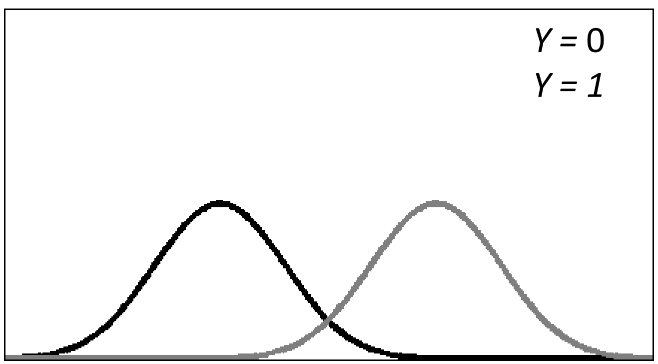
Noise (hardware) is not enough



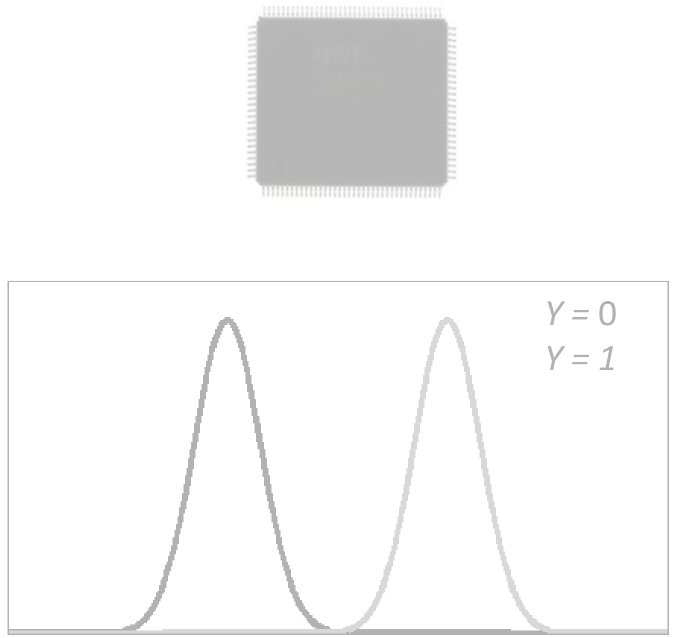
Noise (hardware) is not enough

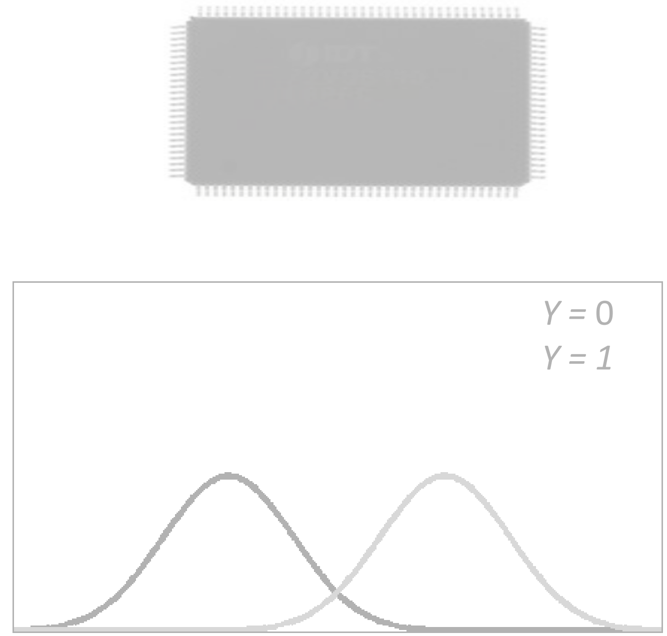






Noise (hardware) is not enough





- Additive noise ≈ cost × 2 ⇒ security × 2
  ⇒ not a good (crypto) security parameter
- ≈ same holds for all hardware countermeasures

Masking (≈ noise amplification)

- Example: Boolean encoding

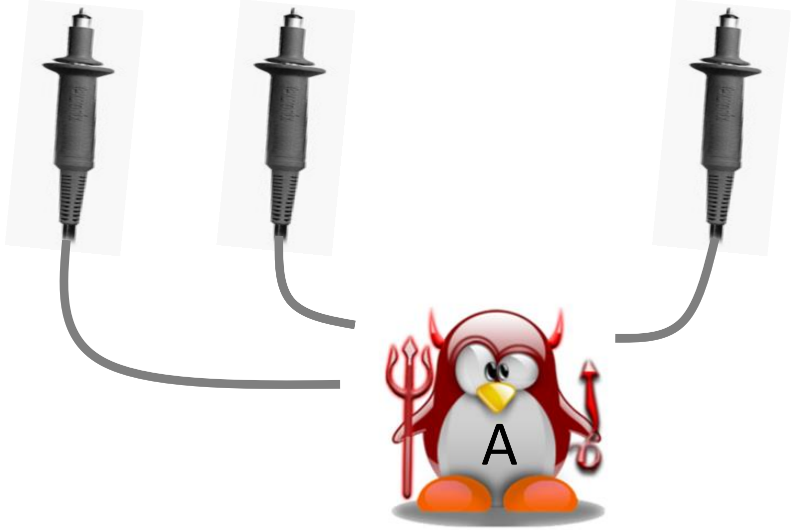
$$y = y_1 \oplus y_2 \oplus \cdots \oplus y_{d-1} \oplus y_d$$

- With $y_1, y_2, \dots, y_{d-2}, y_{d-1} \leftarrow \{0,1\}^n$

Masking (abstract view)

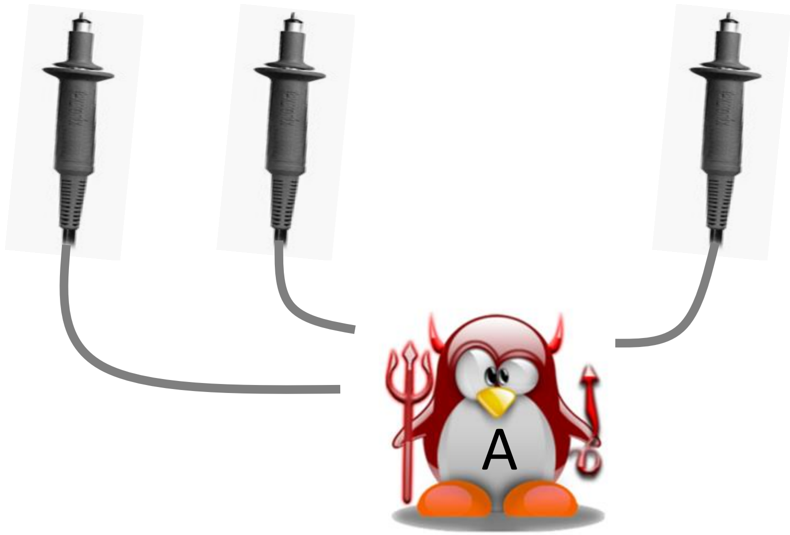
- Private circuits / probing security [ISW03]

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



- Private circuits / probing security [ISW03]

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

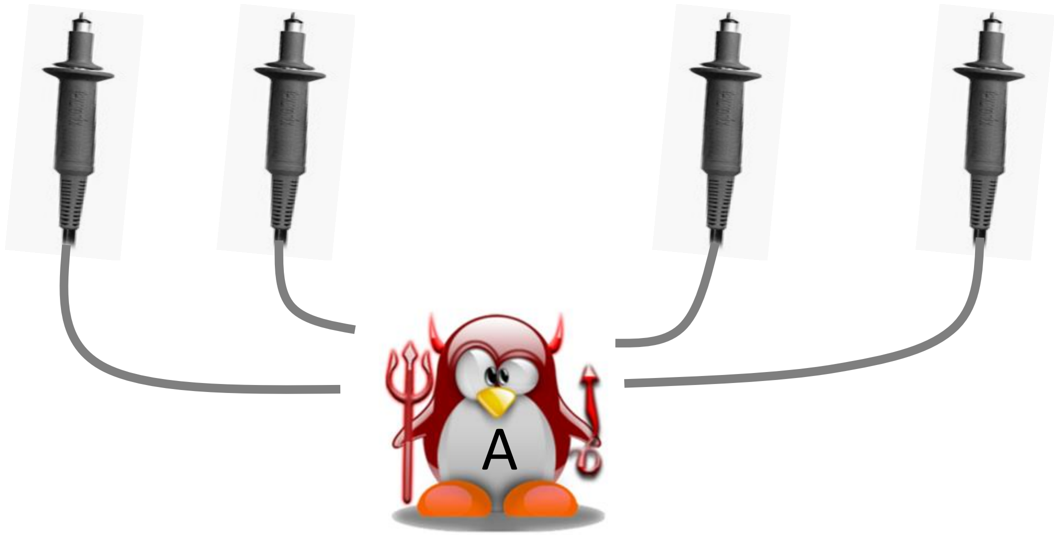


- $d - 1$ probes do not reveal anything on y

Masking (abstract view) 2.2

- Private circuits / probing security [ISW03]

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

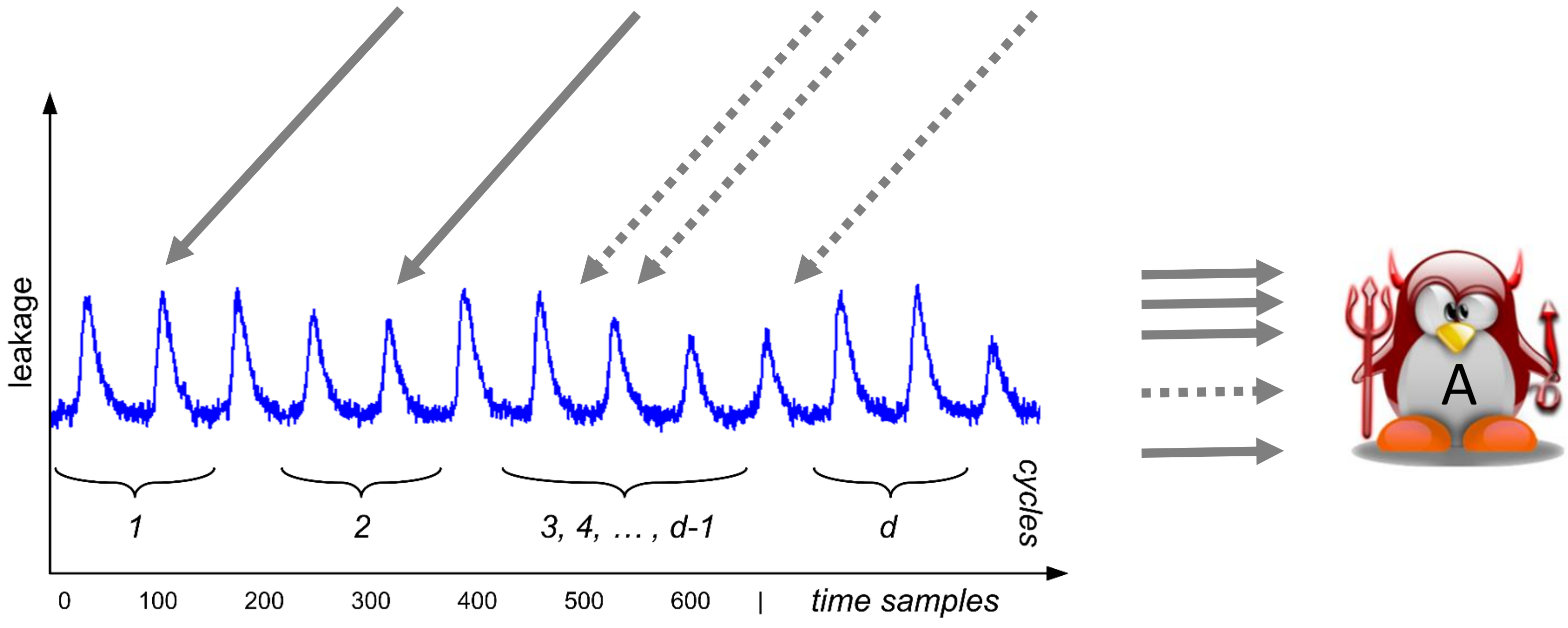


- But d probes completely reveal y

Masking (concrete view) 2.2

- Private circuits / probing security [ISW03]

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



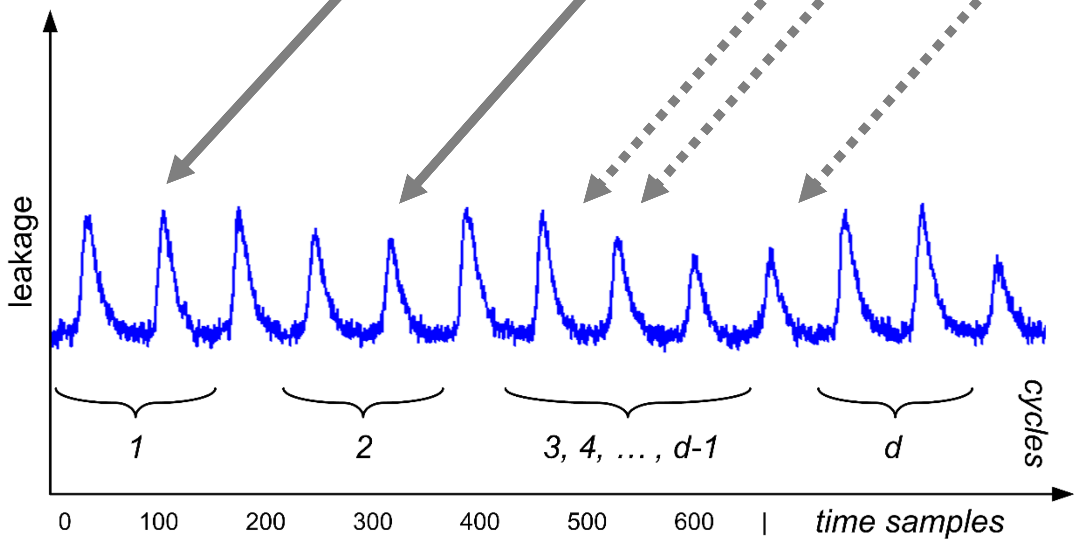
serial implementation.

- Noisy leakage security [PR13]

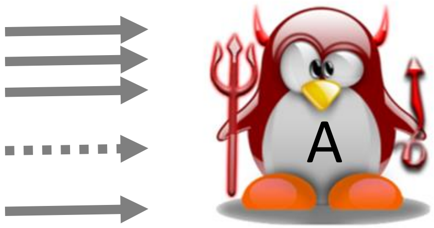
Masking (concrete view)

- Private circuits / probing security [ISW03]

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



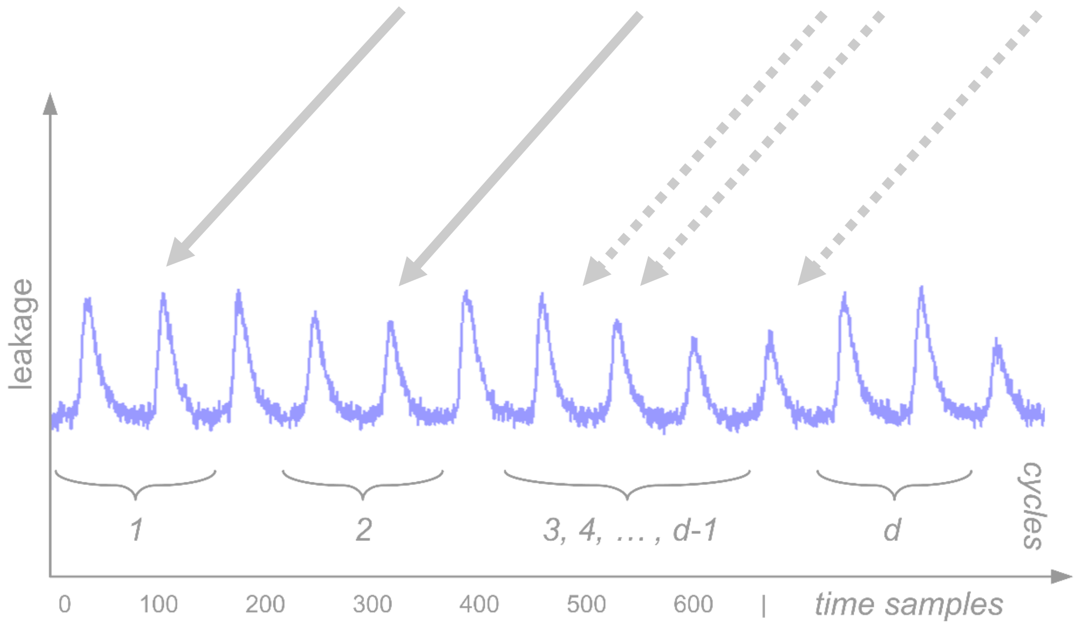
serial implementation.



- Bounded information $\mathrm{MI}(Y; \boldsymbol{L}) < \mathrm{MI}(Y_i; \boldsymbol{L}_{Y_i})^d$

- Private circuits / probing security [ISW03]

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



serial implementation.

noise + independence

[DDF14]

- Bounded information $\mathrm{MI}(Y;\boldsymbol{L}) < \mathrm{MI}(Y_i;\boldsymbol{L}_{Y_i})^d$

Masked operations [ISW03]

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

Masked operations [ISW03]

- Linear operations: $\mathrm{f(a)} = \mathrm{f}(a_1) \oplus \mathrm{f}(a_2) \oplus \cdots \oplus \mathrm{f}(a_d)$
- Multiplications: $\mathrm{c} = a \times b$ in three steps

Masked operations [ISW03]

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

- Multiplications: $\mathrm{c} = a \times b$ 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: $\mathrm{f(a)} = \mathrm{f}(a_1) \oplus \mathrm{f}(a_2) \oplus \cdots \oplus \mathrm{f}(a_d)$

- Multiplications: $\mathrm{c} = a \times b$ in three steps

$$\underbrace{\begin{bmatrix} a_1 b_1 & a_1 b_2 & a_1 b_3 \\ a_2 b_1 & a_2 b_2 & a_2 b_3 \\ a_3 b_1 & a_3 b_2 & a_3 b_3 \end{bmatrix}}_{\text{partial products}} + \underbrace{\begin{bmatrix} 0 & r_1 & r_2 \\ -r_1 & 0 & r_3 \\ -r_2 & -r_3 & 0 \end{bmatrix}}_{\text{refreshing}}$$

Masked operations [ISW03]

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

- Multiplications: $\mathrm{c} = a \times b$ 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: $\mathrm{f(a)} = \mathrm{f}(a_1) \oplus \mathrm{f}(a_2) \oplus \cdots \oplus \mathrm{f}(a_d)$
- Multiplications: $\mathrm{c} = a \times b$ 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

$a_1 b_1 \oplus a_1 b_2 \oplus a_1 b_3 = \boldsymbol{a_1 b}$ leaks on b

Masked operations [ISW03]

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

- Multiplications: $\mathrm{c} = a \times b$ in three steps

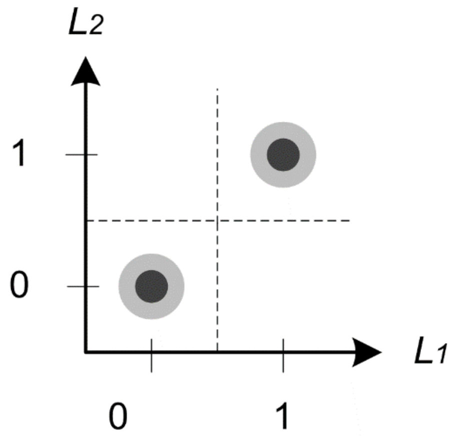
$$\underbrace{\begin{bmatrix} a_1b_1 & a_1b_2 & a_1b_3 \\ a_2b_1 & a_2b_2 & a_2b_3 \\ a_3b_1 & a_3b_2 & a_3b_3 \end{bmatrix}}_{\text{partial products}} + \underbrace{\begin{bmatrix} 0 & r_1 & r_2 \\ -r_1 & 0 & r_3 \\ -r_2 & -r_3 & 0 \end{bmatrix}}_{\textbf{refreshing}} \Rightarrow \underbrace{\begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix}}_{\text{compression}}$$

$a_1b_1 \oplus a_1b_2 \oplus a_1b_3 = \boldsymbol{a_1 b}$ leaks on b

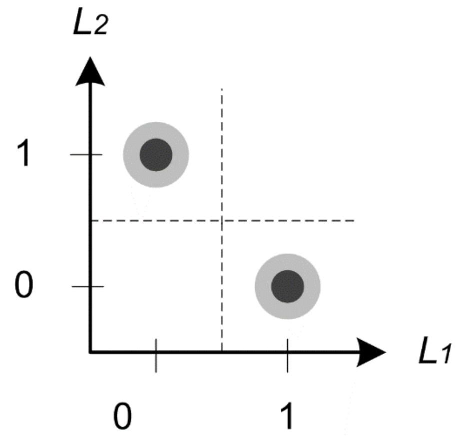
⇒ Quadratic overheads & randomness

- (Many published optimizations [R+15,Be+16,GM18])

Statistical intuition (2 shares)

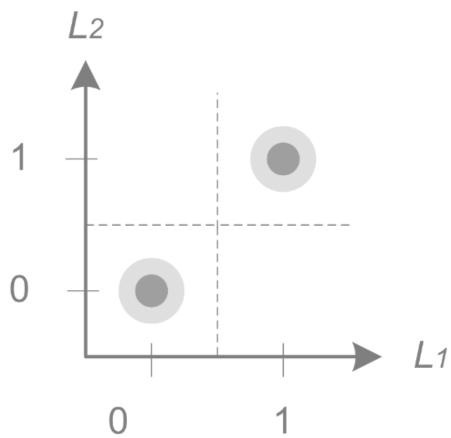


(a) $Y = 0$, serial.

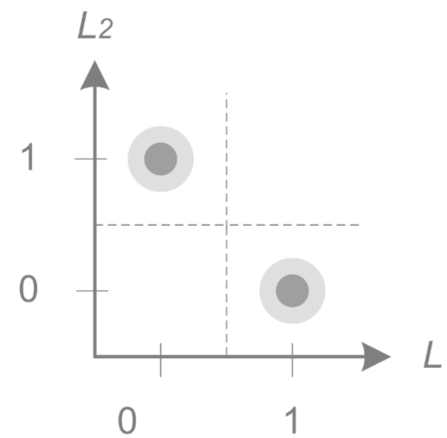


(b) $Y = 1$, serial.

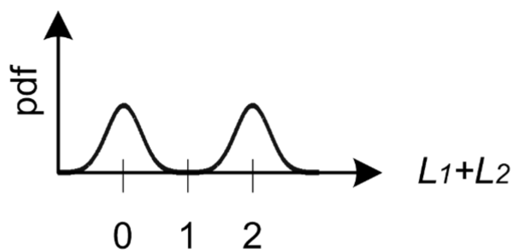
- Leakage mean vector for $Y = 0{,}1 = [0.5\ 0.5]$



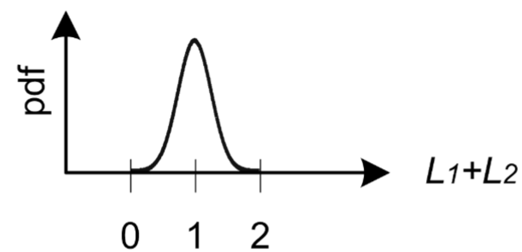
(a) $Y = 0$, serial.



(b) $Y = 1$, serial.



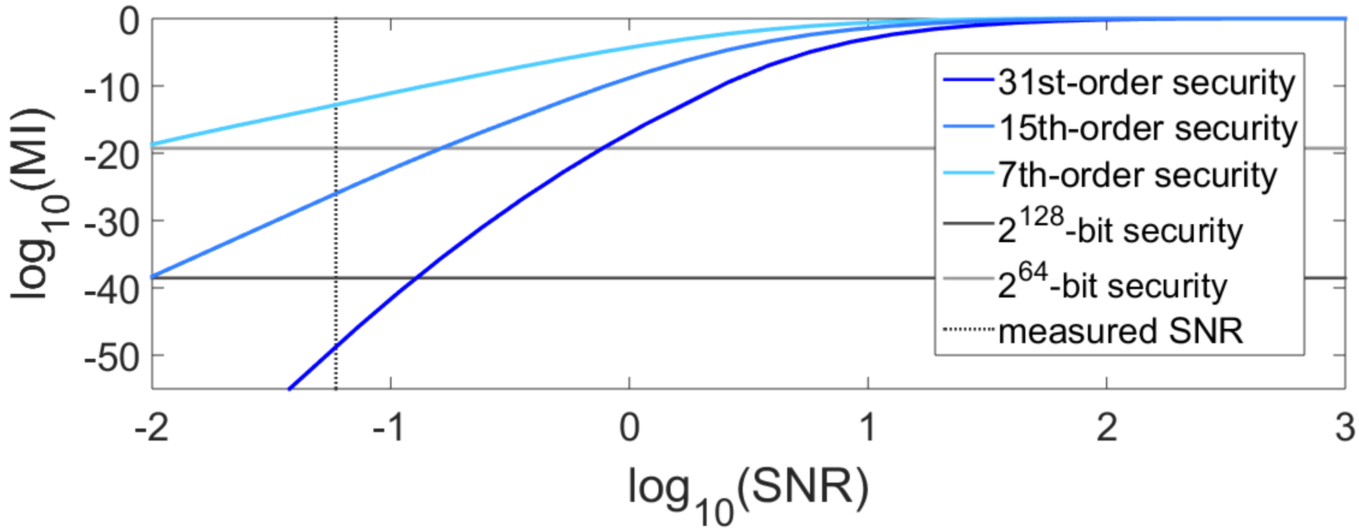
(c) $Y = 0$, parallel.



(d) $Y = 1$, parallel.

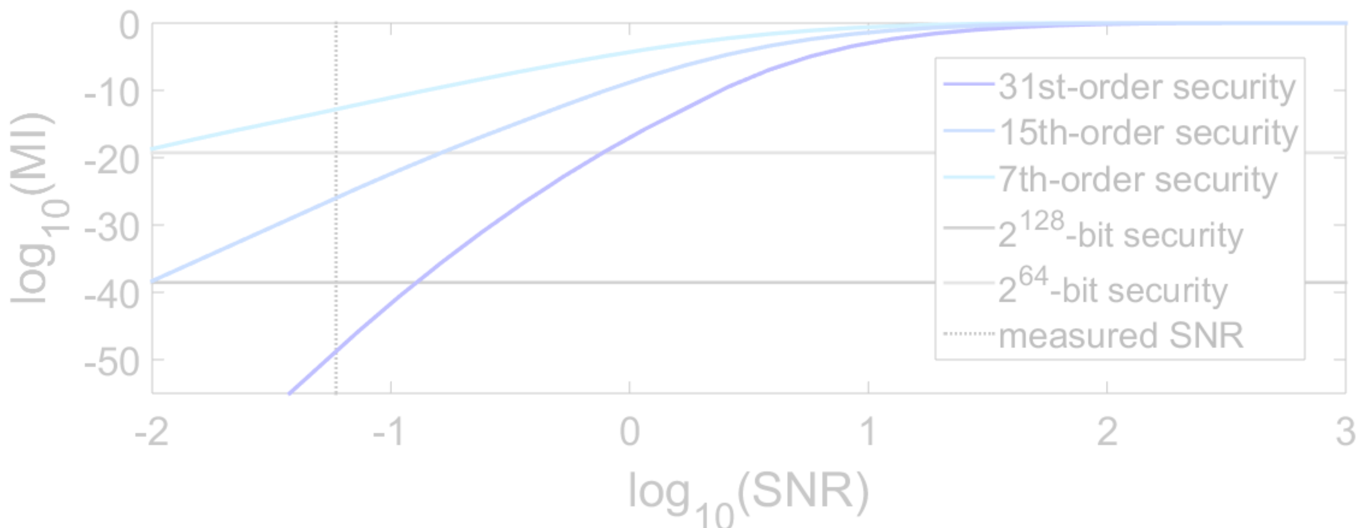
- Leakage mean value for $Y = 0{,}1 = 1$

Case study: ARM Cortex M4 [JS17]

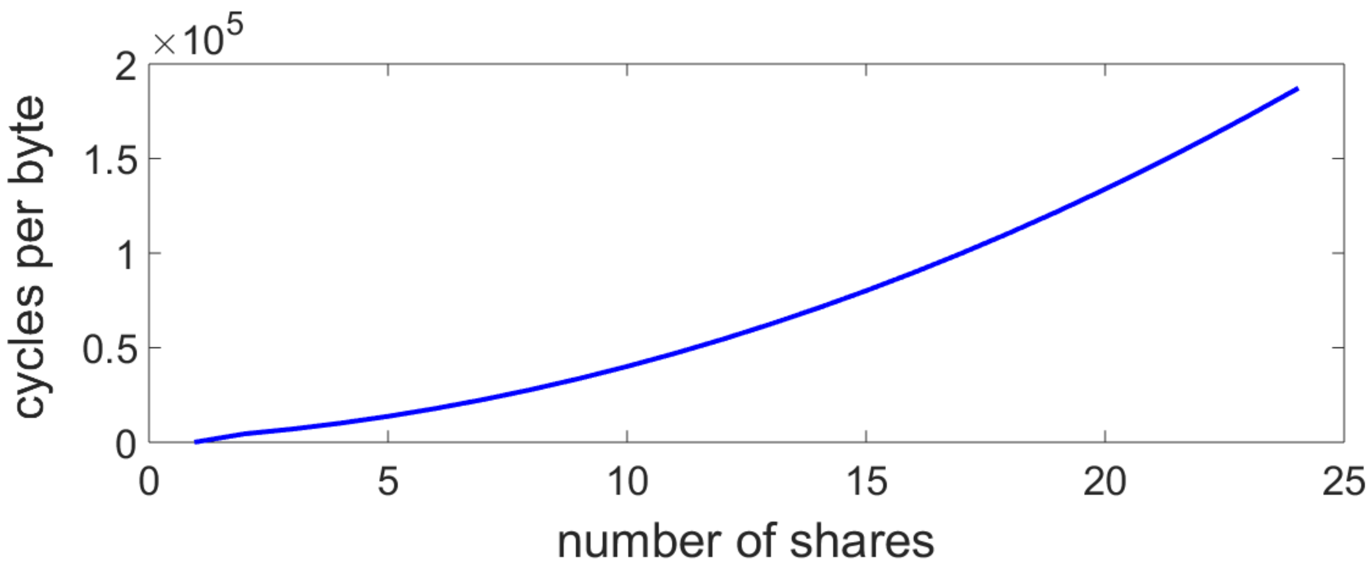


security

Case study: ARM Cortex M4 [JS17]

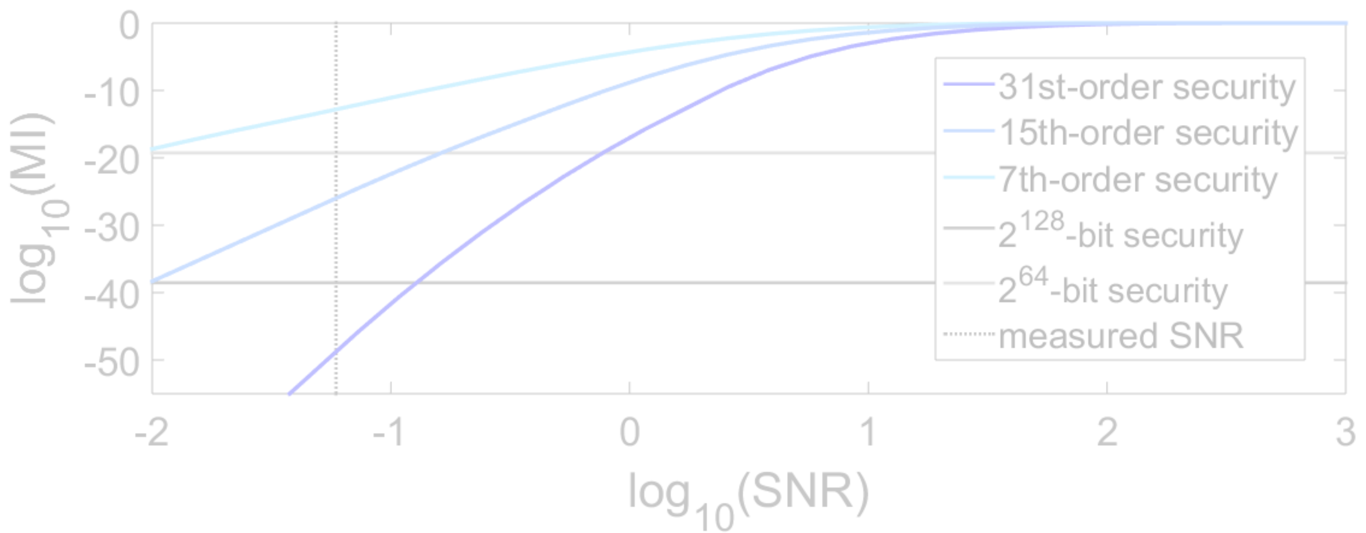


security

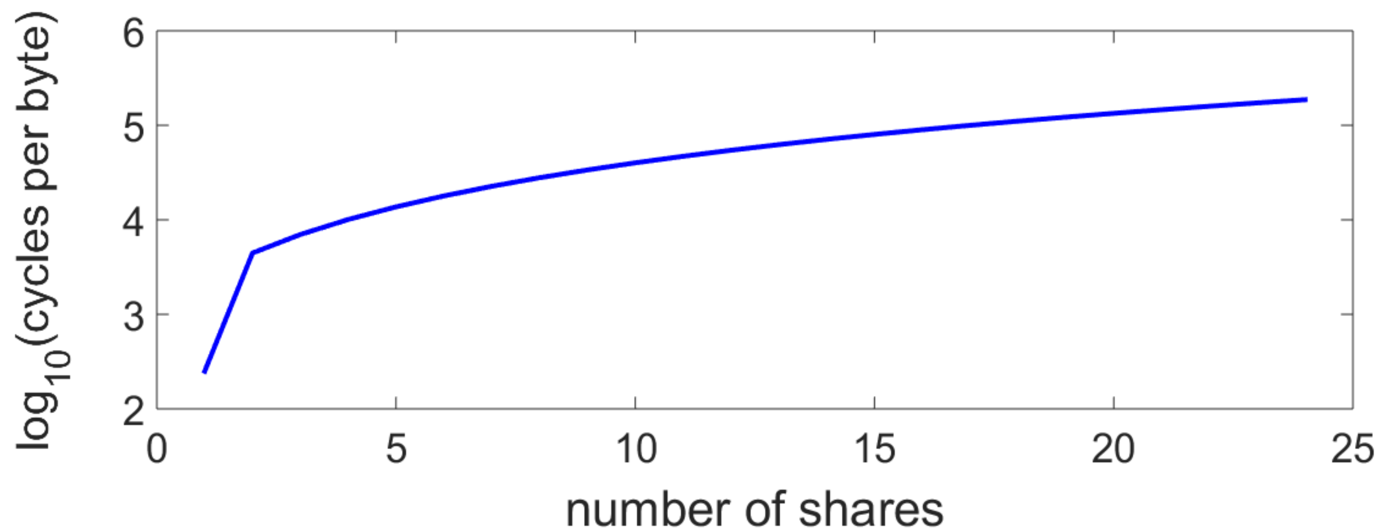


performance

Case study: ARM Cortex M4 [JS17]



security



performance

Summarizing

- Sounds easy but implementation is complex

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]

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)

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)

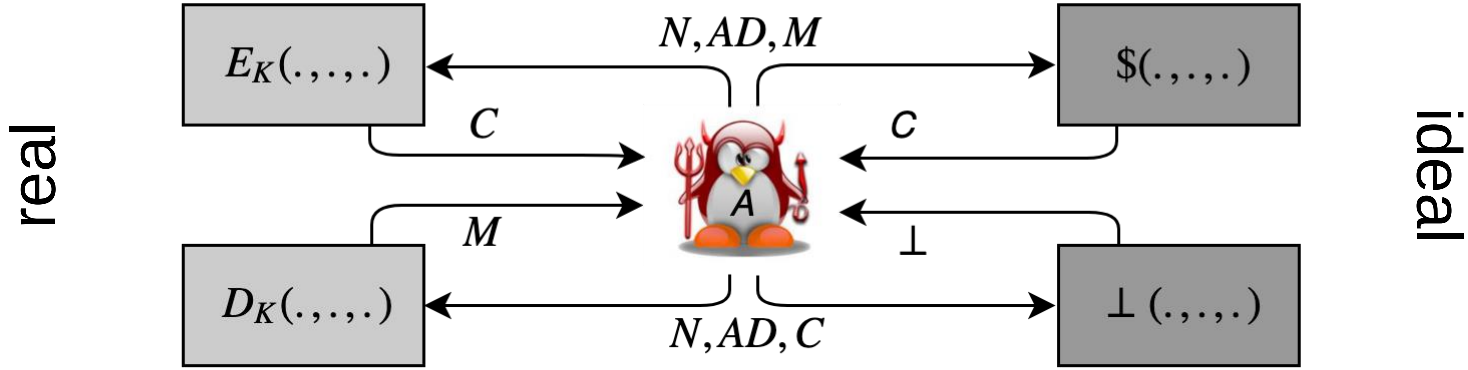
- SPA security expected to be (much) cheaper

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

Authenticated Encryption (AEAD)

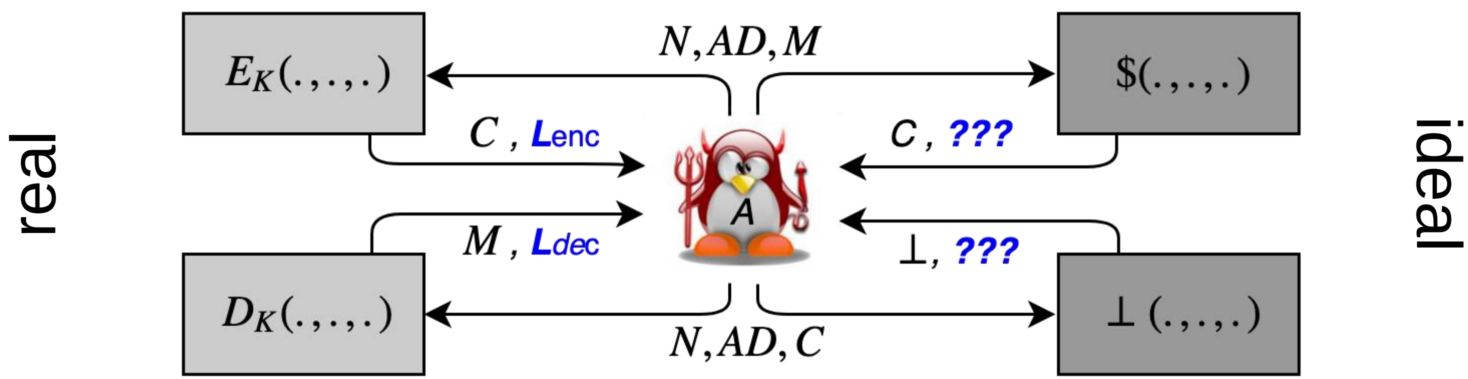
- Why not extending [RS06]'s *all in one* definition?



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

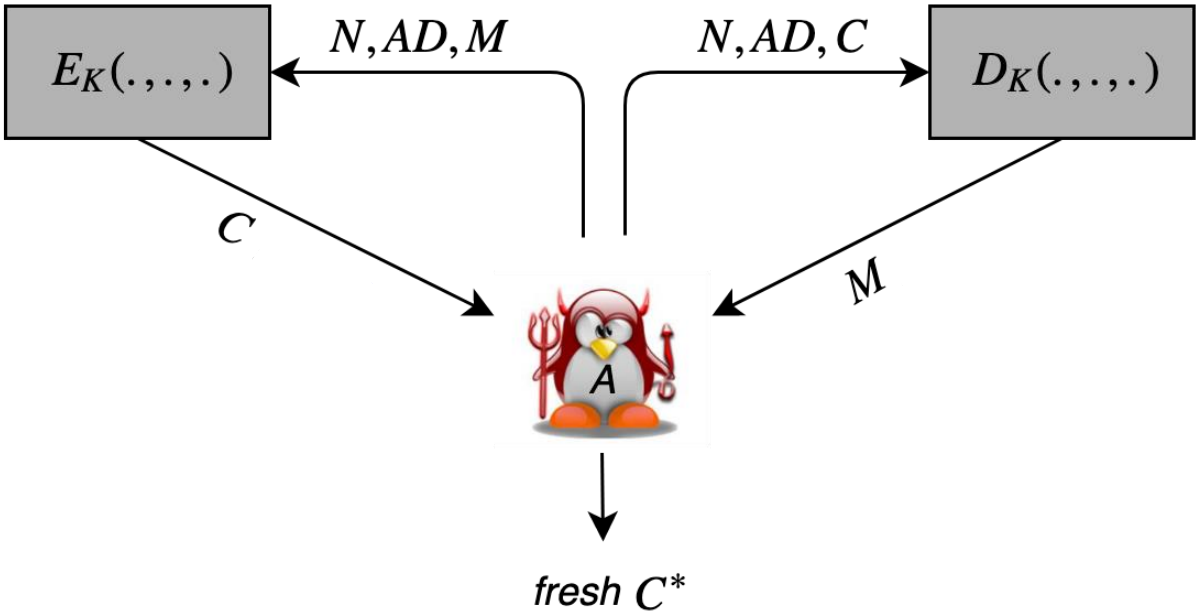
Authenticated Encryption (AEAD)

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

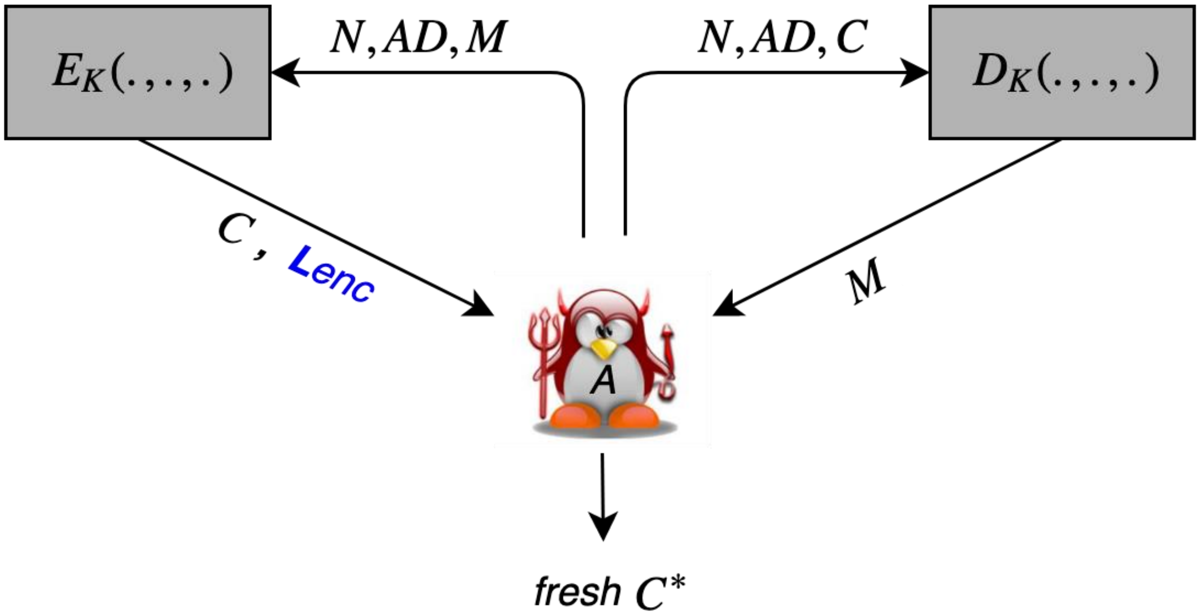


- A cannot ask a decryption query on (N, AD, C) after C is returned by an $(N, AD, .)$ encryption query
- **Problem: the leakage of ideal objects** (which do not have implementations) **seems difficult to define**

Ciphertext Integrity

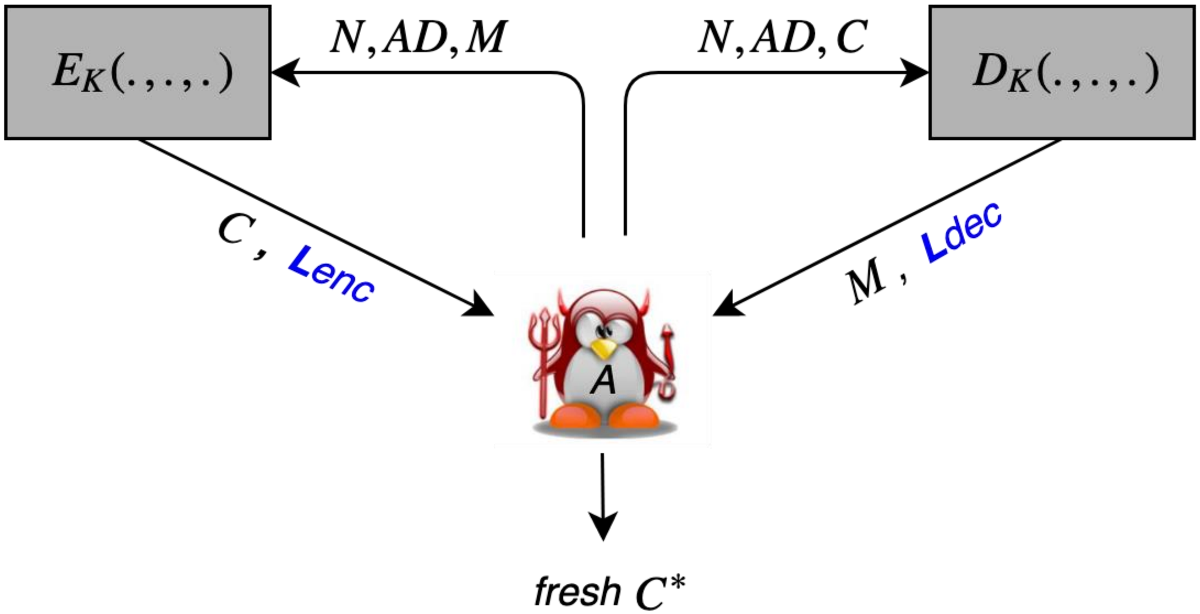


Ciphertext Integrity with Leakage



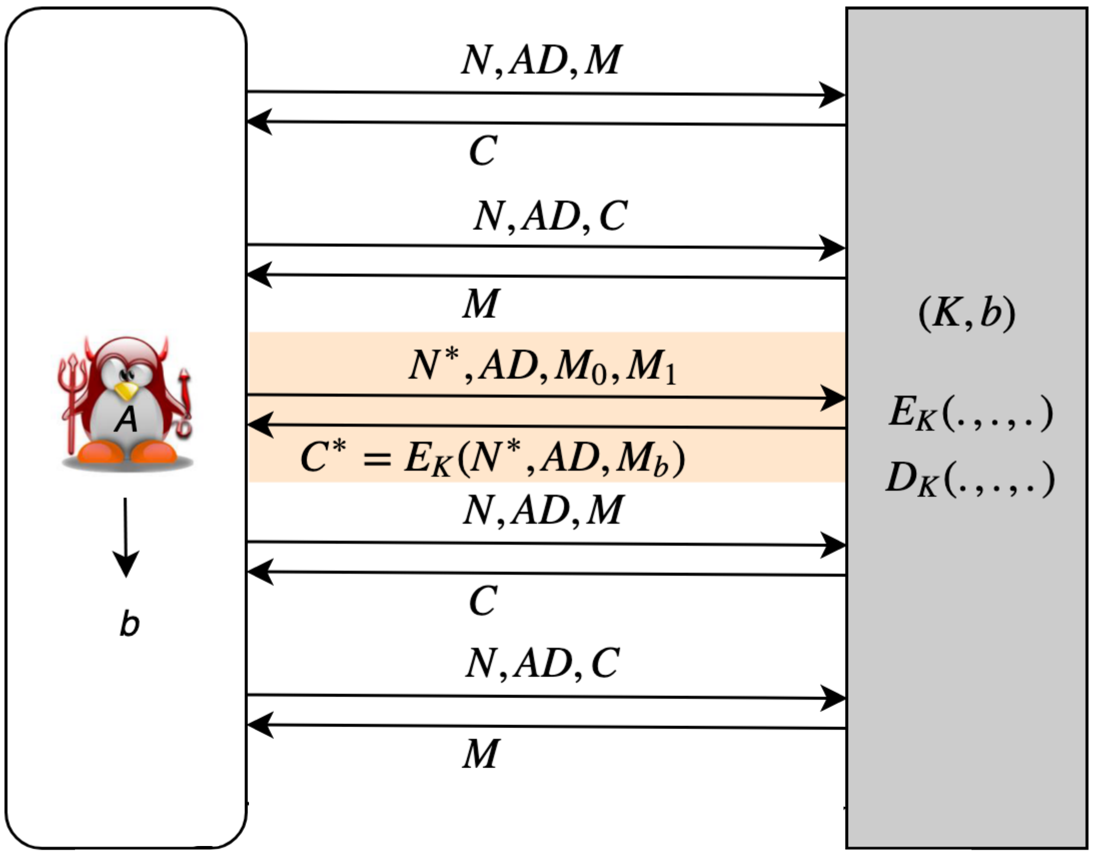
- CIL1: leagage in encryption only [Be+18]

Ciphertext Integrity with Leakage 3.2

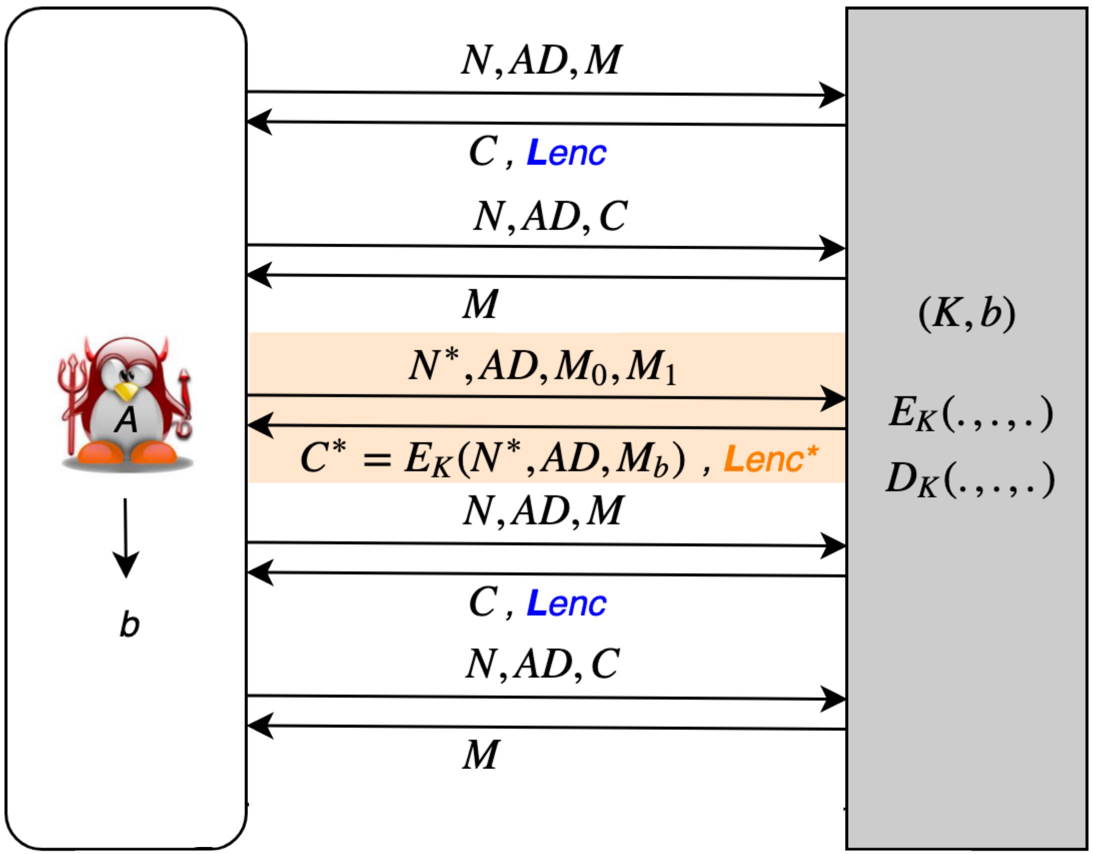


- CIL2: leagage in encryption and decryption [BPPS17]
- Natural extensions (no definitional challenges) with many applications (e.g., secure bootloading)

Chosen Ciphertext Security

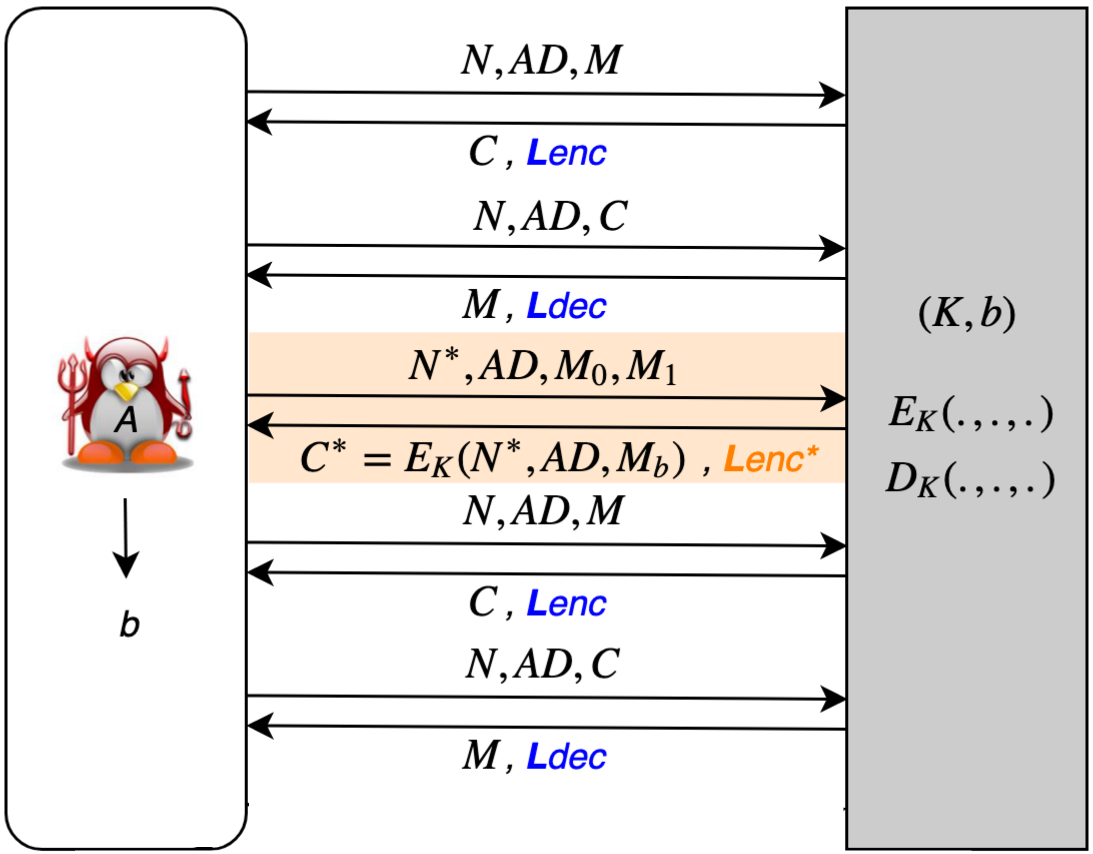


CCA Security with Leakage [GPPS18]



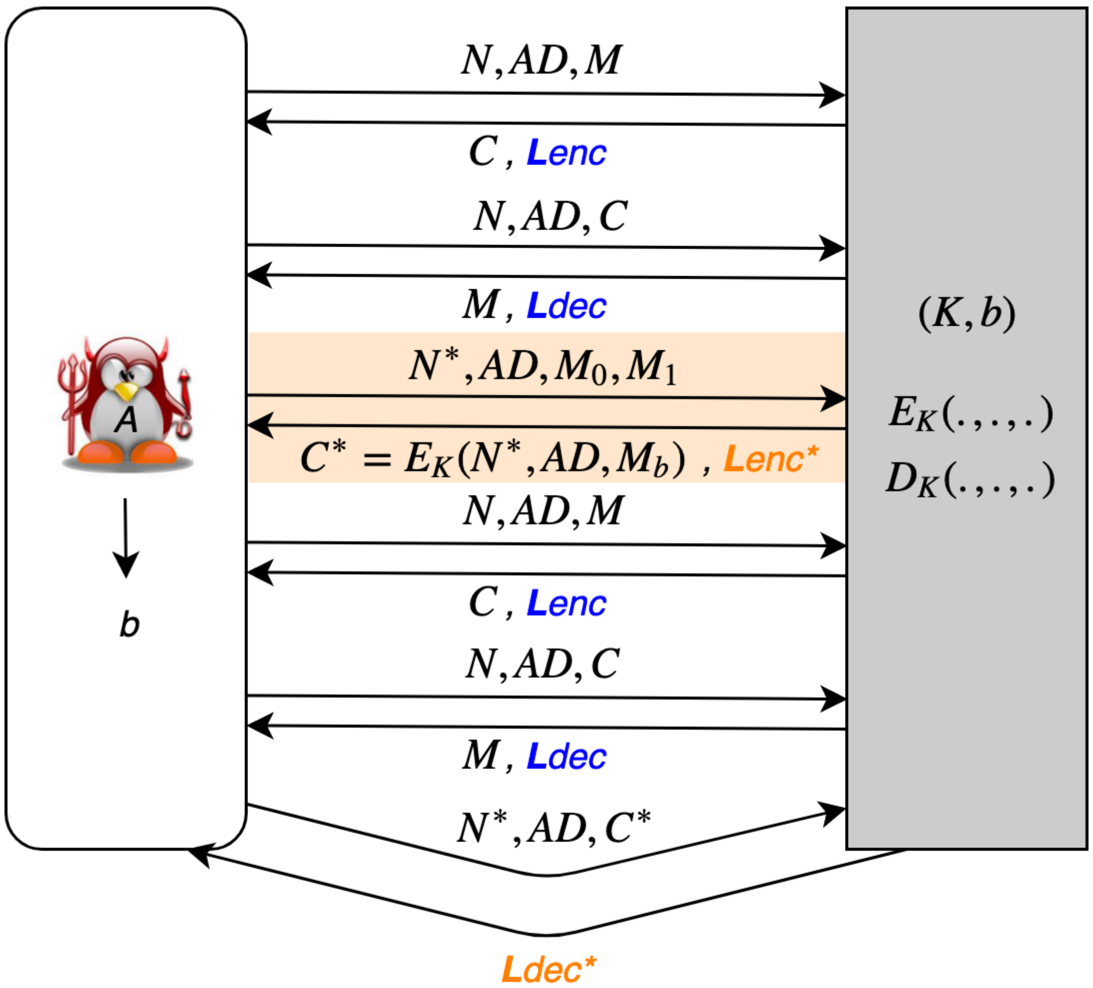
- CCAL1: leakage in encryption

CCA Security with Leakage [GPPS18]



- CCAL2: leakage in encryption and decryption

CCA Security with Leakage [GPPS18]



- + challenge *Ldec** (applications: IP protection, …)

- [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*

The challenge leakage controversy (II) 3.5

- 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)

The challenge leakage controversy (II)

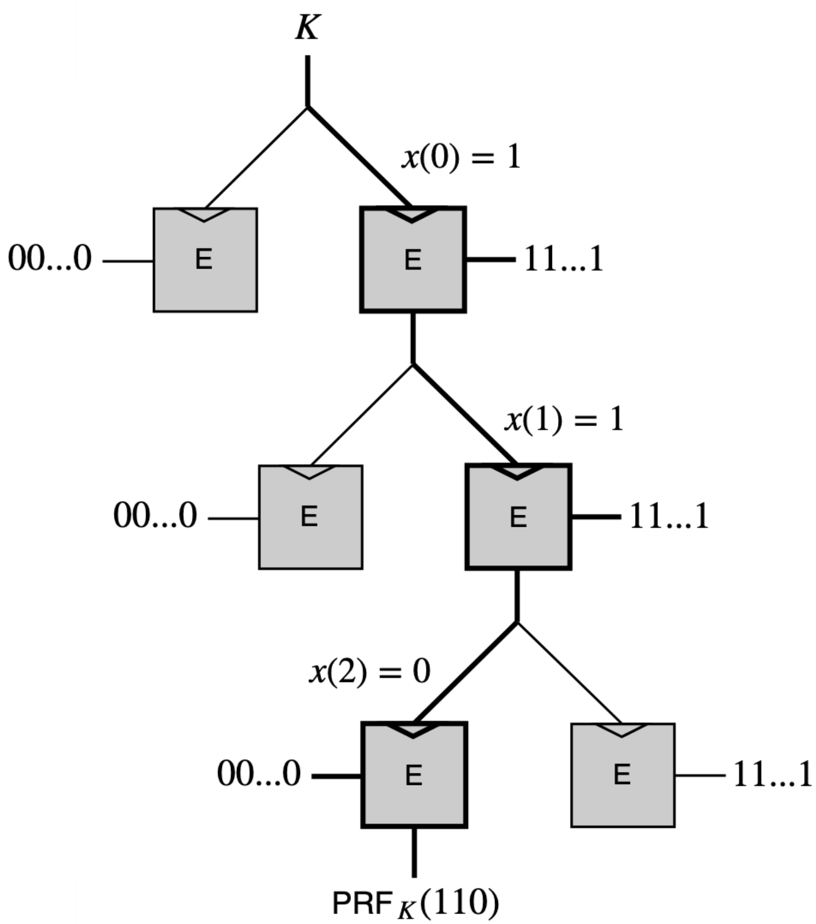
- If we do not make it part of the definition it will never be a goal for cryptographers & engineers
  - Cryptographers: minimize the message manipulation
  - Engineers: minimize message leakage, e.g., with special encodings (which is not much studied yet)
- We need to understand what can be achieved
  - *Even if results are not ideal* (e.g., no negl. Adv.)

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 tivial, e.g.,
    - Remote timing attacks: scalar leakages (vs. vectors)
    - Proxy re-encryption: messages are not chosen

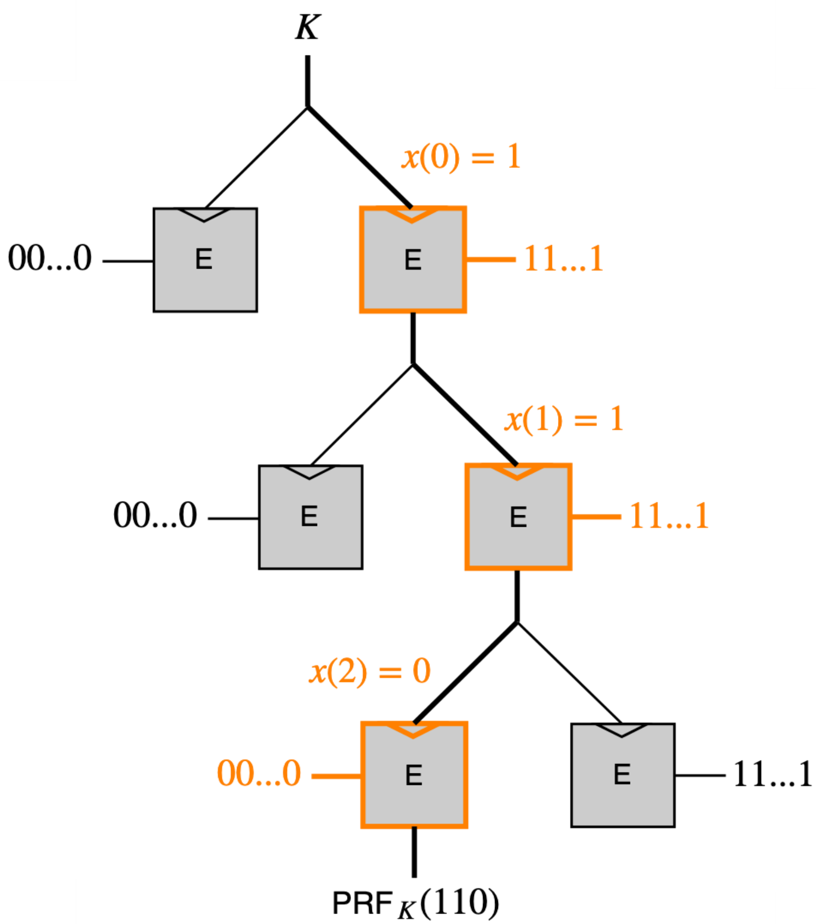
An motivating example

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



An motivating example

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



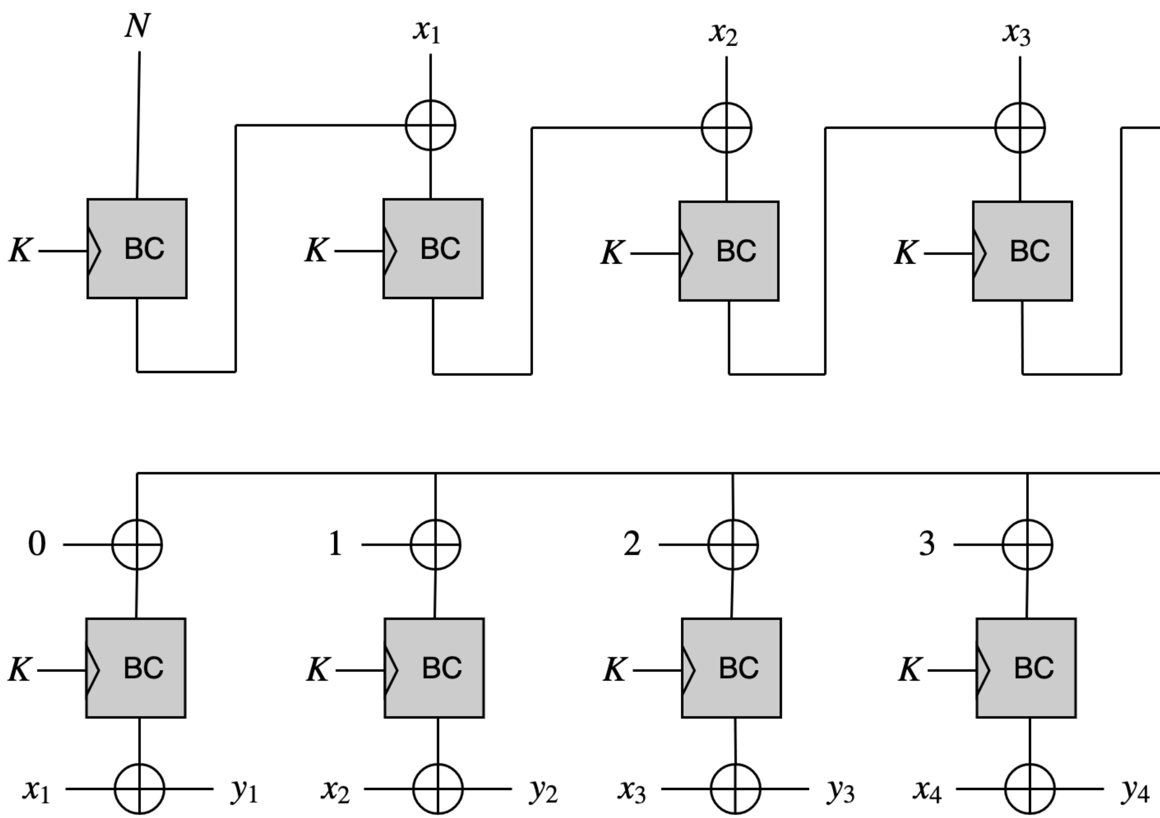
- Leads to simple MC attacks
  - Message encrypted bit per bit ⇒ no algorithmic noise
  - Constant block cipher inputs « all zeros » and « all ones » easy to distinguish with HWs [B12]

- (Yet is quite good against KR)

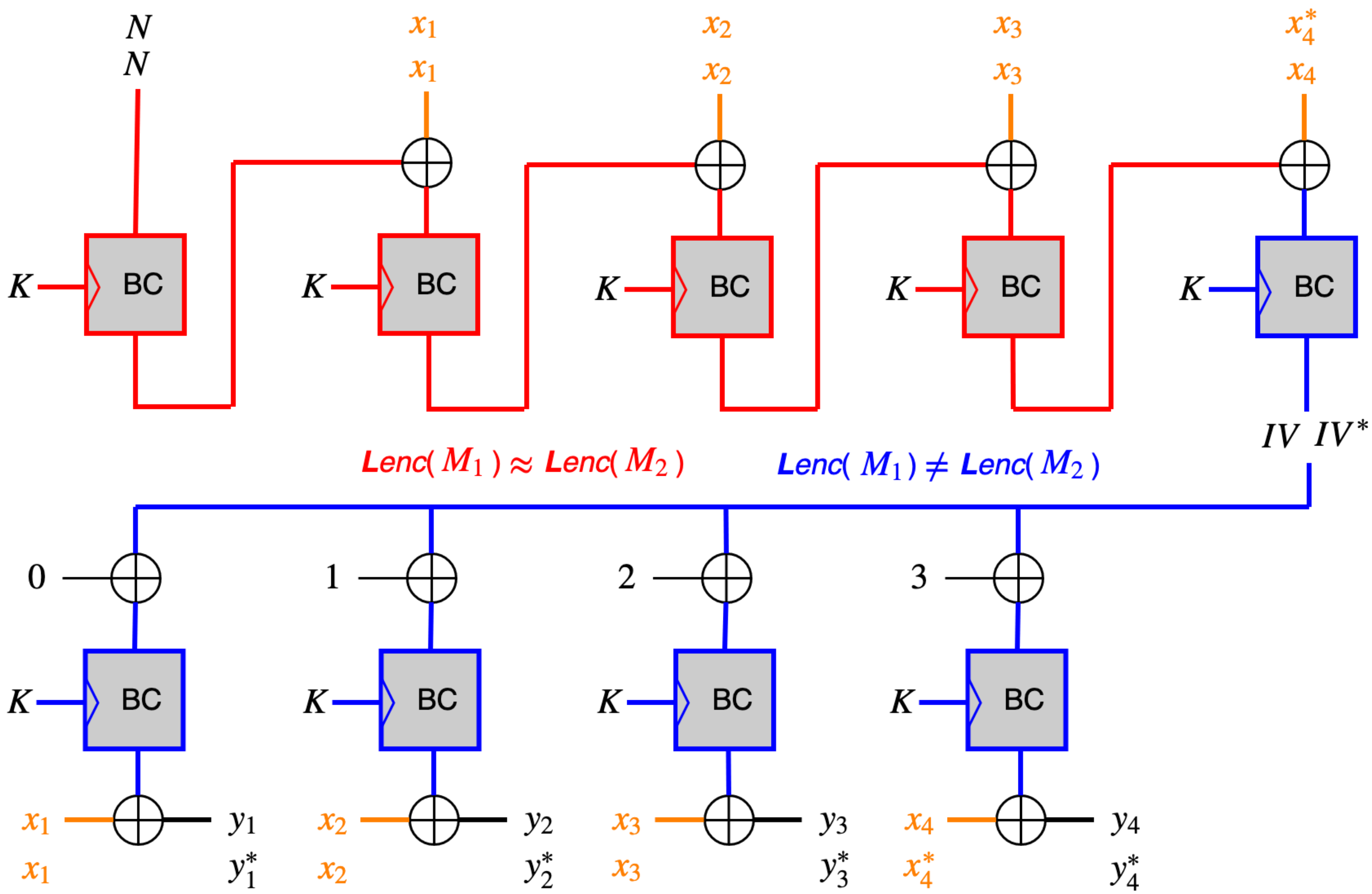
Outline

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)

Misuse-Resistance (MR) [RS06]

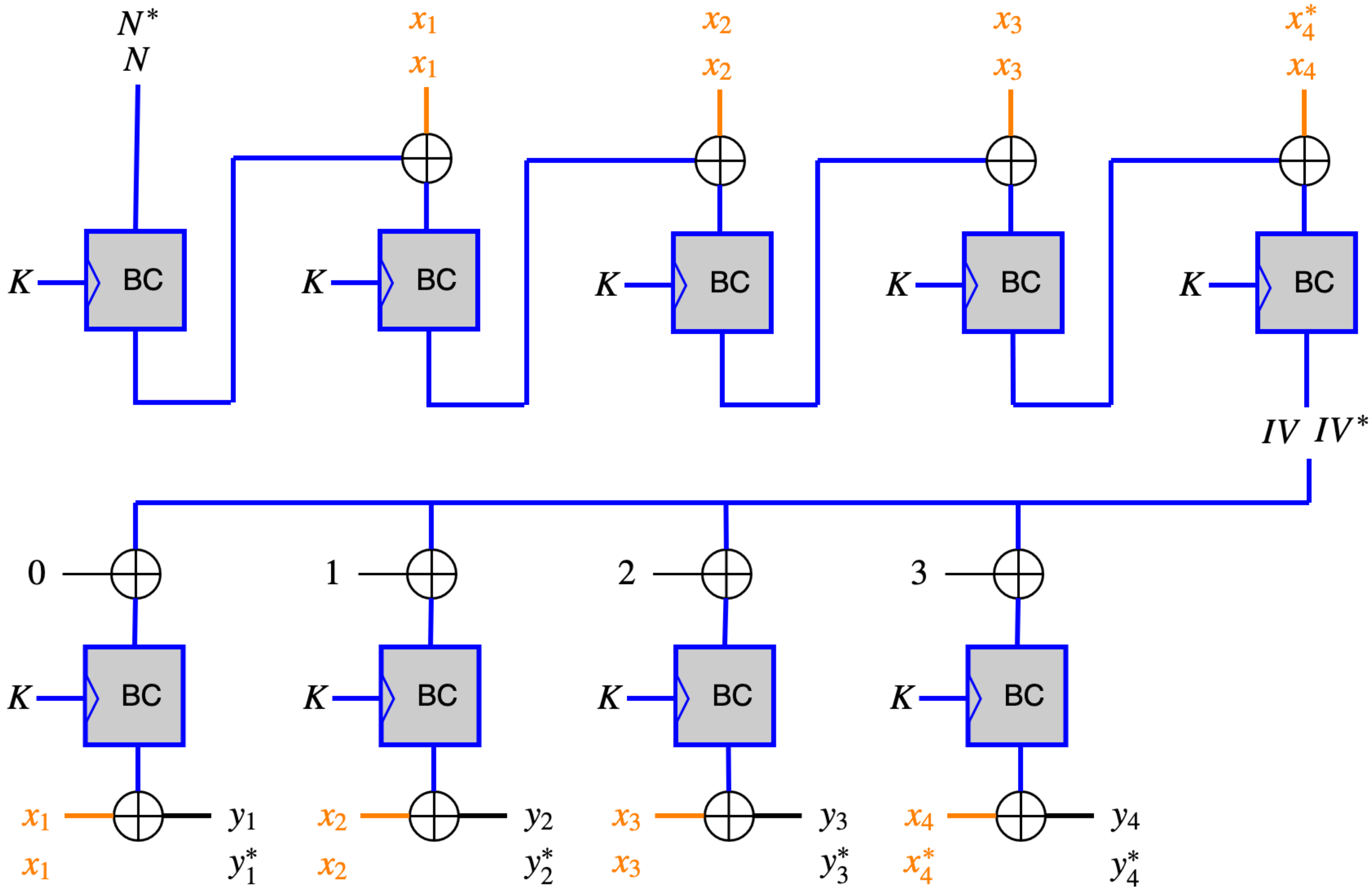


- Black box: only identical (N, M) pairs should be at risk
- Typically achieved by having a 2-pass mode (e.g., SIV)



- 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

Misuse-Resilience (mR) [ADL17]



- Fresh challenge nonce circumvent this impossibility
  - Intuition: leaves mostly MC attacks and DPAs

Summarizing

- For confidentiality, no meaningful encryption scheme can ensure leakage-resistance and (nonce) misuse-resistance (excluding trivial / fully leak-free solutions)

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]

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

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]
- $\approx$ 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

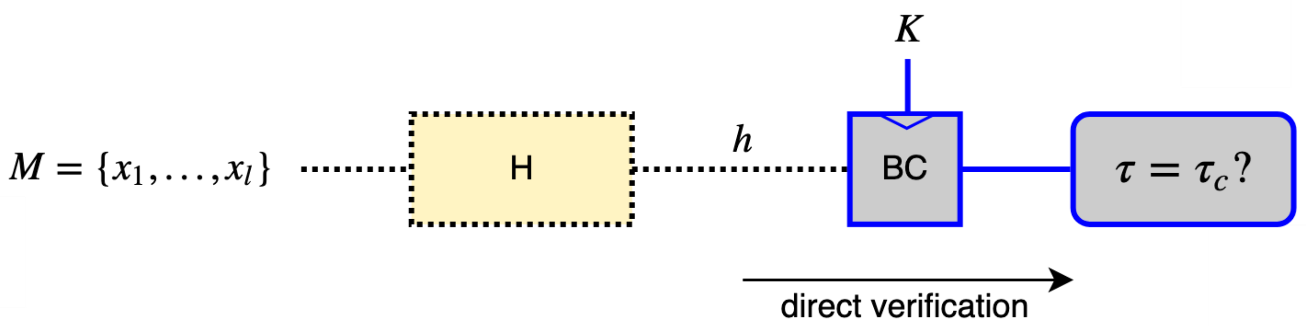
Summarizing

- 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 (*leak.-resist*.) def.: AEML=CIML2+CCAmL2+MR
- Weaker variants can be meaningful: for instance AEmL=CIML2+CCAmL2 [Be+19], CPAl1 [DM19], …

Outline

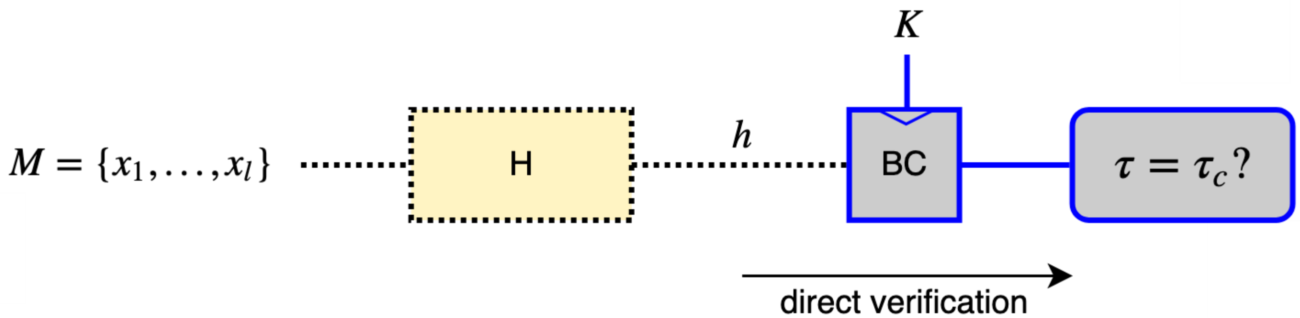
1. Side-channel **(crypt)analysis**: attacks taxonomy
2. Masking **countermeasure**: security vs. cost
3. Security **definitions** (authenticated encryption)
   a. Nonce-respecting setting (i.e., AEL)
   b. Nonce-misuse setting (i.e., AEmL)
4. Leakage-resistant AE **designs** (& implementations)
5. Conclusions (& the need of open evaluations)

Seed: a leakage-resilient MAC



- 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

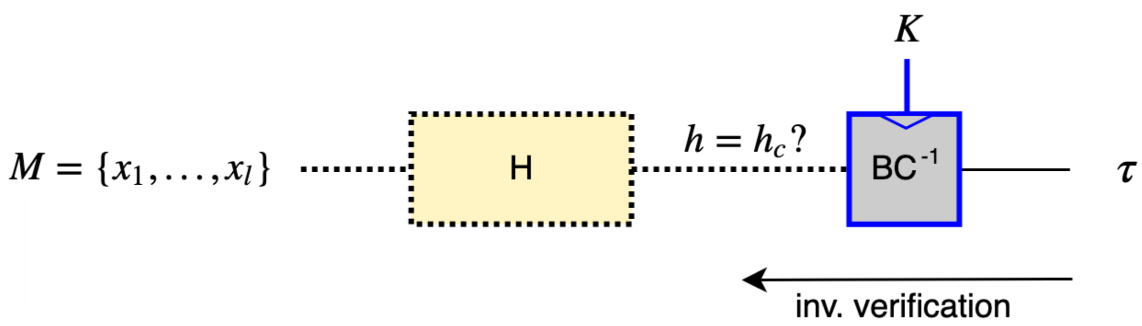
Seed: a leakage-resilient MAC



- 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

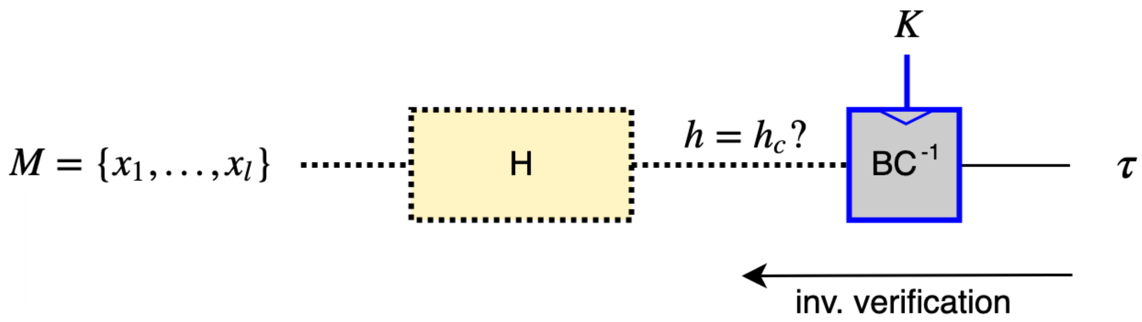
- Can we « minimize the attack » surface?

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 assumtions hard to find

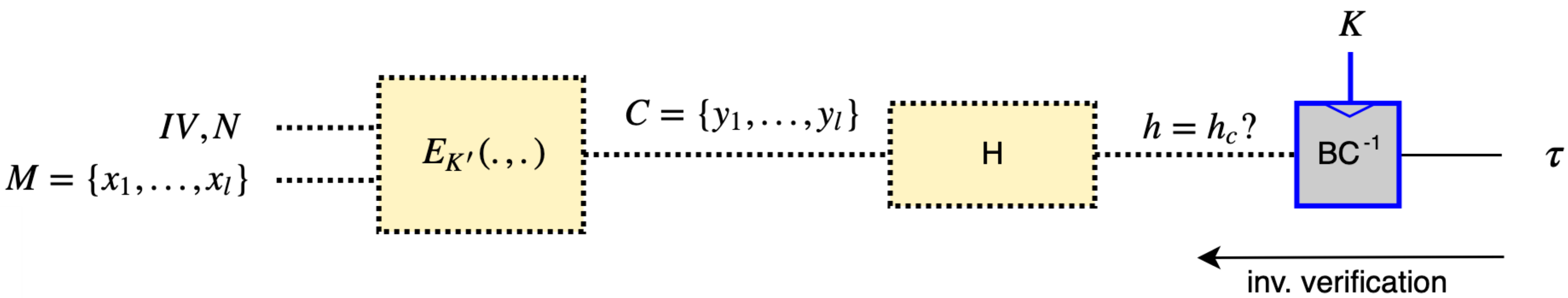
First tweak: LR tag verification



- 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 assumtions hard to find
- How to generalize this to an AE scheme?

An AE scheme satisfying CIML2

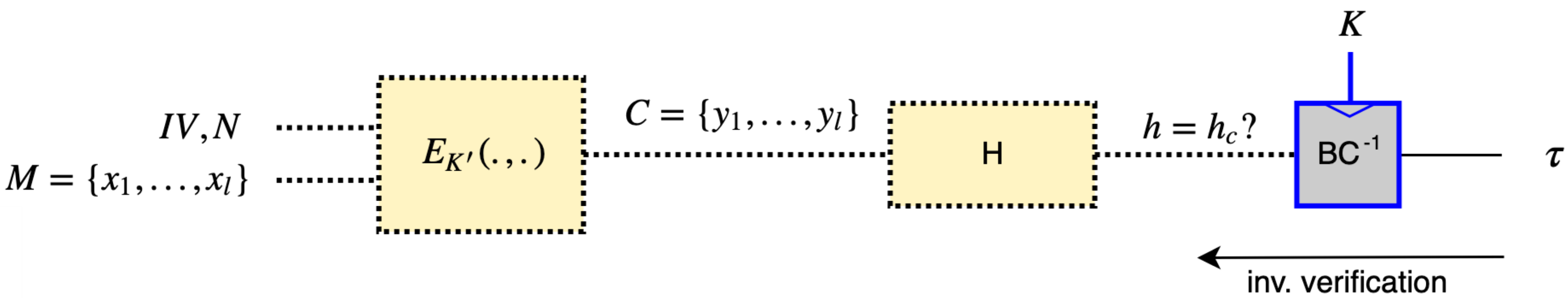
- First ignoring confidentiality with leakage



- 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)

An AE scheme satisfying CIML2

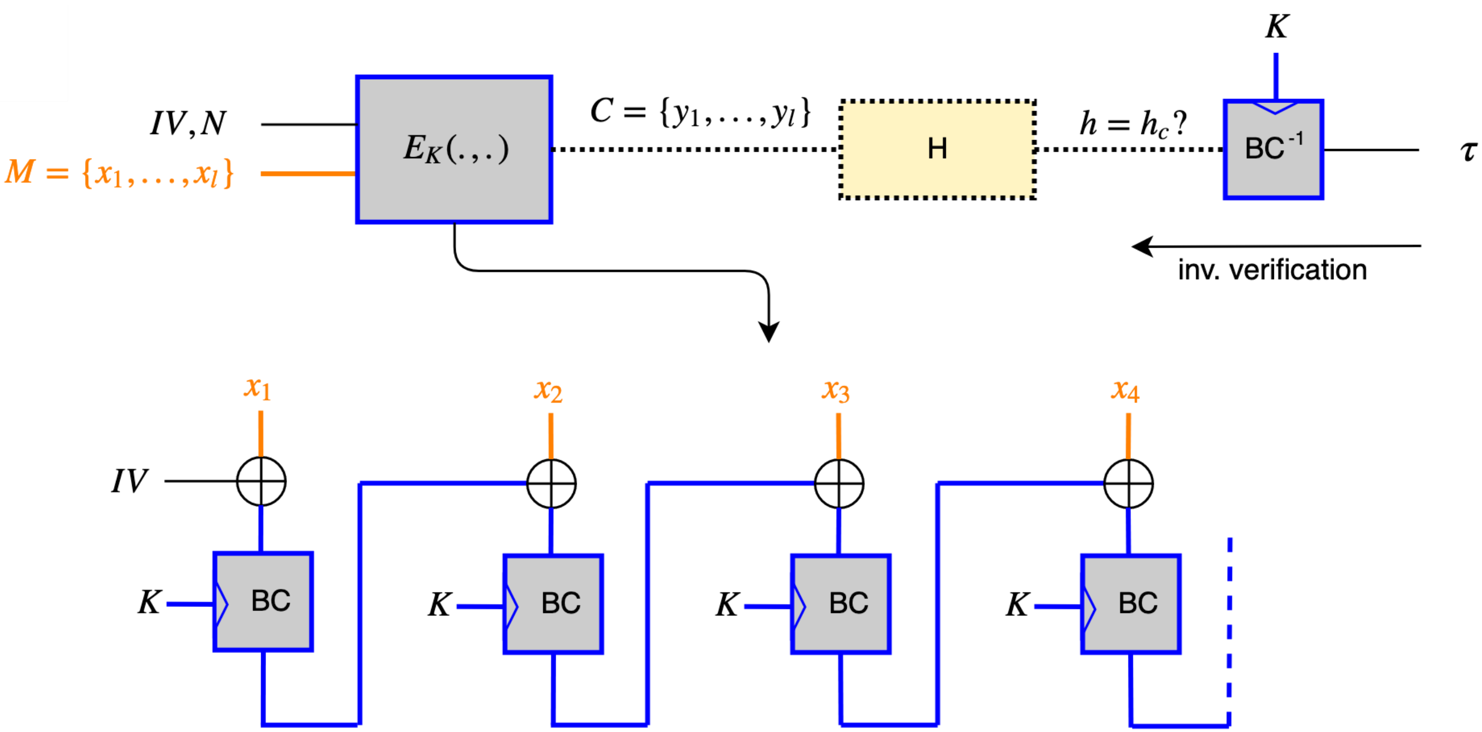
- First ignoring confidentiality with leakage



- 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)

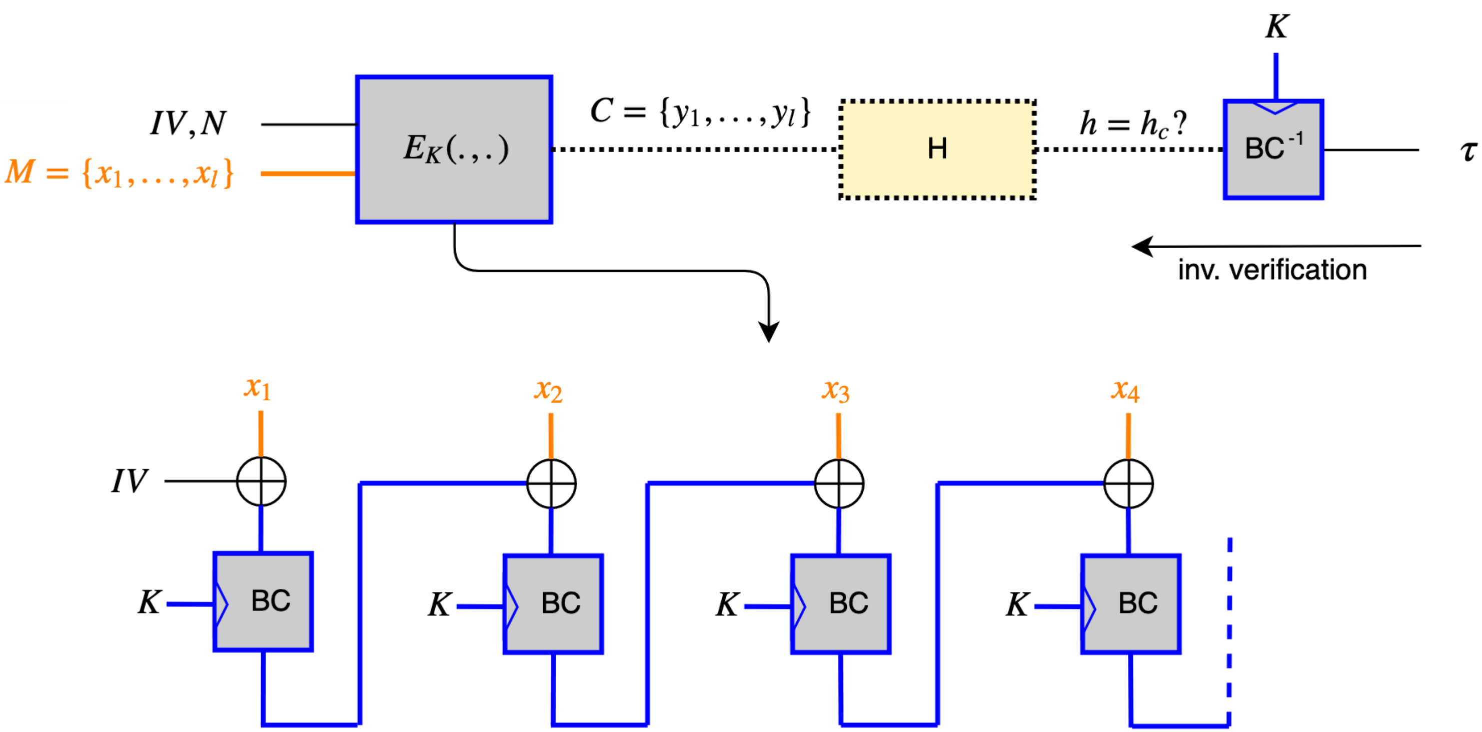
- How to add confidentiality guarantees?

Engineering approach to CCAL security



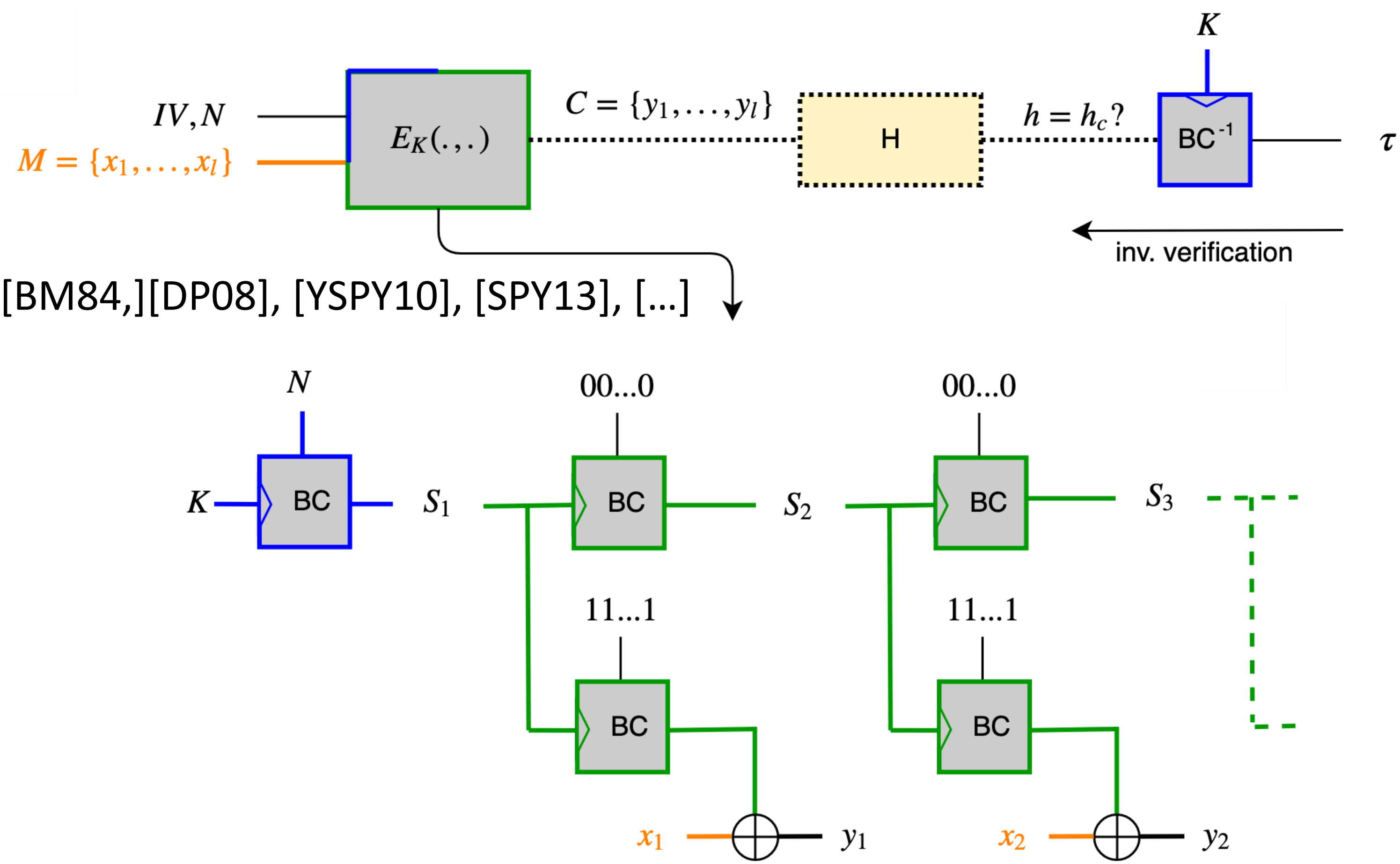
- Requires protecting all the BC blocks against DPA
  - And to deal with the (unavoidable) MC attacks

Engineering approach to CCAL security



- 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

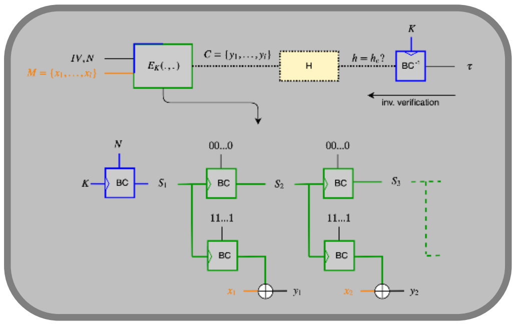


- Most BC executions can be protected against SPA only

(+ two DPA-secure BC calls and security against MC attacks)

Security reductions (simplified)

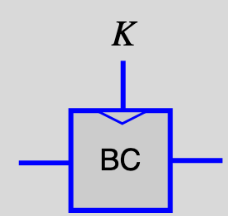
full fledged scheme

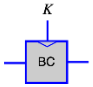


CIML2

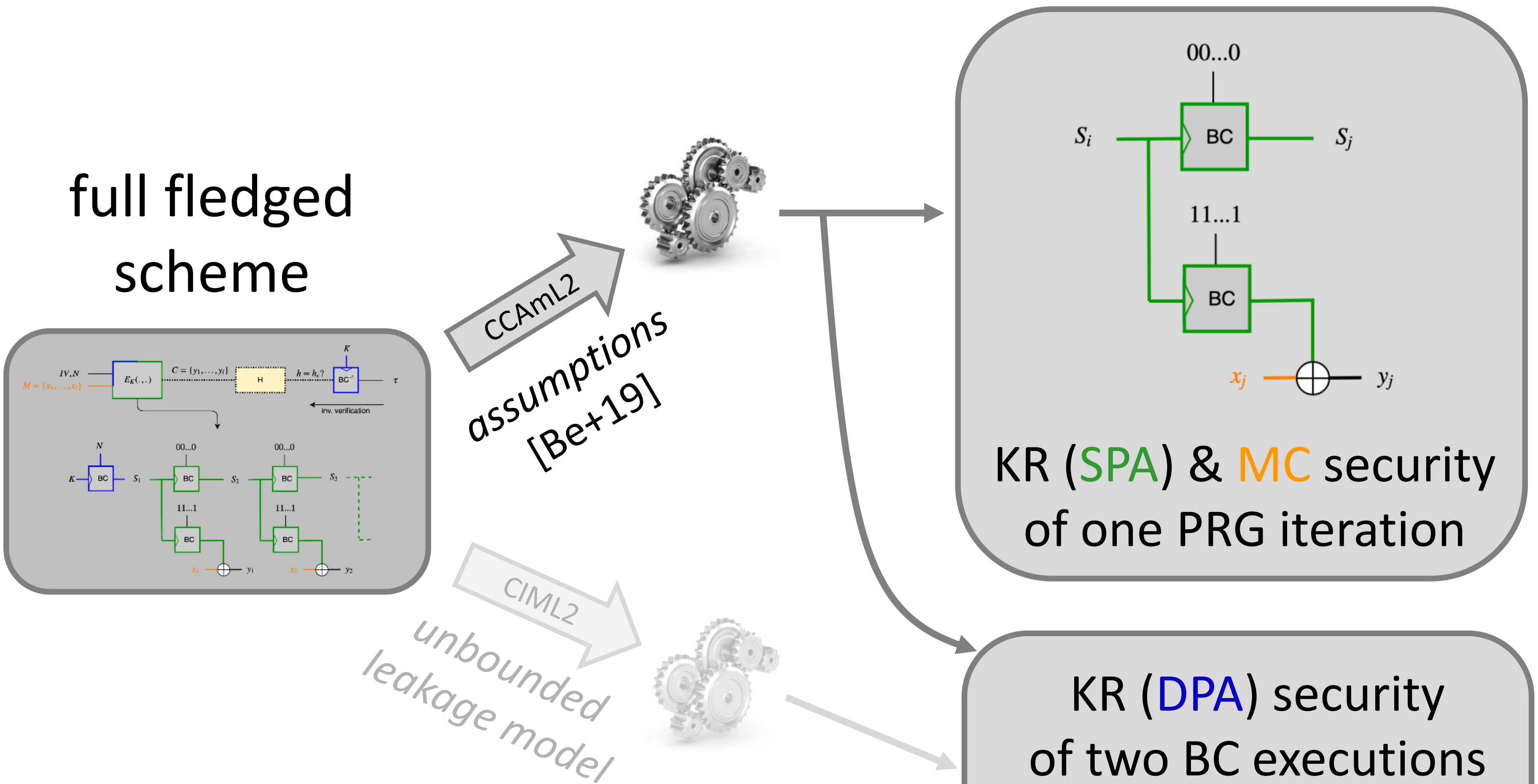
unbounded leakage model

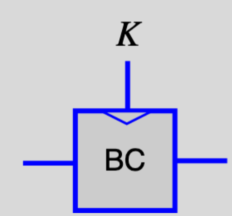
KR (DPA) security of two BC executions

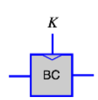


- Formally,  modeled as leak-free
- Graceful degradation seems possible

Security reductions (simplified)

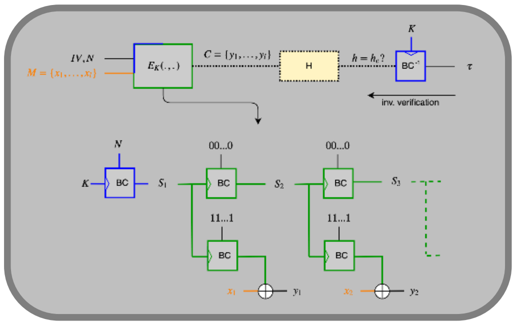




- Formally,  modeled as leak-free
- Graceful degradation seems hard

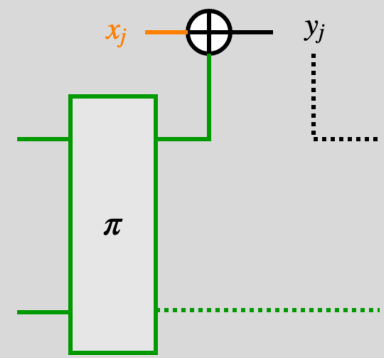
Security reductions (simplified)

full fledged scheme



CCAmL2

assumptions' [GPPS19]

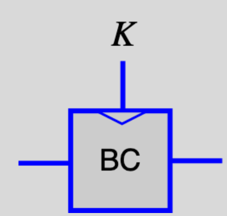


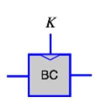
KR (SPA) & MC security of one PRG iteration

CIML2

unbounded leakage model

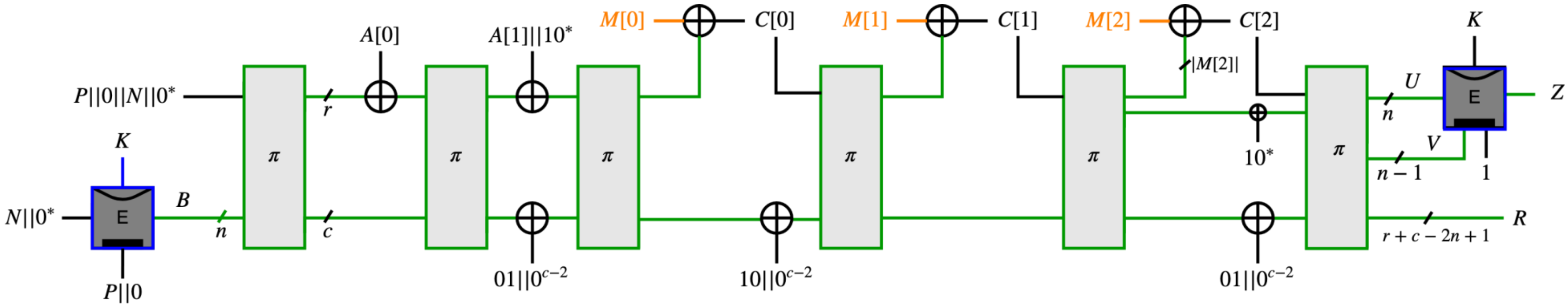
KR (DPA) security of two BC executions



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

Example of full-fledged scheme

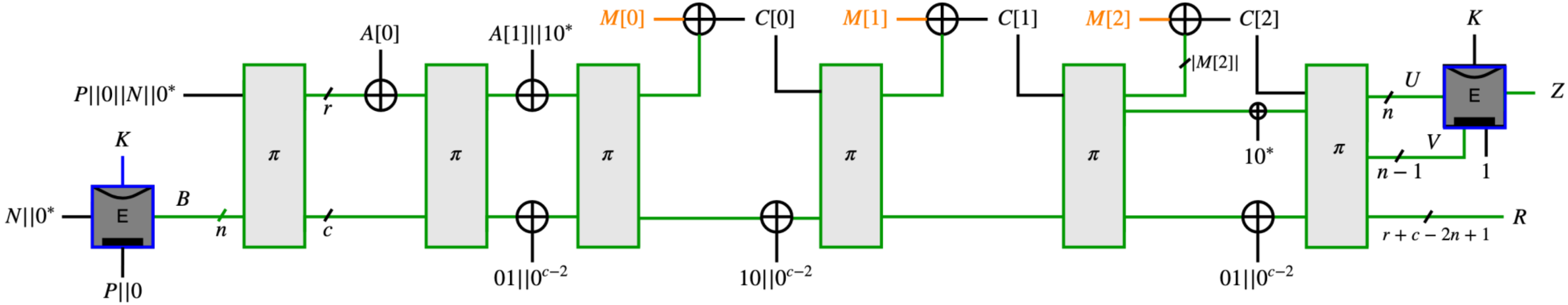
- S1P: 1-pass (online), CIML2, CCAmL1 [GPPS19]



- 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]

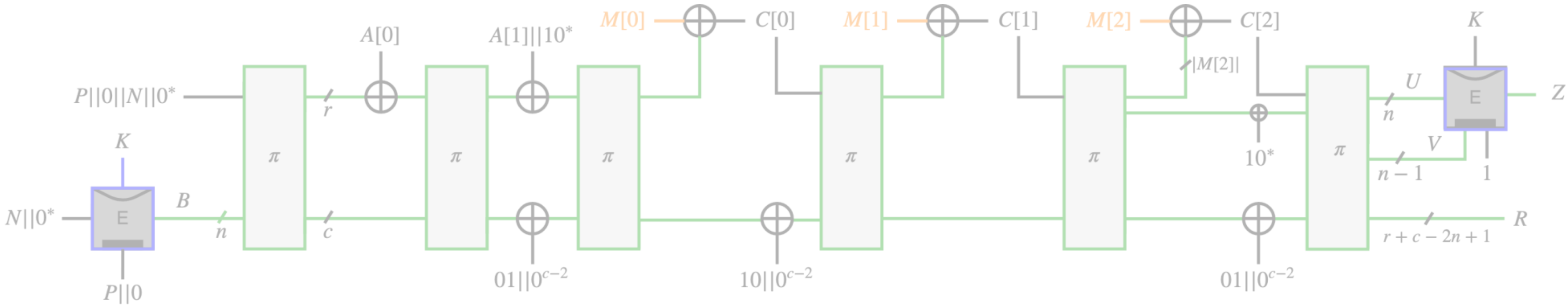


- Encourages « leveled implementations »
  - Strongly protected TBC: high-order masking
  - Weakly protected permutation: low-latency
- For such implementations, two different primitives are not an issue (since implementations are different)

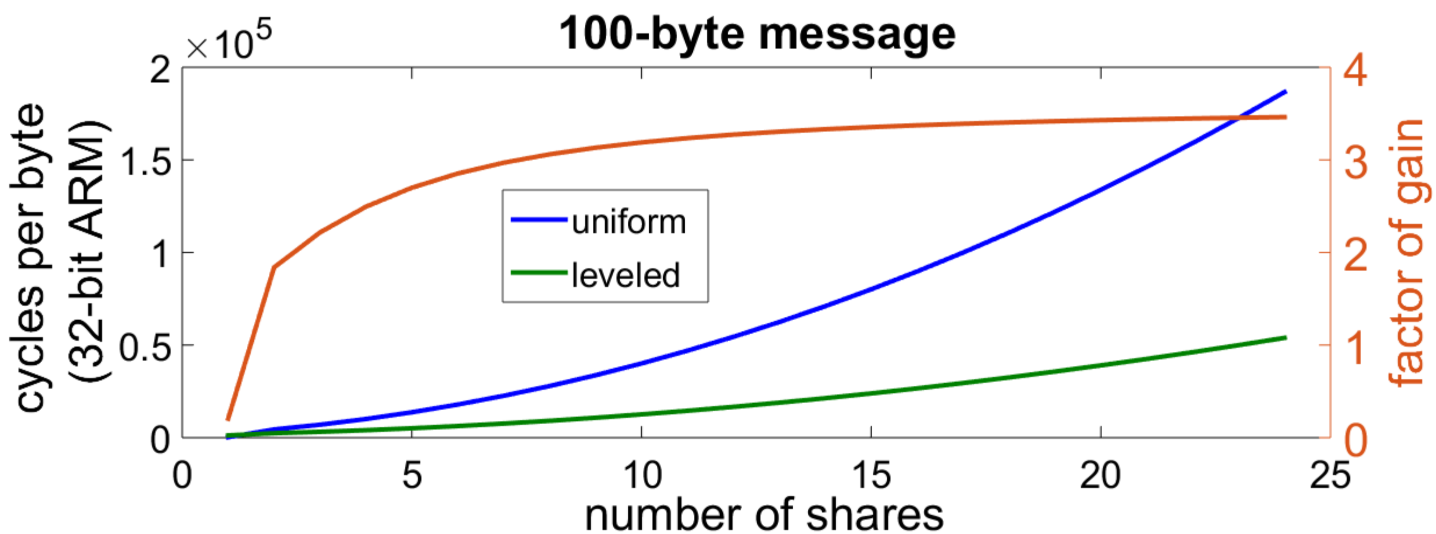
- (+ *beyond birthday w.r.t. TBC key, multi-user security*)

Example of full-fledged scheme

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

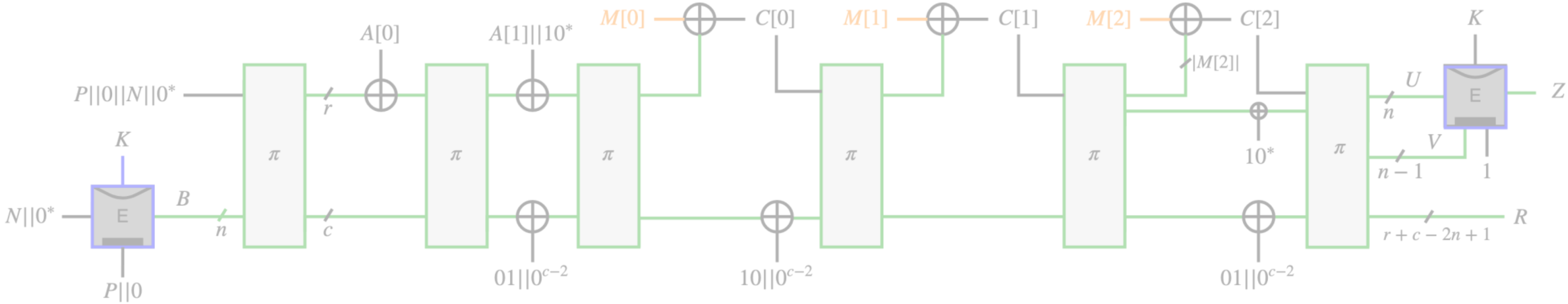


- Performance gains of leveled implementations

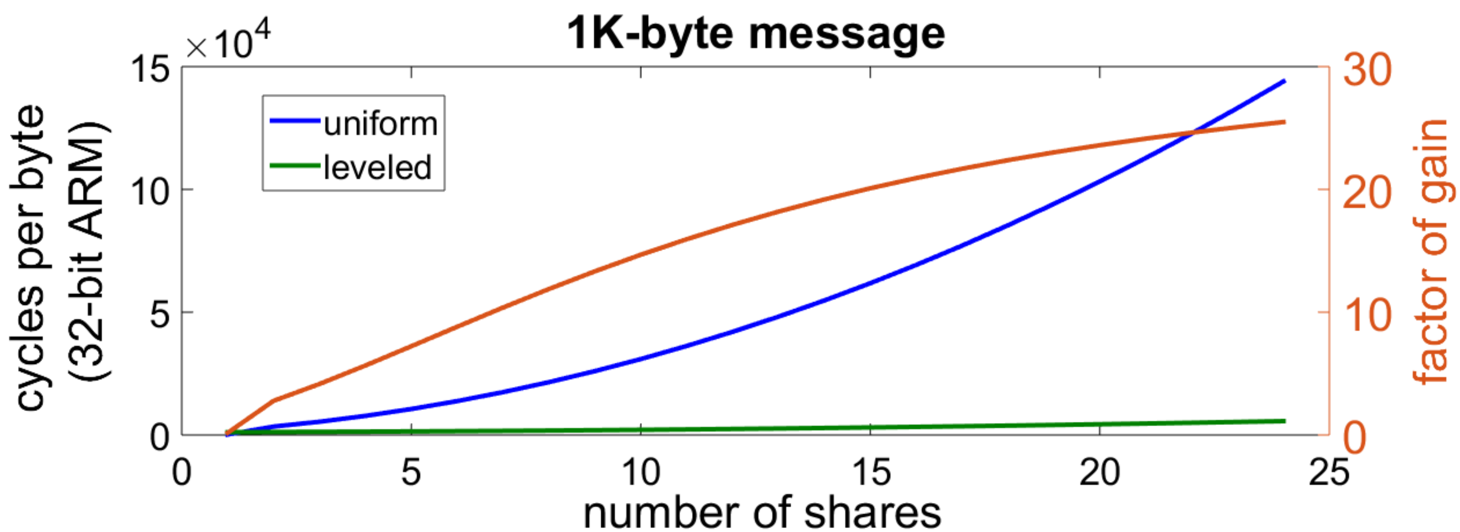


Example of full-fledged scheme

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



- Performance gains of leveled implementations



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)

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

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*

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, 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

Open problems

- 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, …)

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

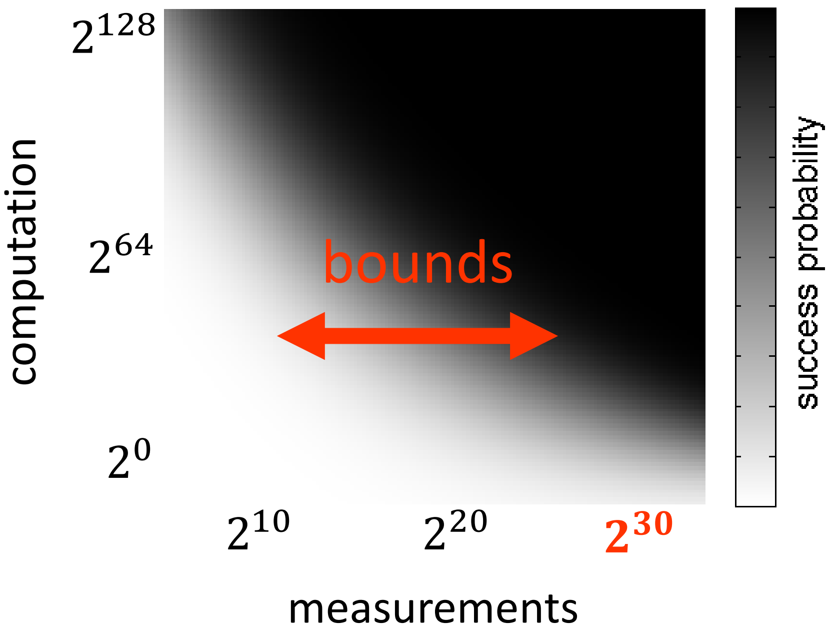
Open problems

- 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

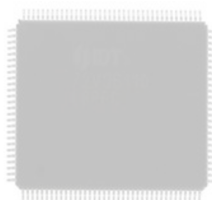
standard practice

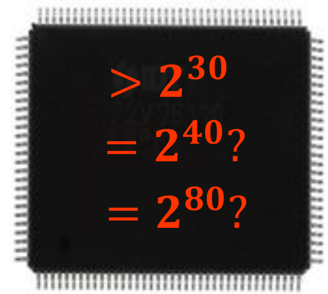
evidence-based evaluations
(*assumptions tested per device!*)



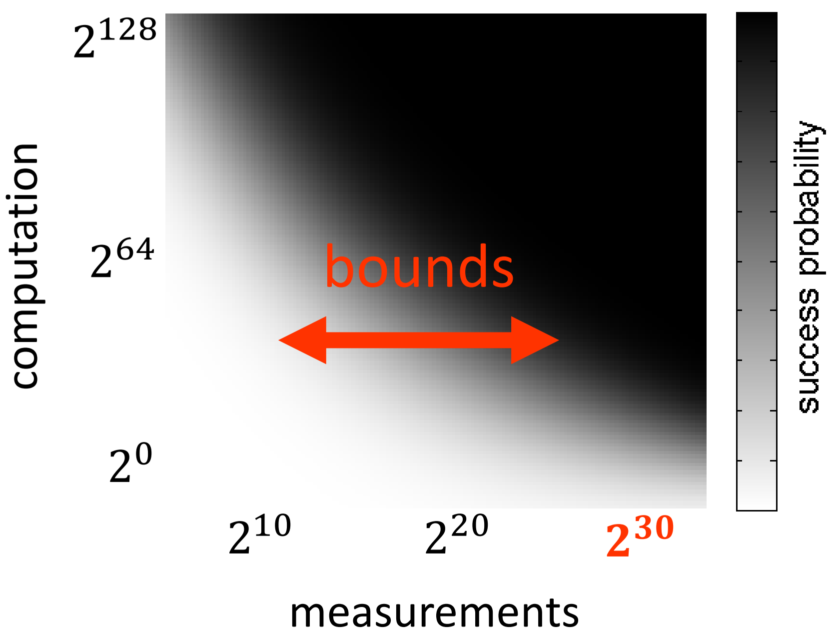
Evaluation challenge

standard practice





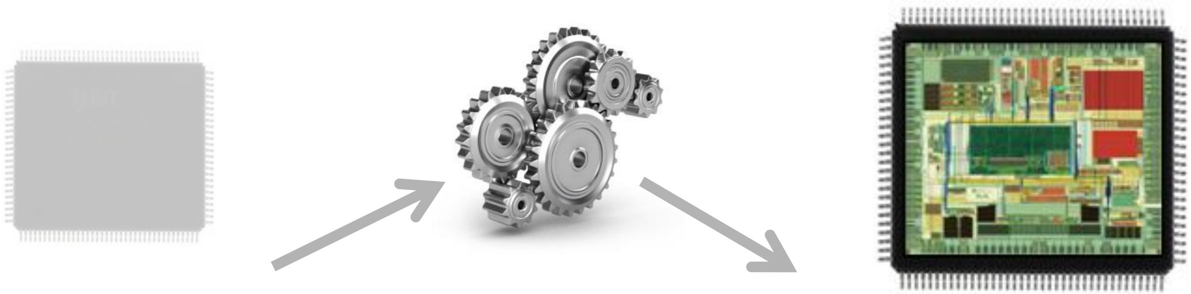
evidence-based evaluations
(*assumptions tested per device!*)



Evaluation challenge

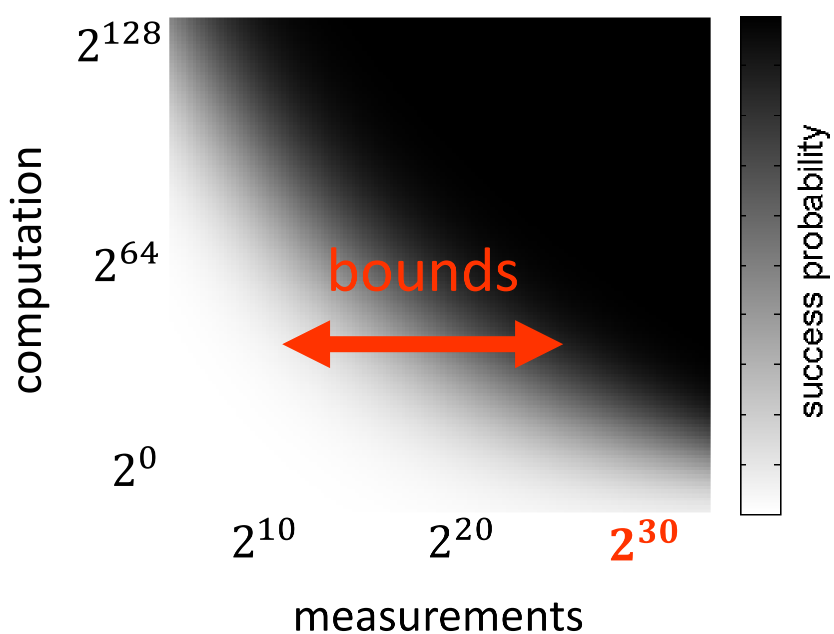
standard practice

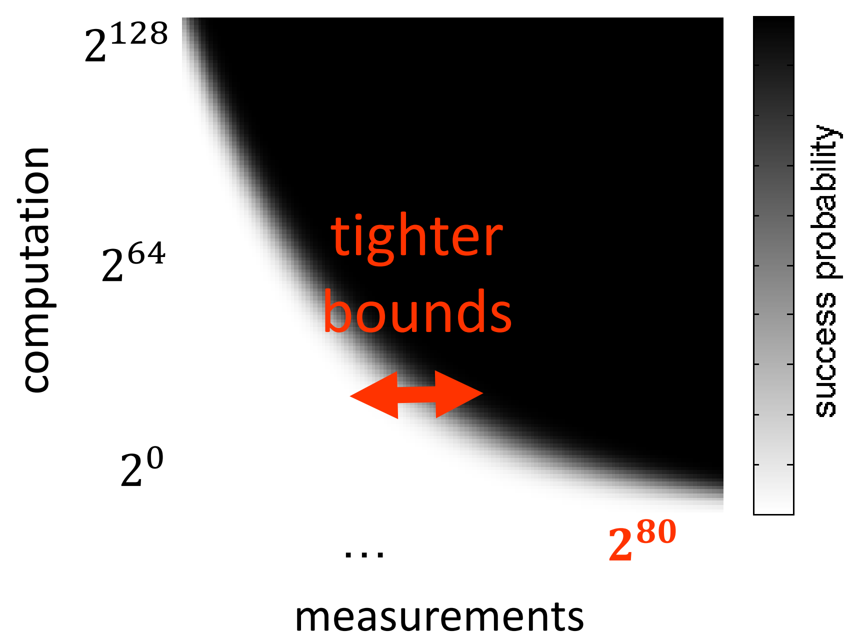
open design & evaluation



evidence-based evaluations
on reduced versions

proof-based evaluations [DFS15,GS18]



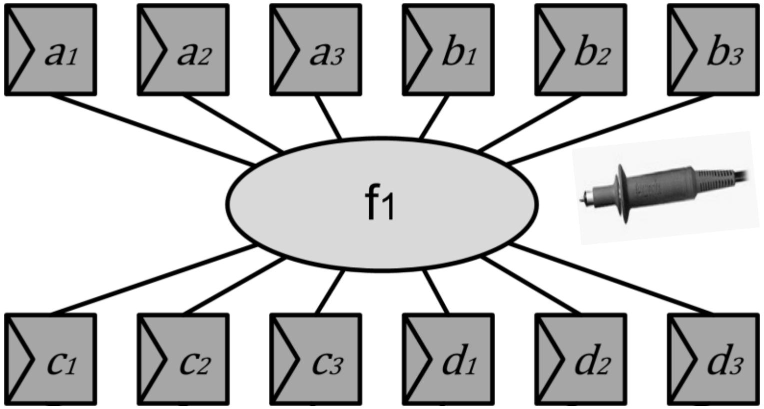


THANKS

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

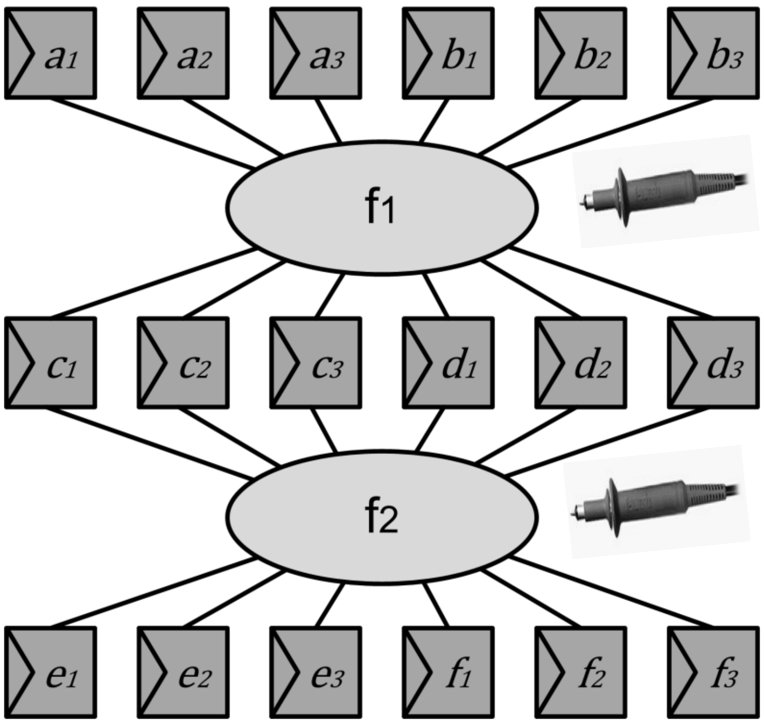
SUPPLEMENTARY SLIDES

Scalability & composability



***t*-probing security** [ISW03]
any t-tuple of shares in the protected circuit is independent of any sensitive variable

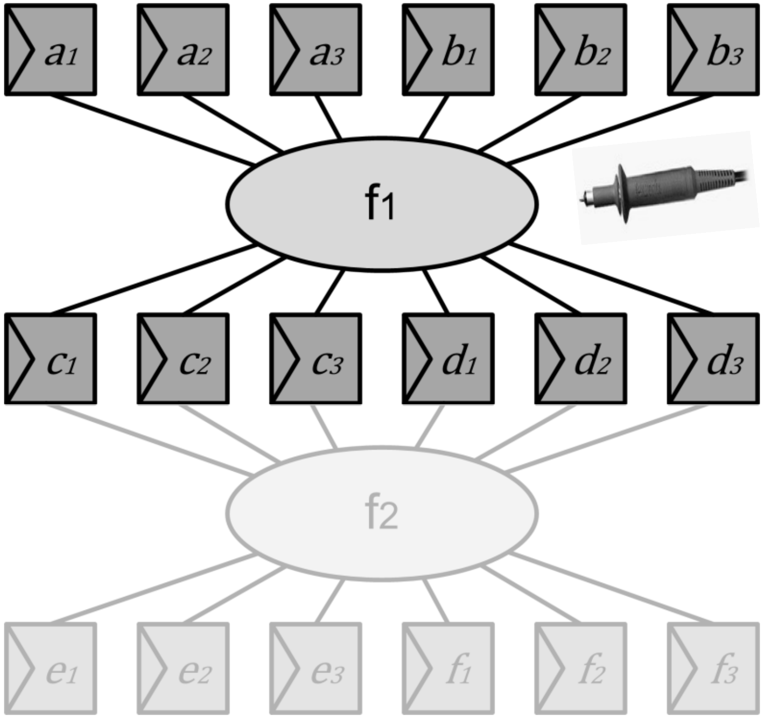
Scalability & composability



***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 (∃ many tuples) [Ba+15]

Scalability & composability A.1



$$q_1 + q_2 \leq q$$

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 (∃ many tuples) [Ba+15]

***q*-(Strong) Non Interference** [Ba+16]**:** a circuit gadget (e.g., f1) 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(input shares||probes) ≈ D(input shares||simulation)

Separation result (simplified)

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

Separation result (simplified)

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

- Let AEAD be CIML2 & CCAmL2 with ***Lenc*** and ***Ldec***
  - Define AEAD' such that
    - ***Lenc***'(K, M) = ***Lenc***(K, M)+***Lenc***(K', M')
    - ***Ldec***'(K, C)=***Ldec***(K, C)+***Lenc***(K', M')
  - AEAD' is still CIML2 but not CCAmL2 anymore
  - Attack: use the ***Ldec***' query to leak about M' and then use M' as challenge plaintext

Separation result (simplified) B.1

- 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

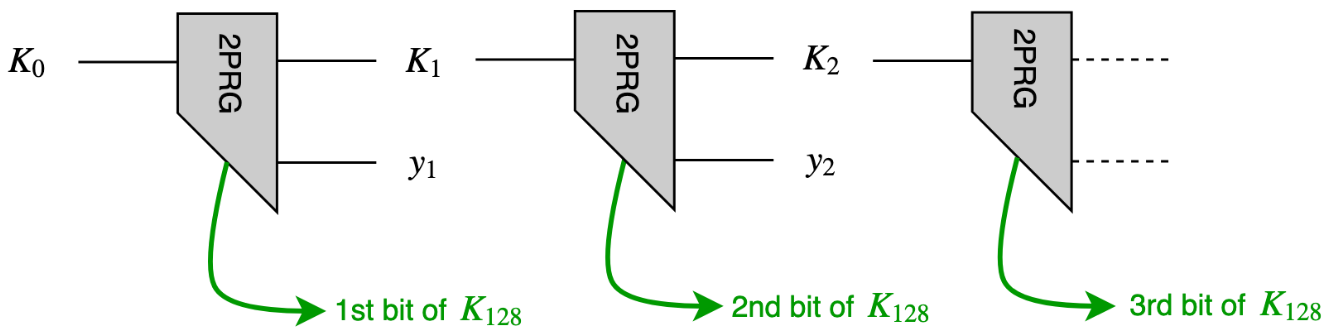
- First apparition of a recuring issue: somewhat artificial attack related to the difficulty to model ***L***

Physical assumptions (challenge) C.1

- Analyses must deal with the **information** and the **computation**al complexity of the leakage function

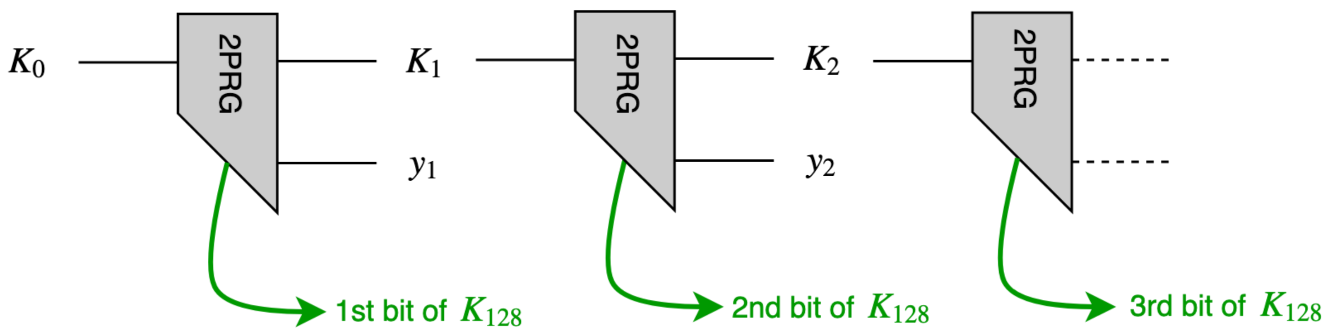
Physical assumptions (challenge)

- Analyses must deal with the **information** and the **computation**al complexity of the leakage function
  - Information: obvious (full leakage ⇒ no secrecy)
  - Computation: avoid « precomputation attacks » [DP08]



Physical assumptions (challenge) C.1

- Analyses must deal with the **information** and the **computation**al complexity of the leakage function
  - Information: obvious (full leakage ⇒ no secrecy)
  - Computation: avoid « precomputation attacks » [DP08]

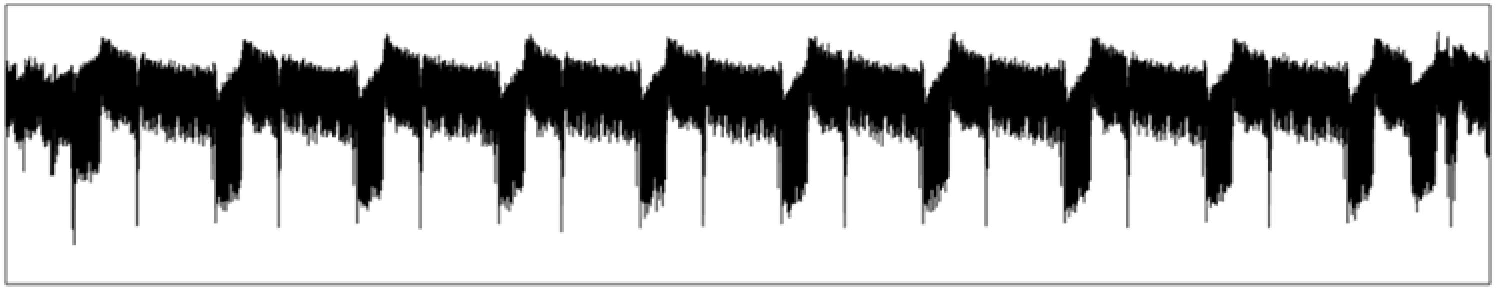


- (Second apparition of a recuring issue: somewhat artificial attack related to the difficulty to model ***L***)

Physical assumptions (challenge) C.1

- Analyses must deal with the **information** and the **computation**al 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) C.2

- Information restriction

HILL pseudoentropy [DP08]

seed-preserving PRG [YPSM10]
(≈ hard to invert leakages [DKL09])

Physical assumptions (symmetric SOTA) C.2

- Information restriction + computation restriction

HILL pseudoentropy [DP08]
only computation leaks [MR04] + alternating structure [DP08]

seed-preserving PRG [YPSM10] (≈ hard to invert leakages [DKL09,D+10])
oracle-free leakage function [YPMS10]

Physical assumptions (symmetric SOTA) C.2

- Information restriction + computation restriction

HILL pseudoentropy [DP08]

only computation leaks [MR04]
+ alternating structure [DP08]

simulatability [SPY13]

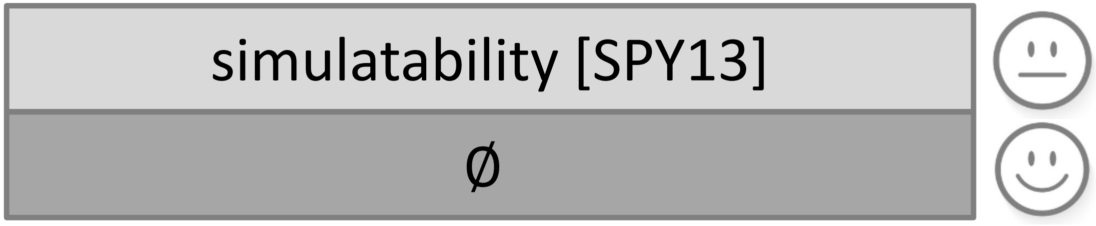
∅

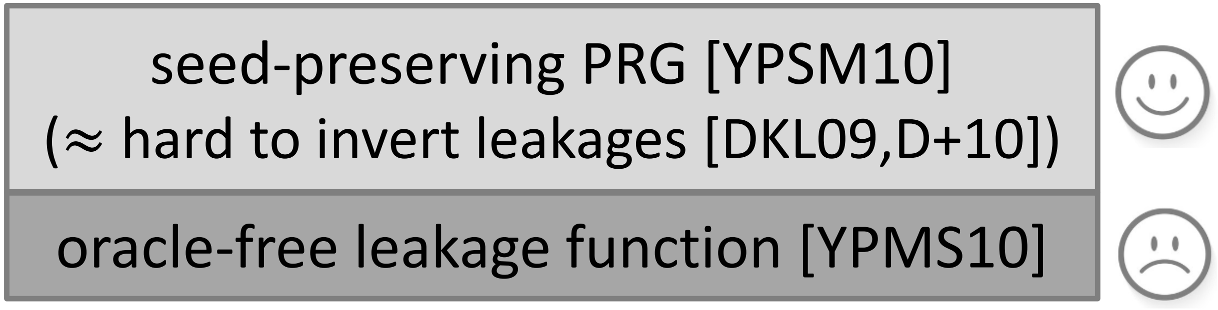
seed-preserving PRG [YPSM10]
(≈ hard to invert leakages [DKL09,D+10])

oracle-free leakage function [YPMS10]

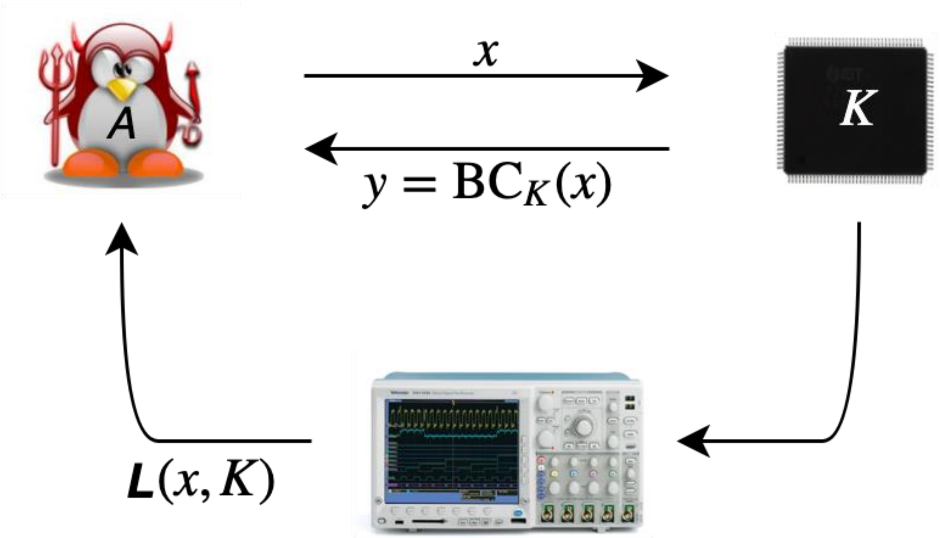
Physical assumptions (symmetric SOTA) C.2

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



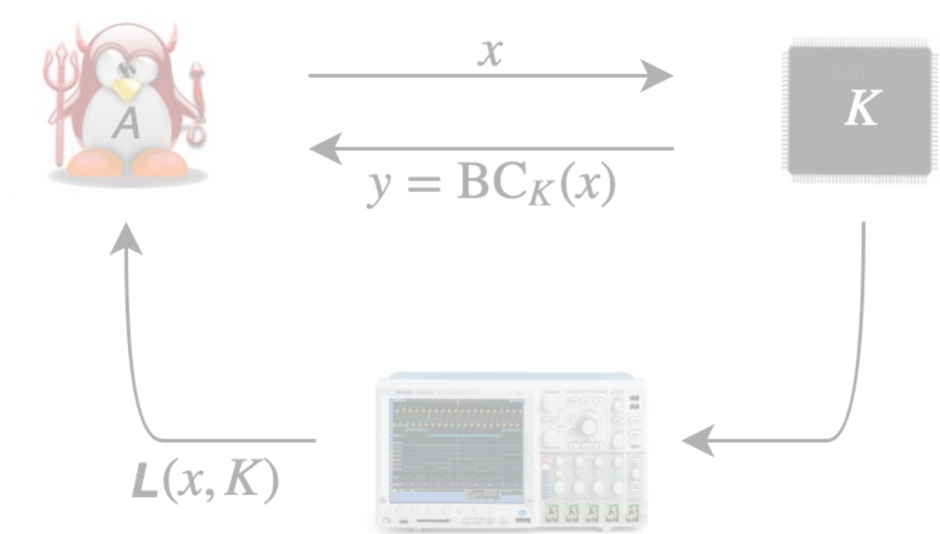


- $\boldsymbol{L}$ is hard to model ⇒ just don't model it
- Give public I/O access to device & setup



Simulatable leakage (I)

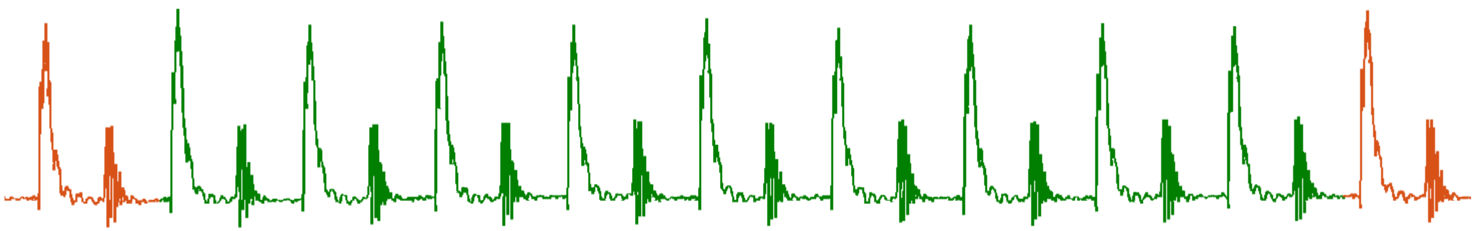
- $\boldsymbol{L}$ is hard to model ⇒ just don't model it
- Give public I/O access to device & setup



- Assume $\boldsymbol{L}(x, K)$ can be simulated
  - Using the same hardware as the target
  - But without knowing the secret key K

Simulatable leakages (II) C.4

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

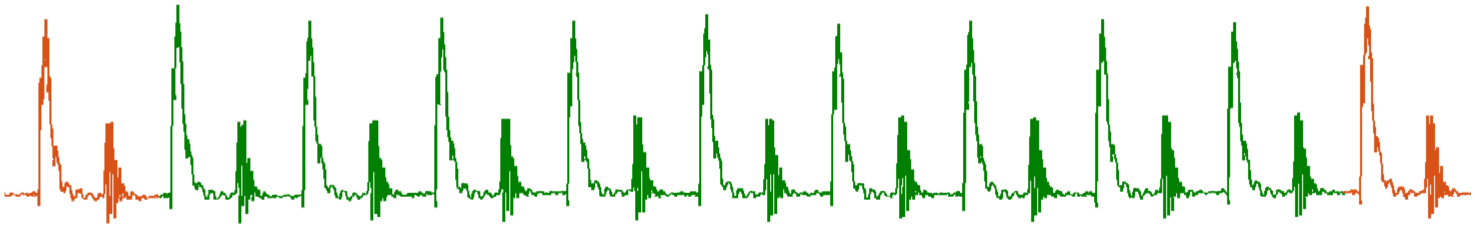


x | rounds (with fresh & unknown data) | $y = \mathrm{BC}_K(x)$

MC | SPA | MC

Simulatable leakages (II)

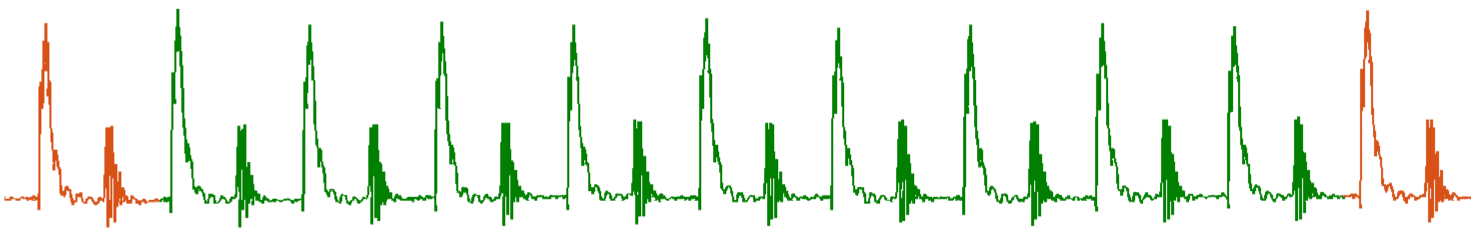
- $L(x, K)$ can be simulated without knowledge of K
  - *E.g.,* FPGA implementation, 128-bit architecture



- Simulated traces should be consistent with x and y

Simulatable leakages (II)

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

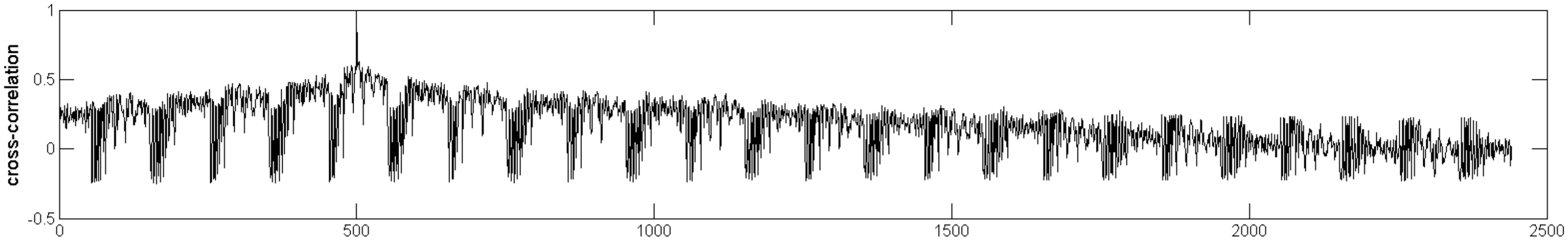


- Simulated traces should be consistent with x and y
- Simple proposal [SPY13]: « split & concatenate »

$$\boldsymbol{S}(x, K, y) = \boldsymbol{L}^{1/2}(x, K^*, y^*)\,||\,\boldsymbol{L}^{2/2}(x^*, K^*, y)$$

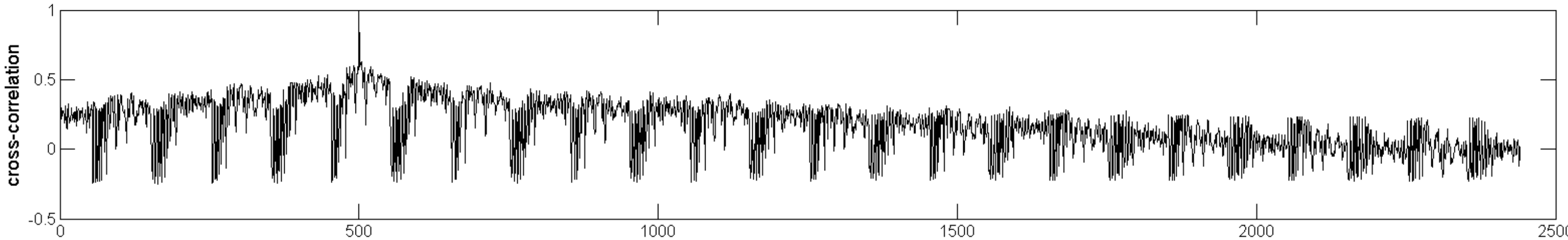
The Longo et al. distinguisher [Lo+14] C.5

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

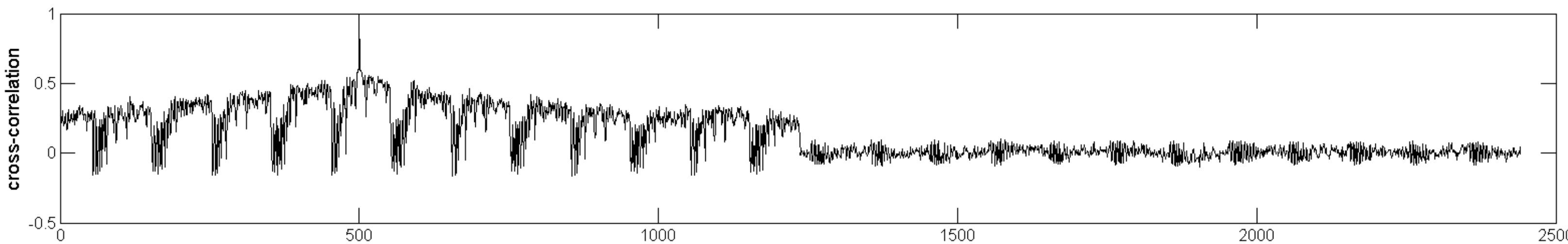


The Longo et al. distinguisher [Lo+14]

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

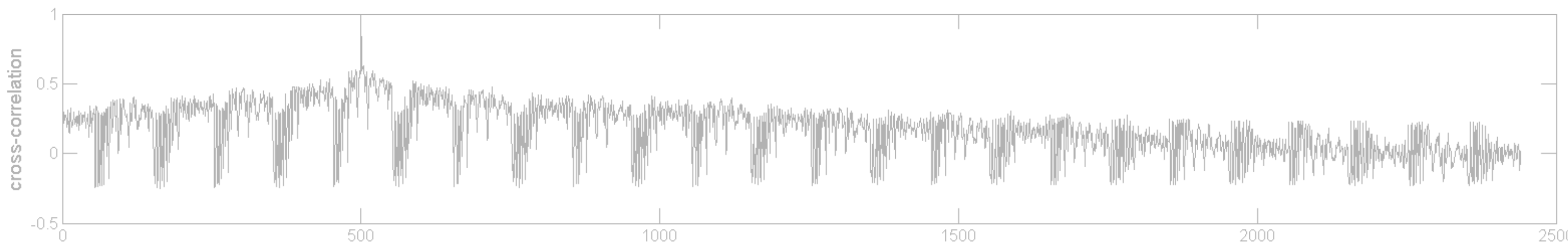


- Same, with simulated traces $\boldsymbol{L}^{1/2}(x, K^*, y^*)||\boldsymbol{L}^{2/2}(x^*, K^*, y)$

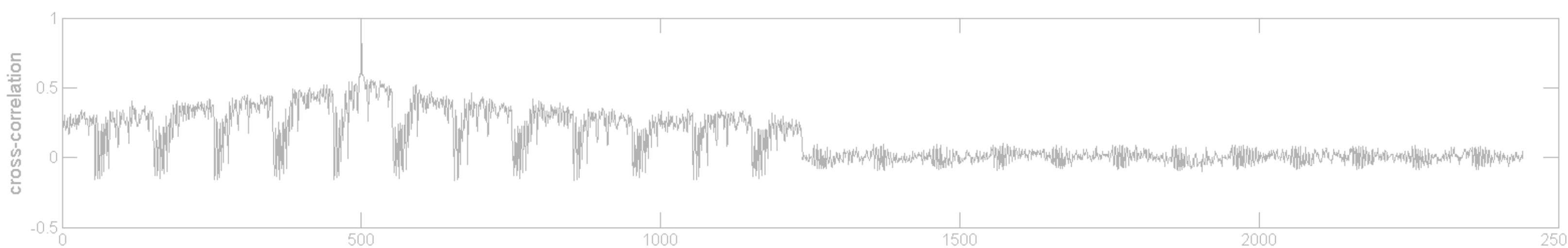


The Longo et al. distinguisher [Lo+14]

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



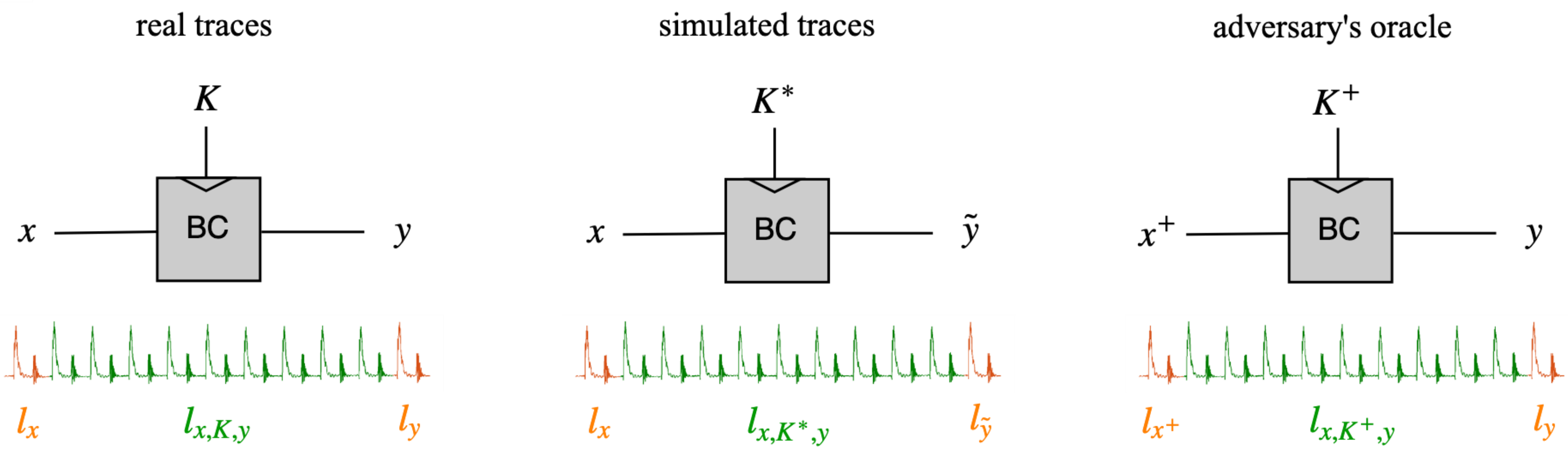
- Same, with simulated traces $\boldsymbol{L}^{1/2}(x, K^*, y^*)||\boldsymbol{L}^{2/2}(x^*, K^*, y)$



- Fixing this requires modeling the physics of ***L*** (*but the goal of simulatability was to avoid such modeling*)

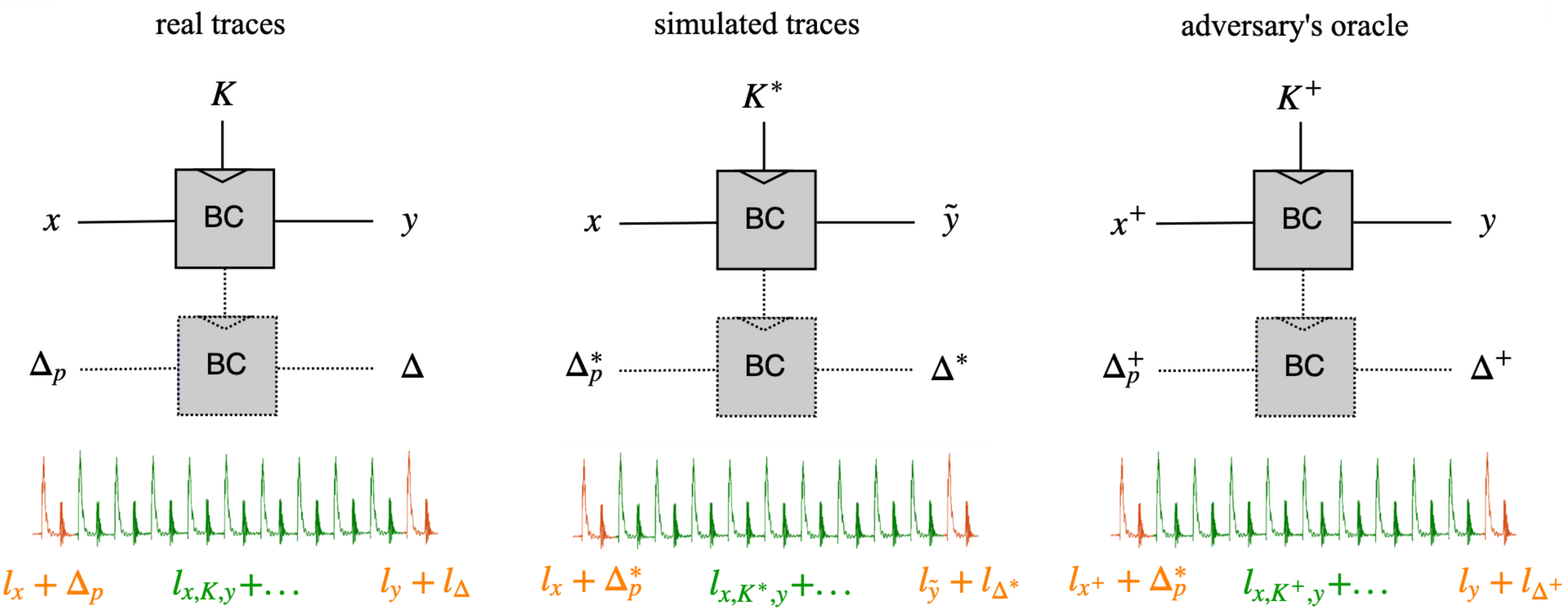
Another (new) approach

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



Another (new) approach

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



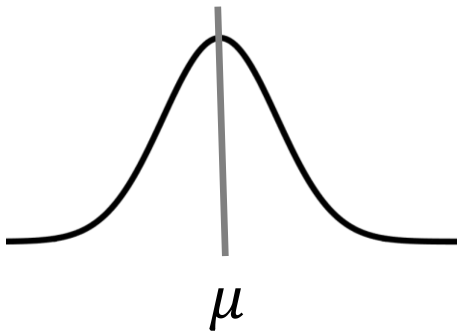
- Add key-dep. noise s.t. $\forall\, x, K, K^*$, $\mathrm{E}(l_\Delta) \neq \mathrm{E}(l_{\Delta^*})$

Statistical distinguisher

- Assume a perfect (additive) leakage model
- Remove the contribution l_y for each x, K pair

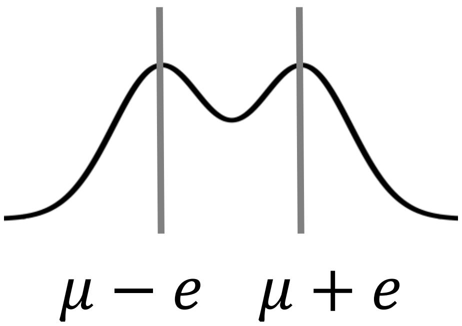
real distributions

l_Δ



simulated distributions

$l_{\Delta^*} + \left(l_y - l_{\tilde{y}}\right)$

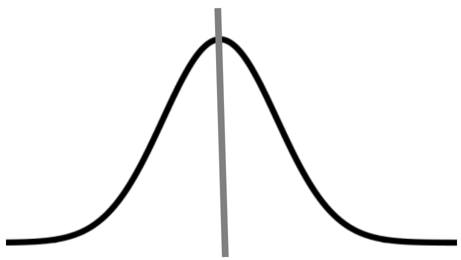


Statistical distinguisher C.6

- Assume a perfect (additive) leakage model
- Remove the contribution l_y for each x, K pair

real distributions

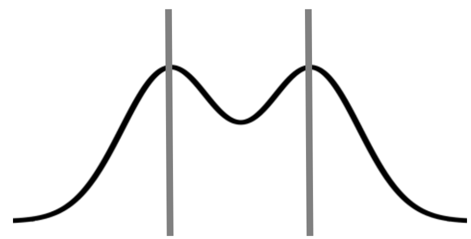
l_Δ



μ

simulated distributions

$l_{\Delta^*} + \left(l_y - l_{\tilde{y}}\right)$



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

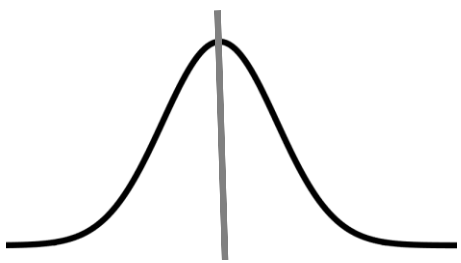
- The difference btw. variances decreases in e^2

Statistical distinguisher

- Assume a perfect (additive) leakage model
- Remove the contribution l_y for each x, K pair

real distributions

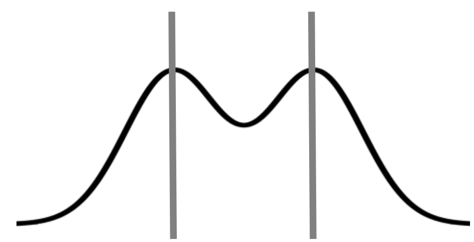
l_Δ



μ

simulated distributions

$l_{\Delta^*} + \left(l_y - l_{\tilde{y}}\right)$



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

- The difference btw. variances decreases in e^2
- Error e dercreases linearly in sim. complexity C
  - Depending on the leakage function (experiments needed)

Distinguishing complexity

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

Distinguishing complexity

- For key-dep. noise variance σ_Δ^2 and simulator error e, the simulation is distinguished in $\geq \frac{\left(\sigma_\Delta^2\right)^2}{e^2}$ 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 $\geq 2^{64}$

Distinguishing complexity

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

Distinguishing complexity

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

⇒ Reductions: k-bit key gives (k-c)-bit confidentiality

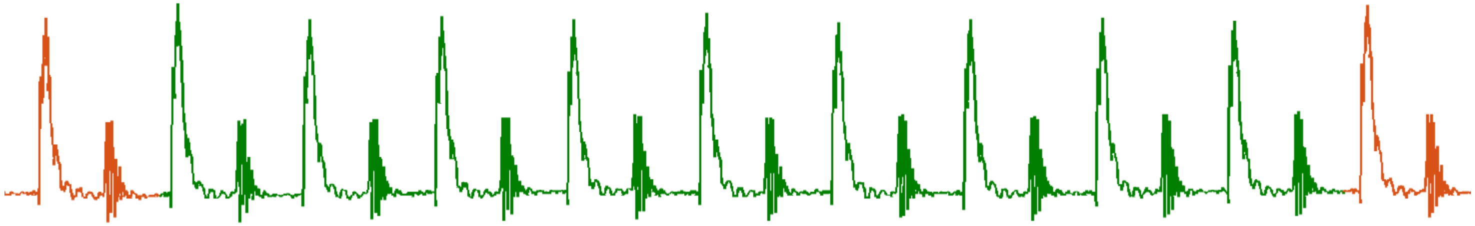
Distinguishing complexity

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

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

- (*Imperfect model:* $\underbrace{l_\Delta + e'}_{\text{real}}$ *and* $\underbrace{l_{\Delta^*} + e + e'}_{\text{simulation}}$ *get closer*)

Simulatability vs. pseudoentropy

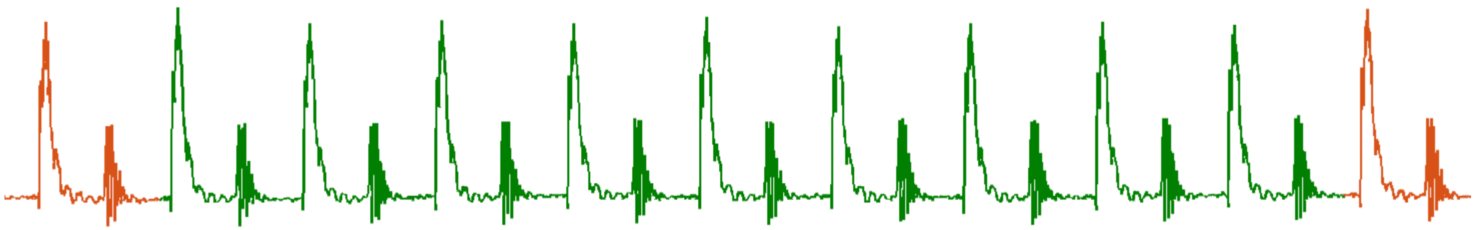


x rounds (fresh & unknown) $y = \mathrm{BC}_K(x)$

MC SPA MC

- [DP08] require $\mathrm{H}^{\mathrm{HILL}}(K|\boldsymbol{L}) > k - \lambda$
- Roughly: $\forall \boldsymbol{l}$, $\exists$ a set of $2^{k-\lambda}$ keys s.t. $\mathrm{BC}_{K^*}(x) \rightarrow \tilde{\boldsymbol{l}} \overset{\mathrm{c}}{\approx} \boldsymbol{l}$

Simulatability vs. pseudoentropy

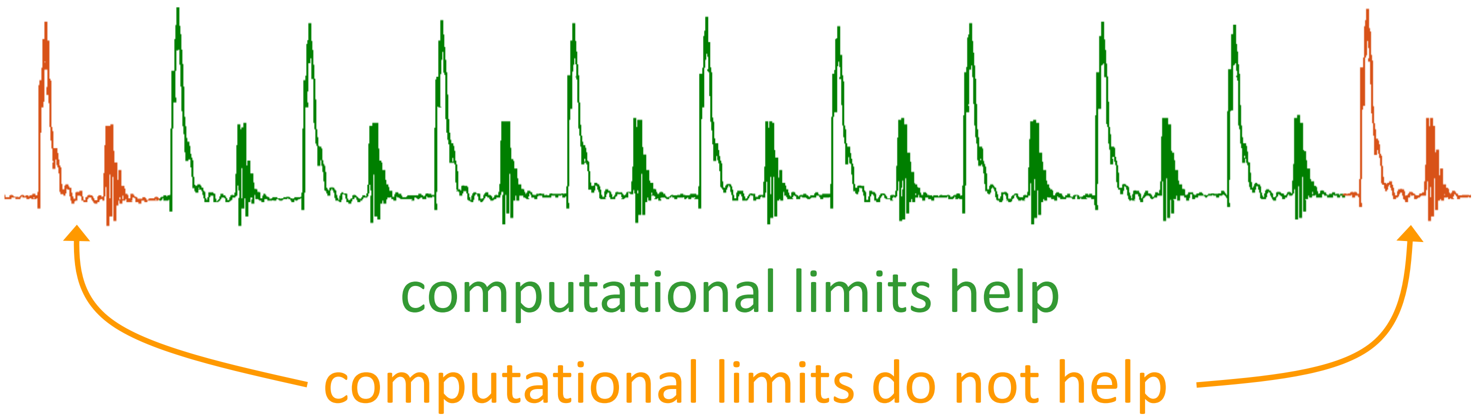


x rounds (fresh & unknown) $y = \mathrm{BC}_K(x)$

MC SPA MC

- [DP08] require $\mathrm{H}^{\mathrm{HILL}}(K|\boldsymbol{L}) > k - \lambda$
- Roughly: $\forall \boldsymbol{l}$, $\exists$ a set of $2^{k-\lambda}$ keys s.t. $\mathrm{BC}_{K^*}(x) \to \tilde{\boldsymbol{l}} \overset{\mathrm{c}}{\approx} \boldsymbol{l}$
- (Without the comp. limit., SPA samples become SC samples)

Simulatability vs. pseudoentropy



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

Simulatability vs. pseudoentropy

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

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

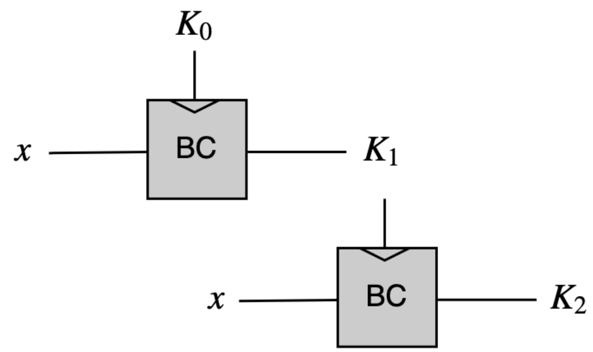
- Theory [FH15]: q-SIM and $\mathrm{H}^{\mathrm{HILL}}$ disconnected
- Cryptanalysis: q-SIM $\leq \mathrm{H}^{\mathrm{HILL}} \ll$ CCAmL(for 1 block)

Simulatable leakage definition

- More formally, (, ) has q-simulatable leakages of ∃ a simulator $S^{\boldsymbol{L}(.,.)}$ such that the bit b in the following game is hard to guess

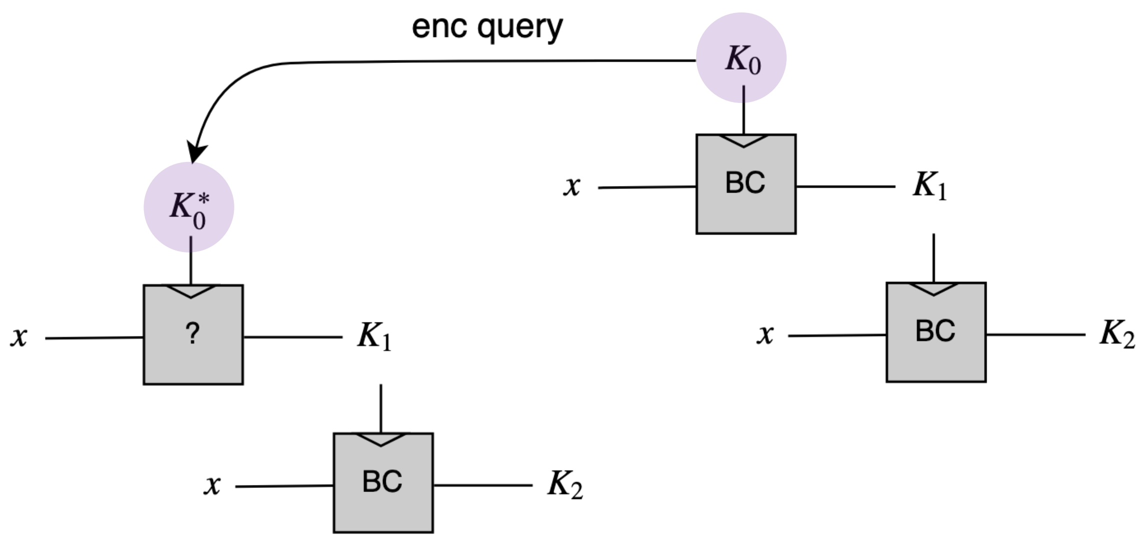
Game q-sim(A,, $S^{\boldsymbol{L}(.,.)}$,b) with $K, \boldsymbol{K}^*$ uniformly random		
q queries	response if $b = 0$	response if $b = 1$
enc(x)	$y = \mathrm{BC}_k(x)$, $\boldsymbol{L}(x, K)$	$y = \mathrm{BC}_k(x)$, $S^{\boldsymbol{L}(.,.)}(x, \boldsymbol{K}^*, y)$
1 query	response if $b = 0$	response if $b = 1$
gen(z, x)	$S^{\boldsymbol{L}(.,.)}(\mathrm{z}, x, K)$	$S^{\boldsymbol{L}(.,.)}(\mathrm{z}, x, \boldsymbol{K}^*)$

- (Not exactly real vs. simulated due to the gen query)



$\rightarrow \boldsymbol{L}\,(x, K_0, K_1)\ ||\ \boldsymbol{L}\,(x, K_1, K_2)$

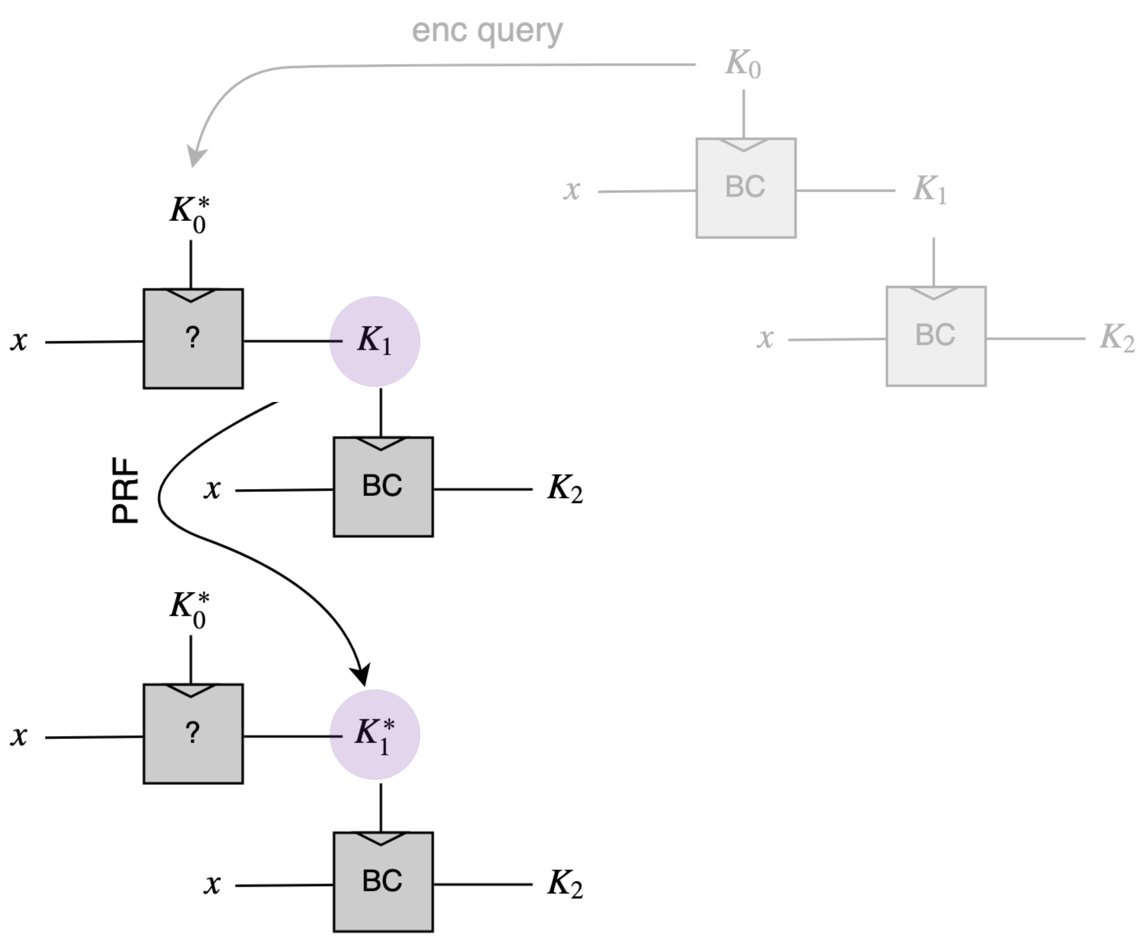
Proof intuition



$\rightarrow$ $\boldsymbol{L}\ (x, K_0, K_1)\ \ ||\ \boldsymbol{L}\ (x, K_1, K_2)$

$\rightarrow$ $\boldsymbol{S}\ (x, K_0^*, K_1)\ \ ||\ \boldsymbol{L}\ (x, K_1, K_2)$

Proof intuition

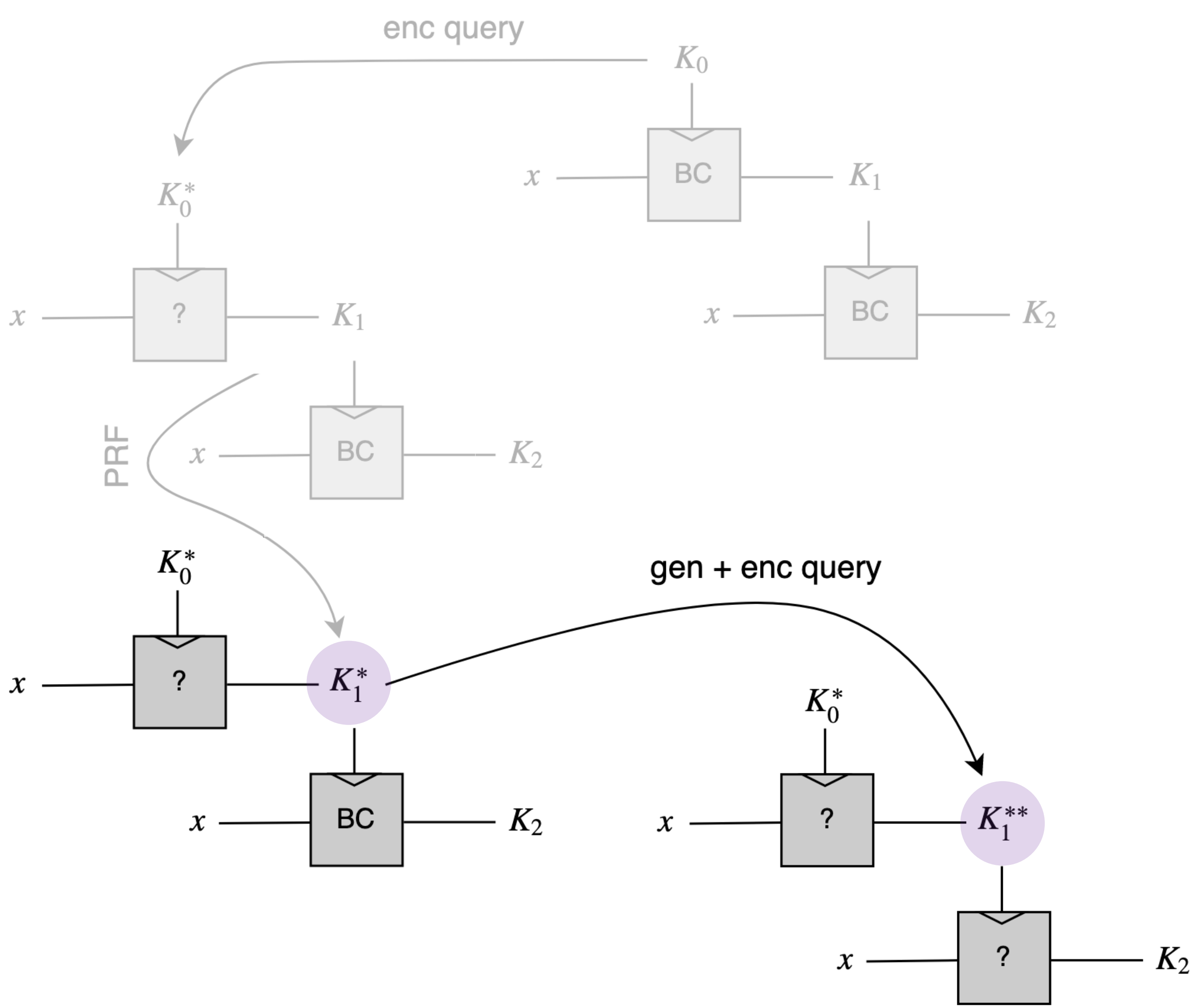


$\rightarrow$ $\boldsymbol{L}\,(x, K_0, K_1)\ ||\ \boldsymbol{L}\,(x, K_1, K_2)$

$\rightarrow$ $\boldsymbol{S}\,(x, K_0^*, K_1)\ ||\ \boldsymbol{L}\,(x, K_1, K_2)$

$\rightarrow$ $\boldsymbol{S}\,(x, K_0^*, K_1^*)\ ||\ \boldsymbol{L}\,(x, K_1, K_2)$

Proof intuition



$\rightarrow$ $\boldsymbol{L}\ (x, K_0, K_1)\ \ ||\ \boldsymbol{L}\ (x, K_1, K_2)$

$\rightarrow$ $\boldsymbol{S}\ (x, K_0^*, K_1)\ \ ||\ \boldsymbol{L}\ (x, K_1, K_2)$

$\rightarrow$ $\boldsymbol{S}\ (x, K_0^*, K_1^*)\ \ ||\ \boldsymbol{L}\ (x, K_1, K_2)$

$\rightarrow$ $\boldsymbol{S}\ (x, K_0^*, K_1^*)\ \ ||\ \boldsymbol{S}\ (x, K_1, K_2)$

Bibliography

[ADL17] Tomer Ashur, Orr Dunkelman, Atul Luykx: Boosting Authenticated Encryption Robustness with Minimal Modifications. CRYPTO (3) 2017: 3-33. **[AP13]** Nadhem J. AlFardan, Kenneth G. Paterson: Lucky Thirteen: Breaking the TLS and DTLS Record Protocols. IEEE Symposium on Security and Privacy 2013: 526-540. **[BCPZ16]** Alberto Battistello, Jean-Sébastien Coron, Emmanuel Prouff, Rina Zeitoun: Horizontal Side-Channel Attacks and Countermeasures on the ISW Masking Scheme. CHES 2016: 23-39. **[BG10]** Zvika Brakerski, Shafi Goldwasser: Circular and Leakage Resilient Public-Key Encryption under Subgroup Indistinguishability - (or: Quadratic Residuosity Strikes Back). CRYPTO 2010: 1-20. **[BMOS17]** Guy Barwell, Daniel P. Martin, Elisabeth Oswald, Martijn Stam: Authenticated Encryption in the Face of Protocol and Side Channel Leakage. ASIACRYPT (1) 2017: 693-723. **[BPPS17]** Francesco Berti, Olivier Pereira, Thomas Peters, François-Xavier Standaert: On Leakage-Resilient Authenticated Encryption with Decryption Leakages. IACR Trans. Symmetric Cryptol. 2017(3): 271-293 (2017). **[Ba+15]** Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, Pierre-Yves Strub: Verified Proofs of Higher-Order Masking. EUROCRYPT (1) 2015: 457-485. **[Ba+16]** Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, Pierre-Yves Strub, Rébecca Zucchini: Strong Non-Interference and Type-Directed Higher-Order Masking. ACM Conference on Computer and Communications Security 2016: 116-129. **[Be+16]** Sonia Belaïd, Fabrice Benhamouda, Alain Passelègue, Emmanuel Prouff, Adrian Thillard, Damien Vergnaud: Randomness Complexity of Private Circuits for Multiplication. EUROCRYPT (2) 2016: 616-648.

[Be+18] Francesco Berti, François Koeune, Olivier Pereira, Thomas Peters, François-Xavier Standaert: Ciphertext Integrity with Misuse and Leakage: Definition and Efficient Constructions with Symmetric Primitives. AsiaCCS 2018: 37-50. **[Be+19]** Francesco Berti, Chun Guo, Olivier Pereira, Thomas Peters, François-Xavier Standaert: TEDT, a Leakage-Resilient AEAD mode for High (Physical) Security Applications. IACR Cryptology ePrint Archive 2019: 137 (2019). **[CHV03]** Brice Canvel, Alain P. Hiltgen, Serge Vaudenay, Martin Vuagnoux: Password Interception in a SSL/TLS Channel. CRYPTO 2003: 583-599. **[CS19]** Gaetan Cassiers, François-Xavier Standaert: Towards Globally Optimized Masking: From Low Randomness to Low Noise Rate or Probe Isolating Multiplications with Reduced Randomness and Security against Horizontal Attacks. IACR Trans. Cryptogr. Hardw. Embed. Syst. 2019(2): 162-198 (2019). **[DDF14]** Alexandre Duc, Stefan Dziembowski, Sebastian Faust: Unifying Leakage Models: From Probing Attacks to Noisy Leakage. EUROCRYPT 2014: 423-440. **[DFS15]** Alexandre Duc, Sebastian Faust, François-Xavier Standaert: Making Masking Security Proofs Concrete - Or How to Evaluate the Security of Any Leaking Device. EUROCRYPT (1) 2015: 401-429. **[DKL09]** Yevgeniy Dodis, Yael Tauman Kalai, Shachar Lovett: On cryptography with auxiliary input. STOC 2009: 621-630. **[DM19]** Christoph Dobraunig, Bart Mennink: Leakage Resilience of the Duplex Construction. IACR Cryptology ePrint Archive 2019: 225 (2019). **[DP08]** Stefan Dziembowski, Krzysztof Pietrzak: Leakage-Resilient Cryptography. FOCS 2008: 293-302.

[D+10] Yevgeniy Dodis, Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, Vinod Vaikuntanathan: Public-Key Encryption Schemes with Auxiliary Inputs. TCC 2010: 361-381. **[FPS12]** Sebastian Faust, Krzysztof Pietrzak, Joachim Schipper: Practical Leakage-Resilient Symmetric Cryptography. CHES 2012: 213-232. **[FH15]** Benjamin Fuller, Ariel Hamlin: Unifying Leakage Classes: Simulatable Leakage and Pseudoentropy. ICITS 2015: 69-86. **[F+18]** Sebastian Faust, Vincent Grosso, Santos Merino Del Pozo, Clara Paglialonga, François-Xavier Standaert: Composable Masking Schemes in the Presence of Physical Defaults & the Robust Probing Model. IACR Trans. Cryptogr. Hardw. Embed. Syst. 2018(3): 89-120 (2018). **[GGM84]** Oded Goldreich, Shafi Goldwasser, Silvio Micali: How to Construct Random Functions (Extended Abstract). FOCS 1984: 464-479. **[GM18]** Hannes Groß, Stefan Mangard: A unified masking approach. J. Cryptographic Engineering 8(2): 109-124 (2018). **[GPPS18]** Chun Guo, Olivier Pereira, Thomas Peters, François-Xavier Standaert: Leakage-Resilient Authenticated Encryption with Misuse in the Leveled Leakage Setting: Definitions, Separation Results, and Constructions. IACR Cryptology ePrint Archive 2018: 484 (2018). **[GPPS19]** Chun Guo, Olivier Pereira, Thomas Peters, François-Xavier Standaert: Towards Lightweight Side-Channel Security and the Leakage-Resilience of the Duplex Sponge. IACR Cryptology ePrint Archive 2019: 193 (2019). **[GS18]** Vincent Grosso, François-Xavier Standaert: Masking Proofs Are Tight and How to Exploit it in Security Evaluations. EUROCRYPT (2) 2018: 385-412. **[ISW03]** Yuval Ishai, Amit Sahai, David A. Wagner: Private Circuits: Securing Hardware against Probing Attacks. CRYPTO 2003: 463-481.

Bibliography D.4

[JS17] Anthony Journault, François-Xavier Standaert: Very High Order Masking: Efficient Implementation and Security Evaluation. CHES 2017: 623-643. **[Lo+14]** Jake Longo, Daniel P. Martin, Elisabeth Oswald, Daniel Page, Martijn Stam, Michael Tunstall: Simulatable Leakage: Analysis, Pitfalls, and New Constructions. ASIACRYPT (1) 2014: 223-242. **[MR04]** Silvio Micali, Leonid Reyzin: Physically Observable Cryptography (Extended Abstract). TCC 2004: 278-296. **[MPG05]** Stefan Mangard, Thomas Popp, Berndt M. Gammel: Side-Channel Leakage of Masked CMOS Gates. CT-RSA 2005: 351-365. **[NRS11]** Svetla Nikova, Vincent Rijmen, Martin Schläffer: Secure Hardware Implementation of Nonlinear Functions in the Presence of Glitches. J. Cryptology 24(2): 292-321 (2011). **[NS09]** Moni Naor, Gil Segev: Public-Key Cryptosystems Resilient to Key Leakage. CRYPTO 2009: 18-35. **[PR13]** Emmanuel Prouff, Matthieu Rivain: Masking against Side-Channel Attacks: A Formal Security Proof. EUROCRYPT 2013: 142-159. **[RS06]** Phillip Rogaway, Thomas Shrimpton: A Provable-Security Treatment of the Key-Wrap Problem. EUROCRYPT 2006: 373-390. **[R+15]** Oscar Reparaz, Begül Bilgin, Svetla Nikova, Benedikt Gierlichs, Ingrid Verbauwhede: Consolidating Masking Schemes. CRYPTO (1) 2015: 764-783. **[SPY13]** François-Xavier Standaert, Olivier Pereira, Yu Yu: Leakage-Resilient Symmetric Cryptography under Empirically Verifiable Assumptions. CRYPTO (1) 2013: 335-352. **[YSPY10]** Yu Yu, François-Xavier Standaert, Olivier Pereira, Moti Yung: Practical leakage-resilient pseudorandom generators. ACM Conference on Computer and Communications Security 2010: 141-151.