CryptoExperts

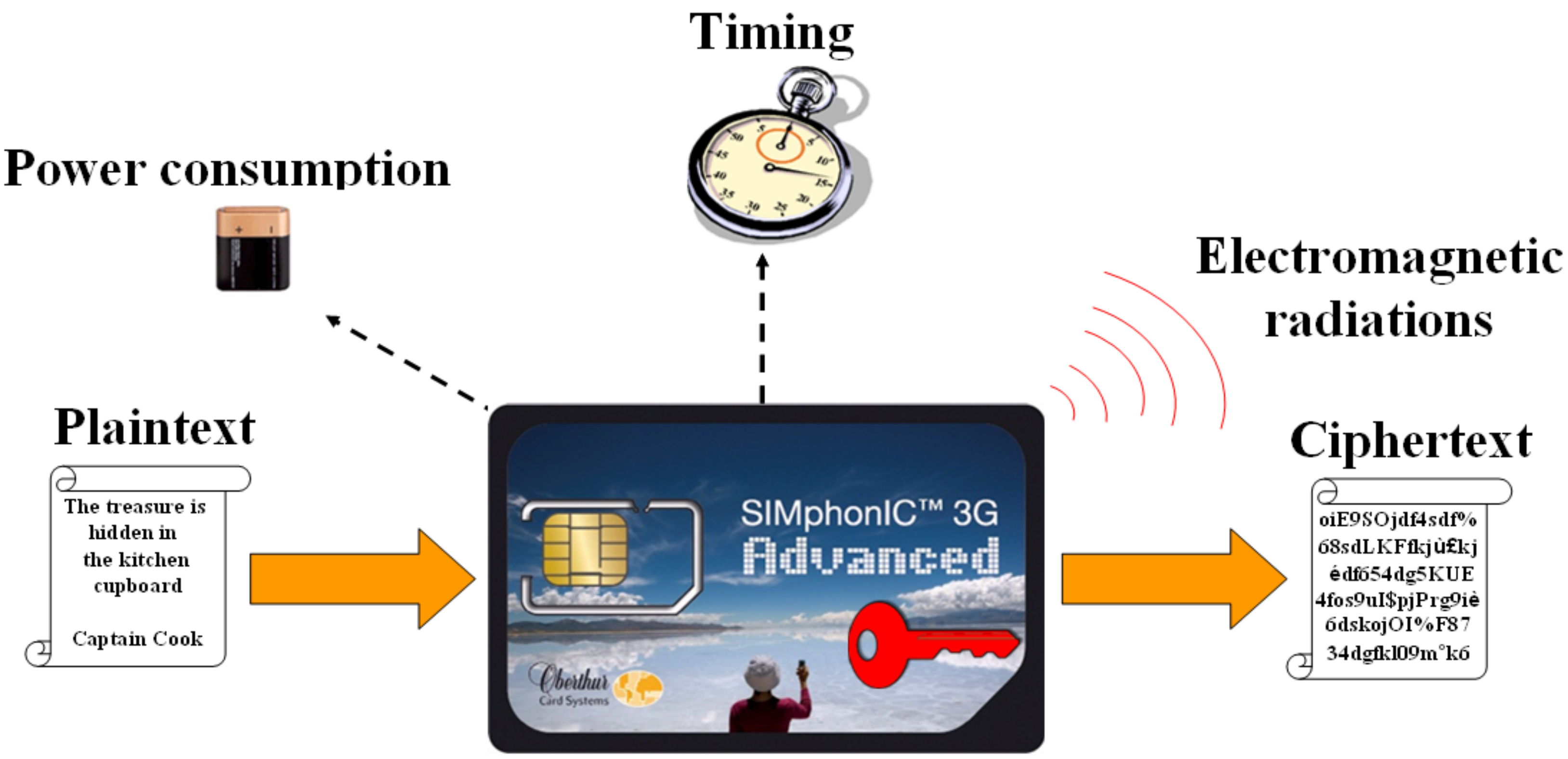
WE INNOVATE TO SECURE YOUR BUSINESS

Secure computation in the presence of noisy leakage

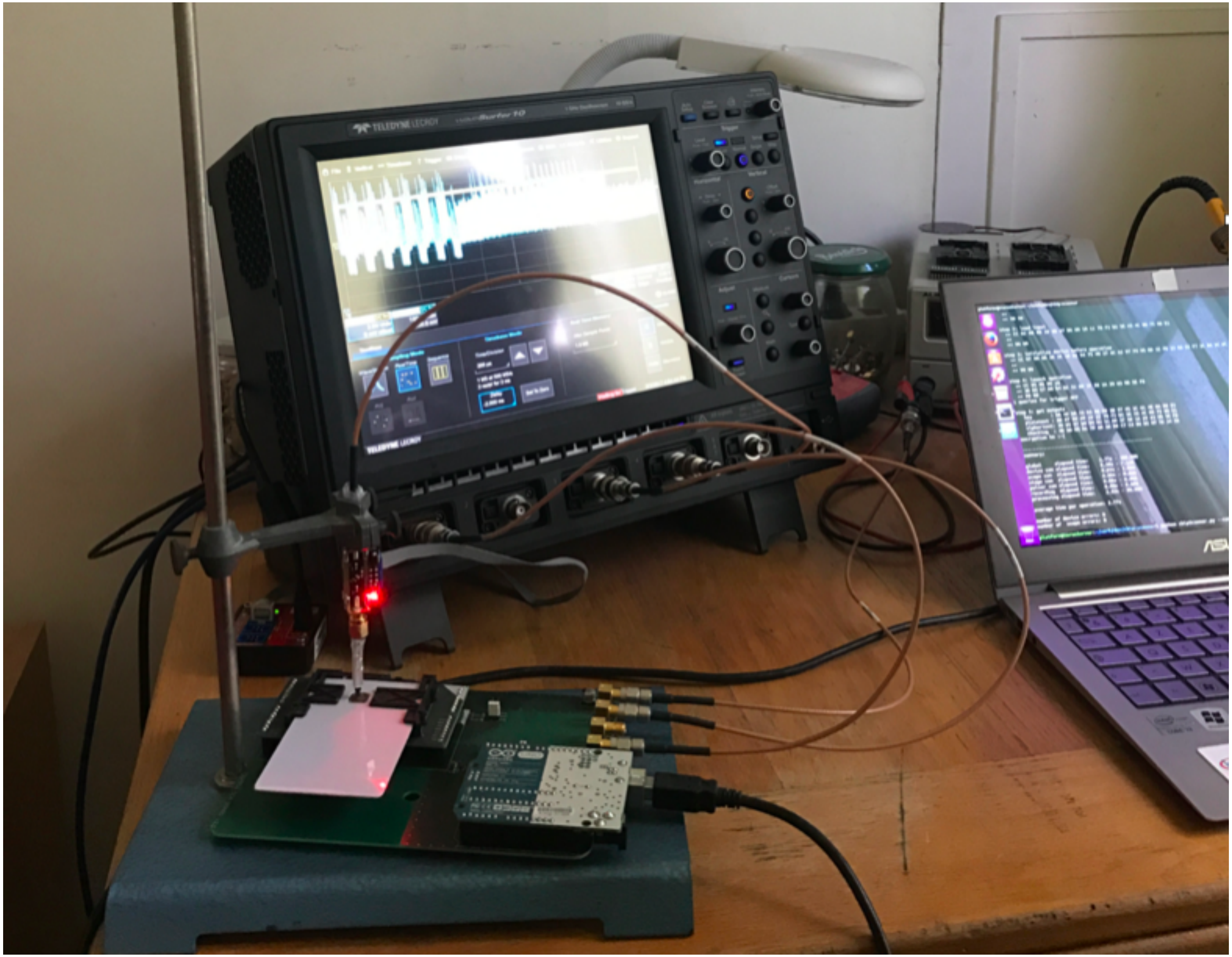
matthieu.rivain@cryptoexperts.com

Journees C2, Aussois, 10 Oct. 2018

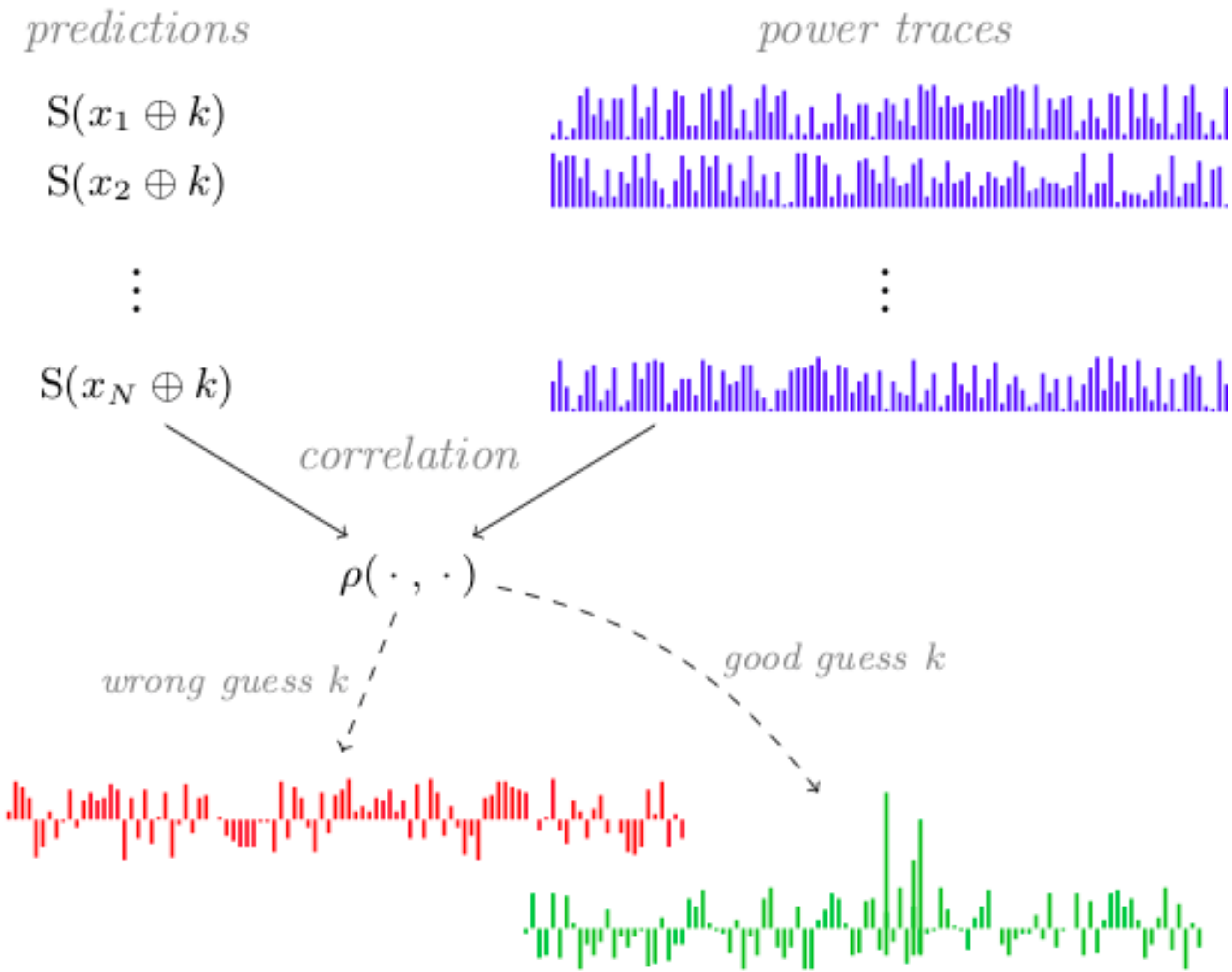
Side-channel attacks



Side-channel attacks

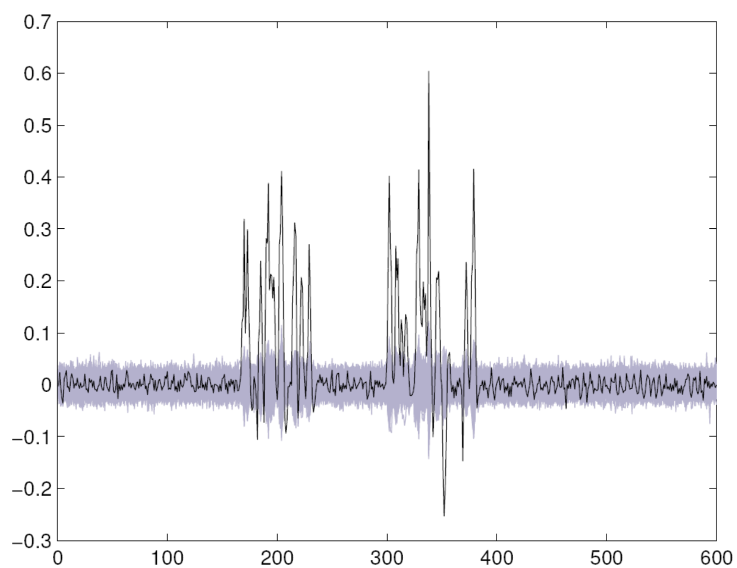


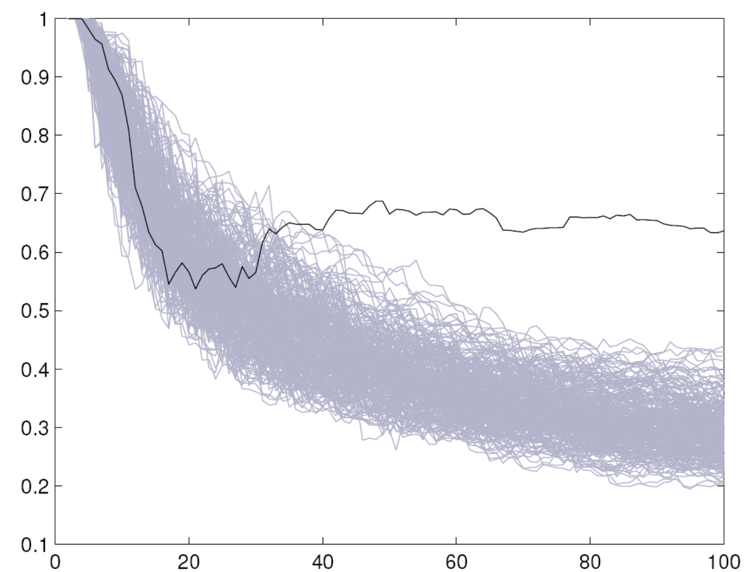
Differential Power Analysis (DPA)



Differential Power Analysis (DPA)

Practical attack on a smart-card AES implementation





Right guess distinguishable after $\sim$ 50 traces

Masking

- Mask intermediate variables with randomness

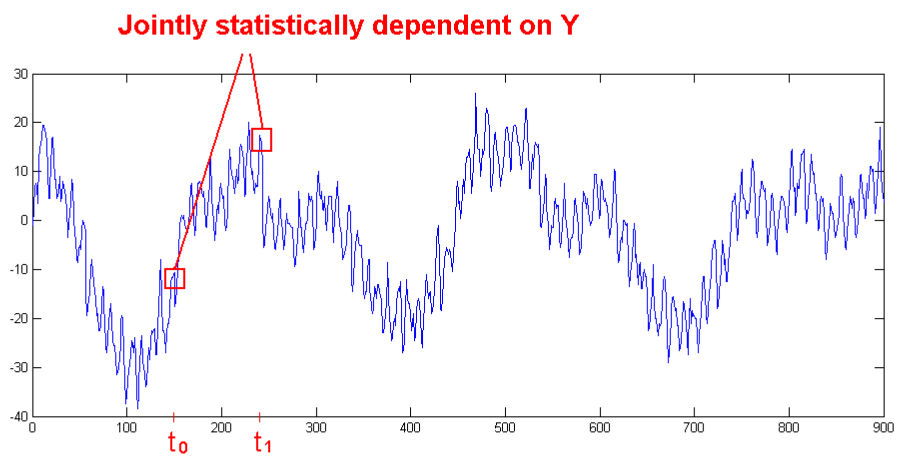
$$Y \quad \rightarrow \quad Y \oplus R$$

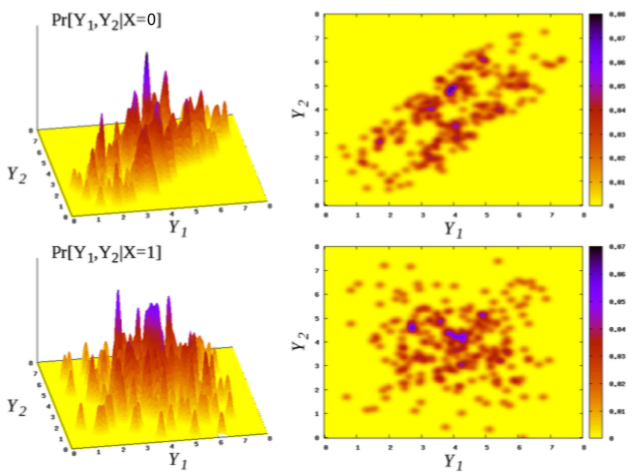
- Perform computation w/o exposing Y
- *e.g.* linear function

$$\ell(Y \oplus R) = \ell(Y) \oplus \ell(R)$$

$\Rightarrow$ no more correlation peaks

Advanced side-channel attacks





- Higher-order attacks
- Template / multivariate attacks
- New trend: machine learning 🤖

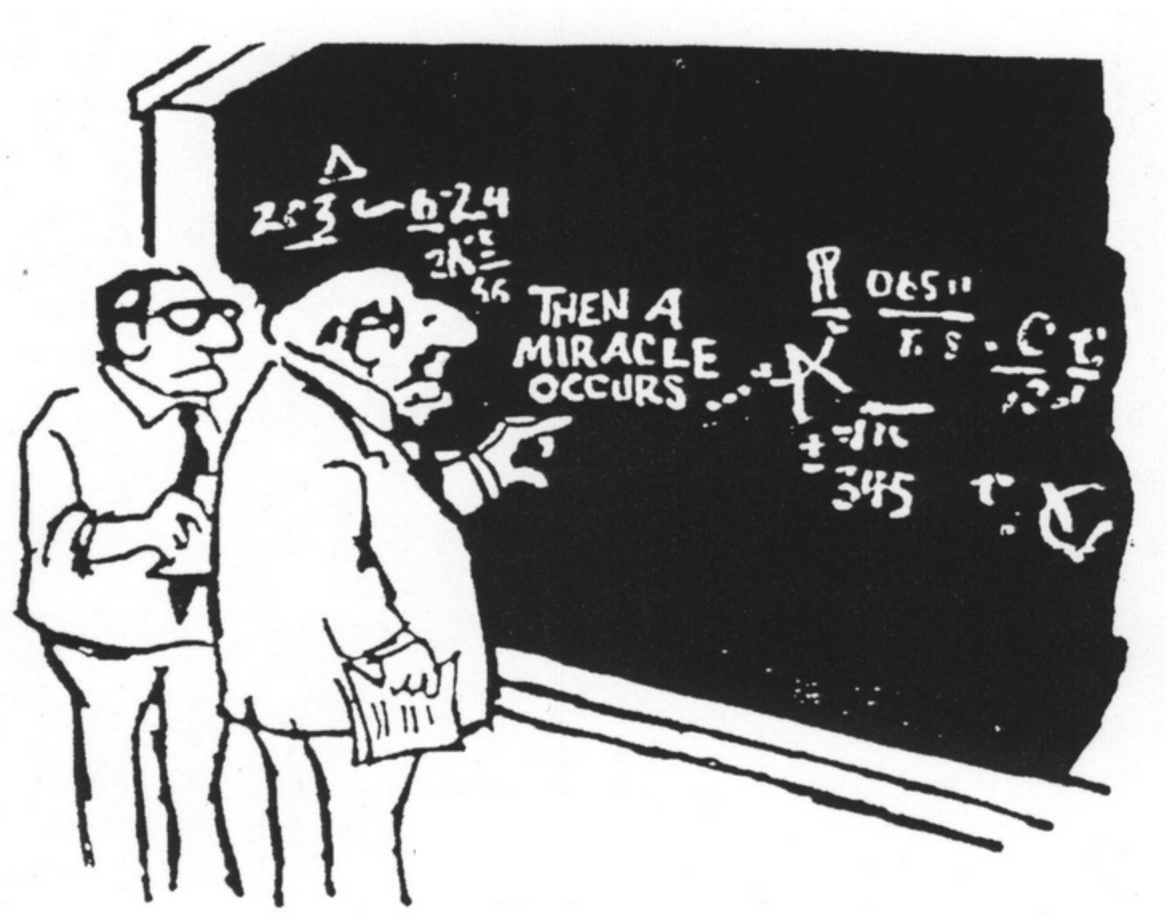
What can we do?

- Increase the number of masks $\Rightarrow$ higher-order masking
- Combine different countermeasures (e.g. noise and masking)
- Try our best attacks in practice (approach followed by the industry)

But what if the attacker has better
equipment / skills / computational power ?

🤔

We want a security proof!

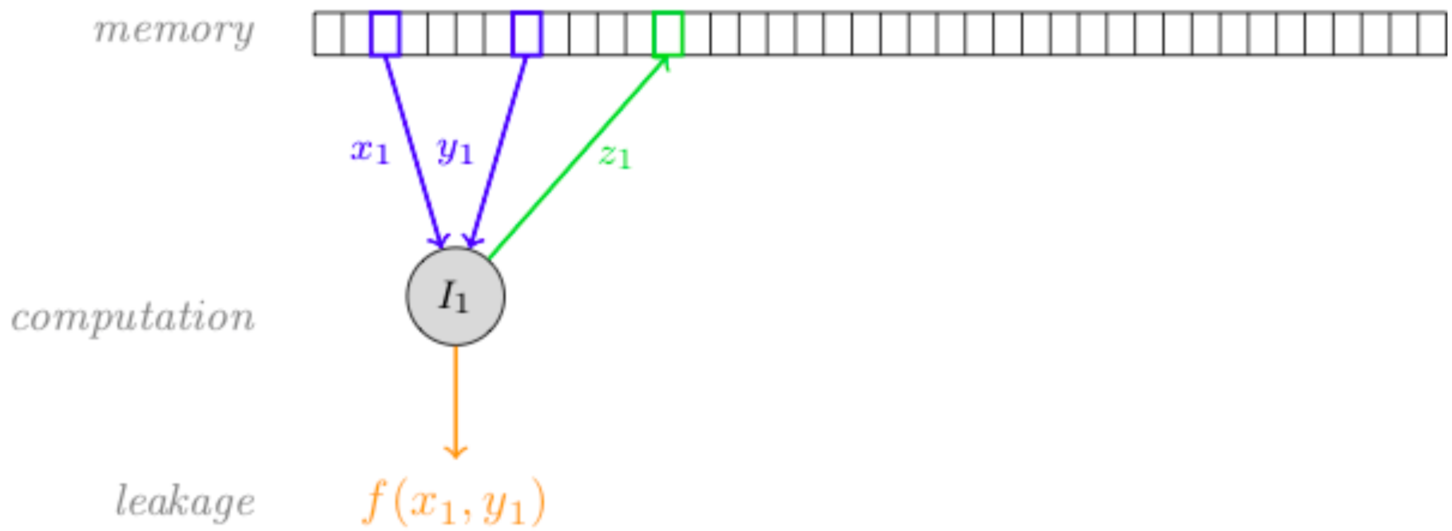


Modeling the side-channel leakage

> Micali, Reyzin. *Physically Observable Cryptography* (TCC 2004)

Key assumption: **only computation leaks information** [MR04]

Computation divided in sub-computations $I_1, I_2, \dots, I_s$

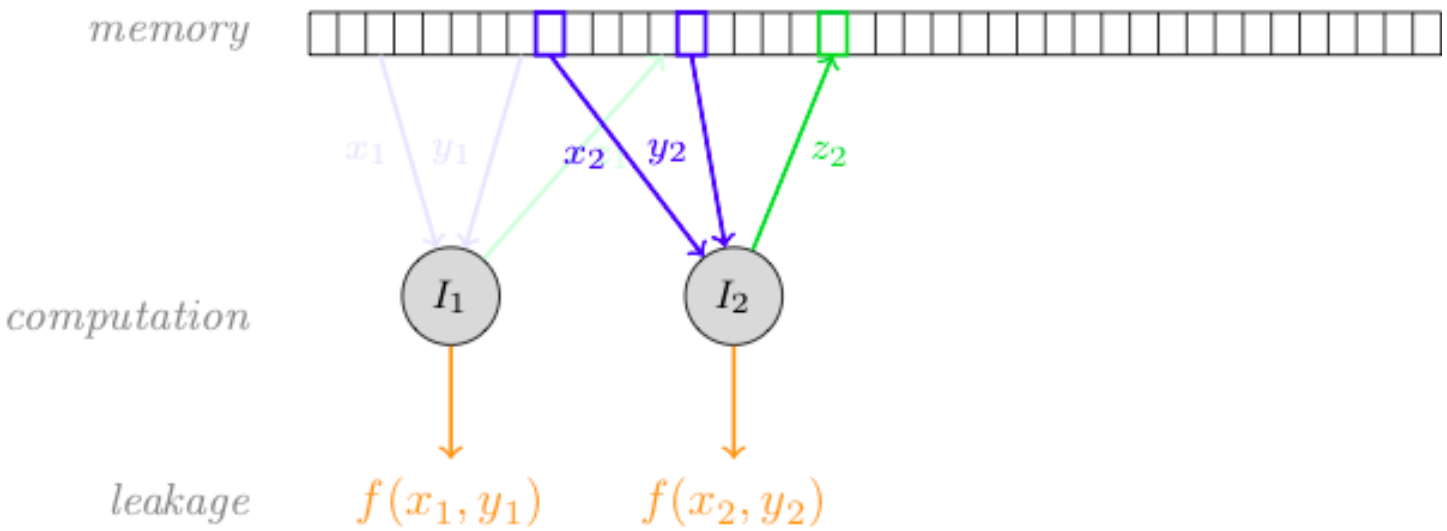


Modeling the side-channel leakage

> Micali, Reyzin. *Physically Observable Cryptography* (TCC 2004)

Key assumption: **only computation leaks information** [MR04]

Computation divided in sub-computations $I_1, I_2, \ldots, I_s$

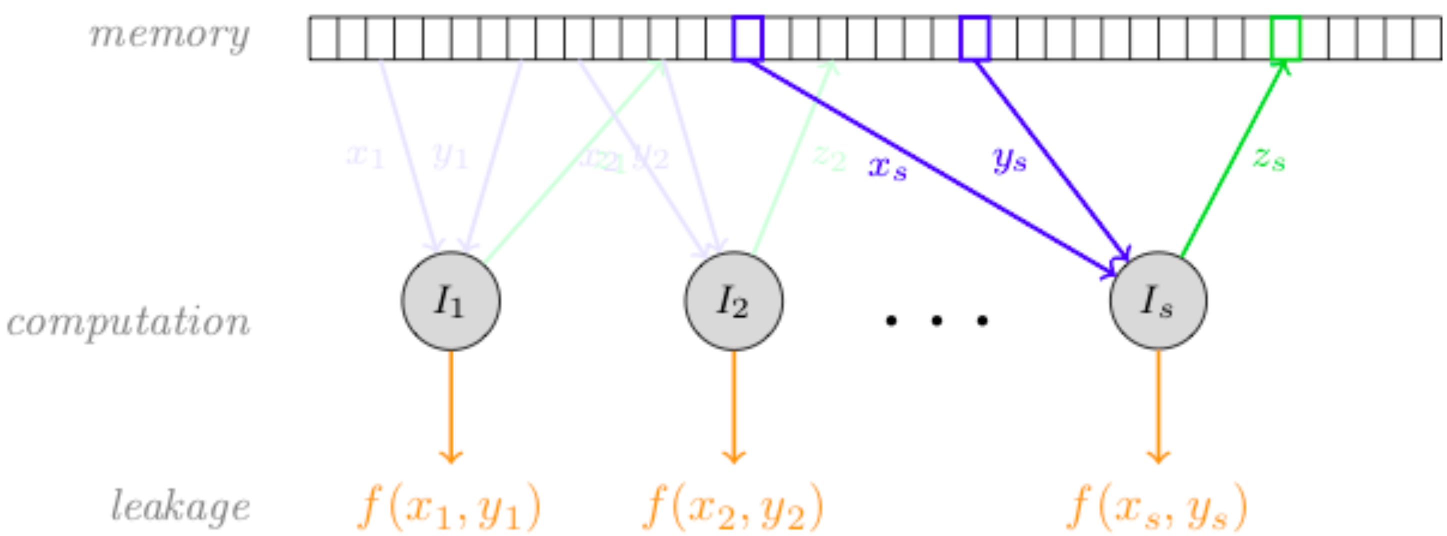


Modeling the side-channel leakage

> Micali, Reyzin. *Physically Observable Cryptography* (TCC 2004)

Key assumption: **only computation leaks information** [MR04]

Computation divided in sub-computations $I_1, I_2, \ldots, I_s$



Modeling the side-channel leakage

- **Granularity of the computation:** I_i might be
  - a logic gate
  - a CPU instruction,
  - an arithmetic instruction (operation on a field $\mathbb{F}$),
  - a cryptographic instruction (e.g. blockcipher, hash function)
- **Leakage function:** f
  - should capture the physical reality,
  - shouldn't reveal its full input (otherwise white-box model)

Bounded-range leakage

> Dziembowski, Pietrzak. *Leakage Resilient Cryptography.* (FOCS 2008)

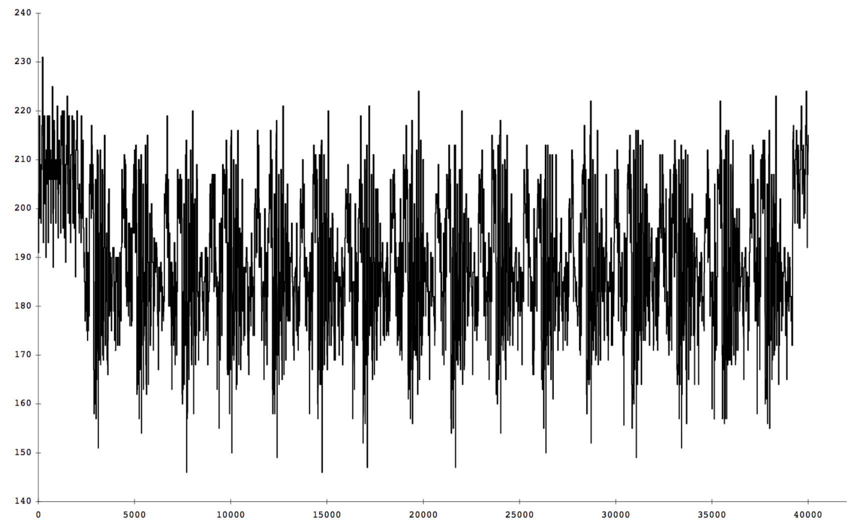
- Leakage function:

$$f : \{0,1\}^m \mapsto \{0,1\}^\lambda \qquad \text{for some } \lambda < m.$$

- Huge amount of (theoretical) **leakage resilient** constructions
  - *e.g.* generic compilers (FOCS 2012, TCC 2012)

Bounded-range leakage

- Doesn't fit the reality of power / EM leakages

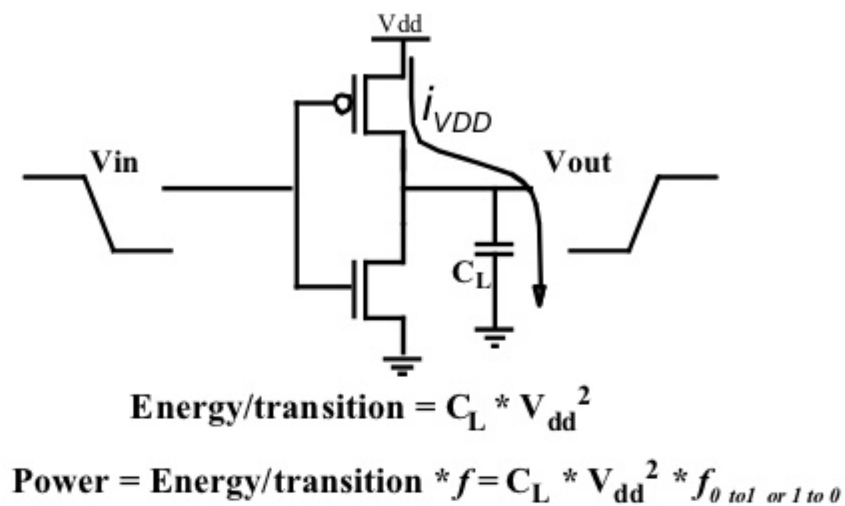


Example: power trace of a DES computation

- $|f(x)| \gg |x|$ in practice (even for lower granularity)
- **but...** $f(x)$ might not reveal full information on x

Power and electromagnetic leakage

- Dynamic power consumption of a logic gate



- Total PC $\sim$ weighted # transitions $\{0 \to 1\,,1 \to 0\}$
- EM leakage depends on the location of the probe

Power and electromagnetic leakage

- Highly dependent on the processed data
- Fairly independent on the stored data
  - *"only computation leaks"* assumption OK
- Revealed information is **noisy** due to
  - non-targeted switching activity
  - hardware security features
  - measurement noise

Noisy leakage model

> Prouff, Rivain. *Masking against Side-Channel Attacks: A Formal Security Proof* (Eurocrypt 2013)

- f is a non-deterministic function: $f(x) = f(x,\ randomness)$
- Informally: an observation $f(X)$ introduces a **bounded bias** δ in the distribution of X
- Statistical distance $\Delta\big(X; (X|f(X))\big) \leq \delta \quad \Rightarrow \quad f$ is δ-noisy
- Capture **any** noisy leakage distribution (single parameter δ)

no information $\quad 0 \leq \delta \leq 1 \quad$ *full information*

How to securely compute in this model?

A first step

> Chari *et al. Towards Sound Approaches to Counteract Power-Analysis Attacks* (Crypto 1999)

- **Boolean masking:** x is randomly shared as

$$(x_1, x_2, \ldots, x_n) \quad \text{s.t.} \quad x = x_1 \oplus x_2 \oplus \cdots \oplus x_n$$

- The adversary gets (for every i)

$$\ell_i \sim x_i + \mathcal{N}(\mu, \sigma)$$

- Information on $x \leq \Theta((\frac{1}{\sigma})^{n/2})$ $\Rightarrow$ negligible as n grows

A first step

- Generalisation to noisy leakage:
  - $\ell_i = f(x_i)$ where f is δ-noisy
  - Info on $x \leq \Theta(\delta^{\,n})$ $\Rightarrow$ negligible as n grows
- Limitation: static leakage of the shares
- Question:

How to securely compute on the shares

in the presence of noisy leakage?

🤔

ISW scheme

> Ishai-Sahai-Wagner. *Private Circuits: Securing Hardware against Probing Attacks* (Crypto 2003)

- Addition (XOR) gates $\Rightarrow$ easy
- Multiplication (AND) gates
  - from $(a_1, a_2, \ldots, a_n)$ and $(b_1, b_2, \ldots, b_n)$
  - compute $(c_1, c_2, \ldots, c_n)$ s.t.

$$\bigoplus_i c_i = a \cdot b = \left(\bigoplus_i a_i\right)\left(\bigoplus_i b_i\right) = \bigoplus_{i,j} a_i b_j$$

- Principle: split the sum $\bigoplus_{i,j} a_i b_j$ into n new shares (with additional fresh randomness)

ISW scheme

- Multiplication gadget for $n = 3$

$$\begin{pmatrix} a_1b_1 & a_1b_2 & a_1b_3 \\ a_2b_1 & a_2b_2 & a_2b_3 \\ a_3b_1 & a_3b_2 & a_3b_3 \end{pmatrix} \mapsto \begin{pmatrix} a_1b_1 & a_1b_2 \oplus a_2b_1 & a_1b_3 \oplus a_3b_1 \\ 0 & a_2b_2 & a_2b_3 \oplus a_3b_2 \\ 0 & 0 & a_3b_3 \end{pmatrix}$$

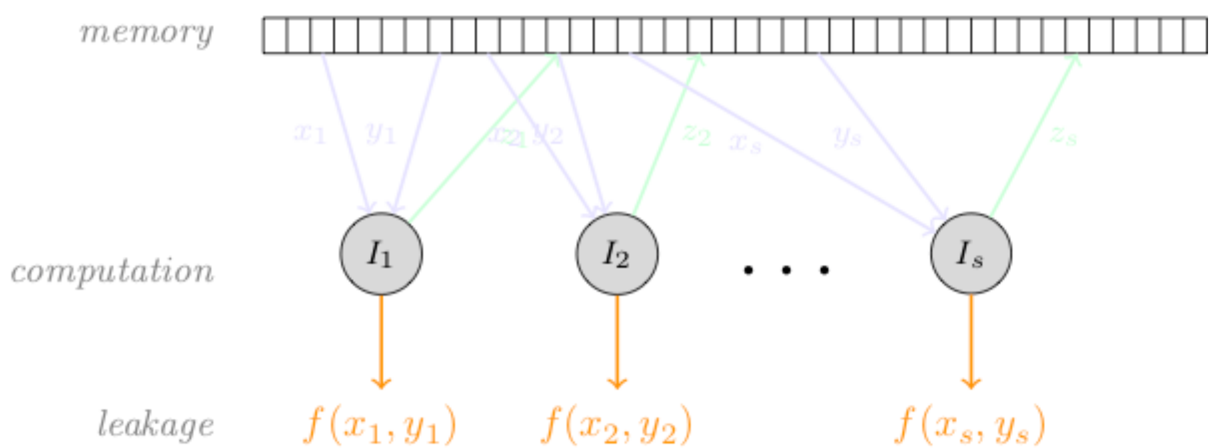
$$\mapsto$$

$$\begin{pmatrix} a_1b_1 & (a_1b_2 \oplus r_{1,2}) \oplus a_2b_1 & (a_1b_3 \oplus r_{1,3}) \oplus a_3b_1 \\ r_{1,2} & a_2b_2 & (a_2b_3 \oplus r_{2,3}) \oplus a_3b_2 \\ r_{1,3} & r_{2,3} & a_3b_3 \end{pmatrix}$$

New share $c_i \equiv$ sum of i-th row

ISW scheme

- Complexity $O(n^2)$ operations / original gate
- Probing security:



  - the adversary can probe t instructions I_i
  - $f(x_i, y_i) = (x_i, y_i)$ for t instructions
  - $f(x_i, y_i) = \perp$ for other instructions
- ISW scheme is t-probing secure for $t < n/2$

Towards noisy leakage security

> Prouff, Rivain. *Masking against Side-Channel Attacks: A Formal Security Proof* (Eurocrypt 2013)

- variant of ISW secure against δ-noisy leakage
- strong assumption: leak-free refreshing procedure

> Duc-Dziembowski-Faust. *Unifying Leakage Models: from Probing Attacks to Noisy Leakage* (Eurocrypt 2014)

- probing security $\Rightarrow$ noisy-leakage security
- noisy leakage security of ISW scheme (w/o leak-free procedure)

Towards noisy leakage security

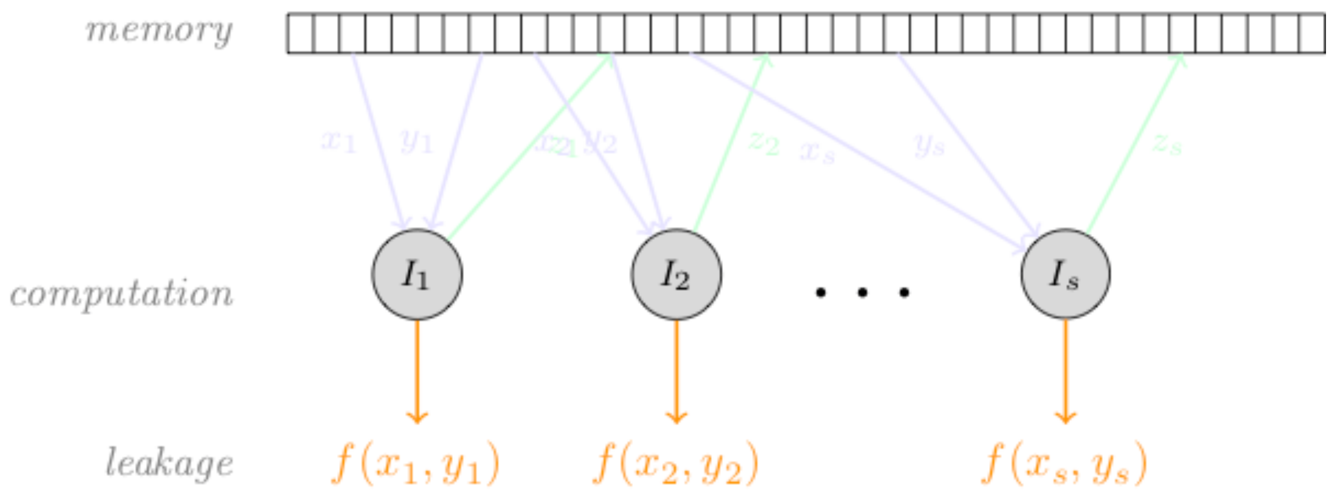
> Prouff, Rivain. *Masking against Side-Channel Attacks: A Formal Security Proof* (Eurocrypt 2013)

- variant of ISW secure against δ-noisy leakage
- strong assumption: leak-free refreshing procedure

> Duc-Dziembowski-Faust. *Unifying Leakage Models: from Probing Attacks to Noisy Leakage* (Eurocrypt 2014)

- probing security $\Rightarrow$ noisy-leakage security
- noisy leakage security of ISW scheme (w/o leak-free procedure)

The random probing model



- The leakage function f is such that

$$f(x_i, y_i) = \begin{cases} (x_i, y_i) & \text{with probability } \varepsilon \\ \perp & \text{with probability } 1 - \varepsilon \end{cases}$$

The random probing model

- We have s instructions, each leaking with probability ε
- # leaking instruction $t \sim \mathcal{B}(s, \varepsilon)$
- On average $t = s \cdot \varepsilon$
- Chernoff bound: $\mathrm{P}(t \geq 2\, s \cdot \varepsilon) \leq \exp(-s \cdot \varepsilon/3)$

  $\Rightarrow$ negligible as $s \cdot \varepsilon$ grows

- **t-probing security $\Rightarrow$ ε-random probing security**

  with $\varepsilon = O(t/s)$

Random probing $\Rightarrow$ noisy model

- **Key lemma (DDF, Eurocrypt 2014)**

  Every δ-noisy function f can be written as

  $$f(\cdot) = f' \circ \varphi(\cdot)$$

  where φ is an ε-random probing function with $\varepsilon = \Theta(\delta)$

- ε-random probing security

  $\Rightarrow$ δ-noisy leakage security with $\delta = \Theta(\varepsilon)$

Probing security $\Rightarrow$ noisy leakage security

- Wrapping up

  t-probing security

  $\Rightarrow$ ε-random probing security with $\varepsilon = O(t/s)$

  $\Rightarrow$ δ-noisy leakage security with $\delta = O(\varepsilon) = O(t/s)$

- For ISW
  - n shares
  - $s = O(n^2)$ instructions
  - $t = O(n)$ probes tolerated

  $\Rightarrow \quad \delta = O(1/n)$ -noisy leakage tolerated

Limitations

- The leakage rate is $O(1/n) \quad \equiv \quad$ the noise is $O(n)$
- Intuition: each share is manipulated n times

$$\begin{pmatrix} a_1b_1 & a_1b_2 & \cdots & a_1b_n \\ a_2b_1 & a_2b_2 & \cdots & a_2b_n \\ \vdots & \vdots & \ddots & \vdots \\ a_nb_1 & a_nb_2 & \cdots & a_nb_n \end{pmatrix}$$

- Can we tolerate a constant amount of leakage rate?
- Other question: Can we do better than $O(n^2)$ in complexity?

Towards a constant leakage rate

> Andrychowicz, Dziembowski, Faust. *Circuit Compilers with $O(1/\log(n))$ Leakage Rate.* (Eurocrypt 2016)

- based on Ajtai (STOC 2011)
- noisy leakage security with $\delta = O(1)$
- complexity blow-up of $\tilde{O}(n^2)$

> Goudarzi, Joux, Rivain. *How to Securely Compute with Noisy Leakage in Quasilinear Complexity.* (Asiacrypt 2018)

- noisy leakage security with $\delta = \tilde{O}(1)$
- complexity blow-up of $\tilde{O}(n)$

Towards a constant leakage rate

> Andrychowicz, Dziembowski, Faust. *Circuit Compilers with $O(1/\log(n))$ Leakage Rate.* (Eurocrypt 2016)

- based on Ajtai (STOC 2011)
- noisy leakage security with $\delta = O(1)$
- complexity blow-up of $\tilde{O}(n^2)$

> Goudarzi, Joux, Rivain. *How to Securely Compute with Noisy Leakage in Quasilinear Complexity.* (Asiacrypt 2018)

- noisy leakage security with $\delta = \tilde{O}(1)$
- complexity blow-up of $\tilde{O}(n)$

Our encoding

- A variable $a \in \mathbb{F}$ is randomly encoded as

$$\mathsf{Enc}(a) = (a_0, a_1, \ldots, a_{n-1}) \quad \text{where} \quad \sum_{i=0}^{n-1} a_i \cdot \omega^i = a\ .$$

- ω is random in $\mathbb{F}$ but can be leaked to the adversary
- Encoding linearity

$$\mathsf{Enc}(a) + \mathsf{Enc}(b) = \mathsf{Enc}(a + b)$$

Multiplying encodings

- Goal: compute $\mathsf{Enc}(a \cdot b)$ from $\mathsf{Enc}(a)$ and $\mathsf{Enc}(b)$
- Consider

$$P_a(X) = \sum_{i=0}^{n-1} a_i \cdot X^i \quad \text{and} \quad P_b(X) = \sum_{i=0}^{n-1} b_i \cdot X^i$$

- By defintion $P_a(\omega) = a$ and $P_b(\omega) = b$
- Define

$$P_t(X) = P_a(X) \cdot P_b(X) = \sum_{i=0}^{2n-1} t_i \cdot X^i$$

- We have $P_t(\omega) = a \cdot b$ but $\deg(P_t) = 2n - 2 \geq n - 1$

Multiplying encodings

- Compression procedure:

$$a \cdot b = P_t(\omega) = \sum_{i=0}^{2n-1} t_i \cdot \omega^i$$

$$a \cdot b = P_c(\omega) = \sum_{i=0}^{n-1} c_i \cdot \omega^i$$

with $c_i = t_i + t_{n+i} \cdot \omega^n$

- Enc$(a \cdot b) = (c_0, c_1, \ldots, c_{n-1})$

Number Theoretic Transform (NTT)

- Compute $P_t(X) = P_a(X) \cdot P_b(X)$ in $O(n \log n)$
- Evaluate P_a and P_b in $2n$ points α, β, ..., γ

$$\begin{pmatrix} \alpha^0 & \alpha^1 & \cdots & \alpha^{n-1} \\ \beta^0 & \beta^1 & \cdots & \beta^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ \gamma^0 & \gamma^1 & \cdots & \gamma^{n-1} \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix} = \begin{pmatrix} P_a(\alpha) \\ P_a(\beta) \\ \vdots \\ P_a(\gamma) \end{pmatrix}$$

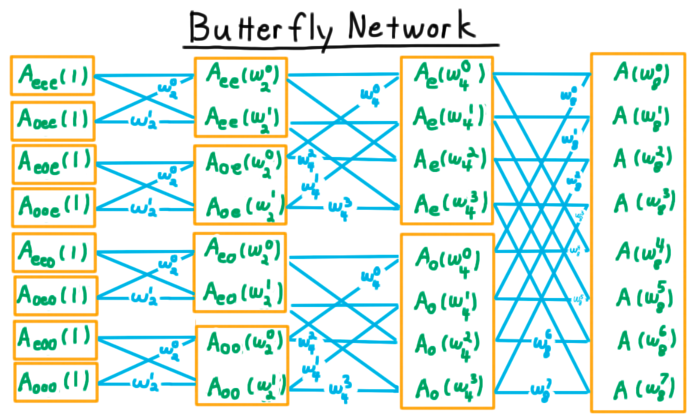
- Get the corresponding evaluations

$$P_t(\alpha) = P_a(\alpha) \cdot P_b(\alpha) \quad \ldots$$

- Integrate the coefficients of P_t from the $2n$ evaluations (multiplication by the inverse matrix)

Number Theoretic Transform (NTT)

- Takes points α, β, ..., γ as the $2n$-th roots of unity
- Each matrix multiplication computed as a **butterfly network**



- $(\log 2n)$ steps of n butterfly operations $\Rightarrow O(n \log n)$
- Constraint: $2n$-th roots of unity $\in \mathbb{F}$

Random probing security

- Consider the NTT:

$$\begin{pmatrix} \alpha^0 & \alpha^1 & \cdots & \alpha^{n-1} \\ \beta^0 & \beta^1 & \cdots & \beta^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ \gamma^0 & \gamma^1 & \cdots & \gamma^{n-1} \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix} = \begin{pmatrix} P_a(\alpha) \\ P_a(\beta) \\ \vdots \\ P_a(\gamma) \end{pmatrix}$$

- ε-random probing model

$\Rightarrow t \approx O(\varepsilon\, n \log n)$ intermediate variables leak

- Leakage: $M \cdot \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix}$ for some $(t \times n)$ matrix M

Random probing security

Recall: $a = (\omega^0, \omega^1, \ldots, \omega^{n-1}) \cdot (a_0, a_1, \ldots, a_{n-1})^\mathsf{T}$

Lemma 1:

$$M \cdot \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix} \text{ reveals info on } a \quad \Leftrightarrow \quad (\omega^0, \omega^1, \ldots, \omega^{n-1}) \in \langle M \rangle$$

Lemma 2:

if $t < n$: $\mathrm{P}\left[(\omega^0, \omega^1, \ldots, \omega^{n-1}) \in \langle M \rangle\right] \;\leq\; n/|\mathbb{F}|$

Conditions:

- $t < n \qquad \Leftarrow \quad$ Chernoff with $\varepsilon = O(1/\log n)$
- $n/|\mathbb{F}| \leq 2^{-\lambda} \quad \Leftarrow \quad |p| = \log n + \lambda$

Composition

- Similar proof for the full multiplication (simpler for addition)
- Secure program composed of gadgets
  - $\mathsf{Enc}(a) + \mathsf{Enc}(b) \mapsto \mathsf{Enc}(a+b)$
  - $\mathsf{NTT\text{-}Mult} : (\mathsf{Enc}(a), \mathsf{Enc}(b)) \mapsto \mathsf{Enc}(a \cdot b)$
- Encodings **refreshed** after each gadget

$$\mathsf{Enc}(a) + \mathsf{Enc}(0) \mapsto \mathsf{Enc}(a)$$

- Sample fresh encoding of 0:

$$\mathsf{NTT\text{-}Mult} : (\text{fixed } \mathsf{Enc}(0),\ \text{random } n\text{-tuple}) \mapsto \mathsf{Enc}(0)$$

Composition

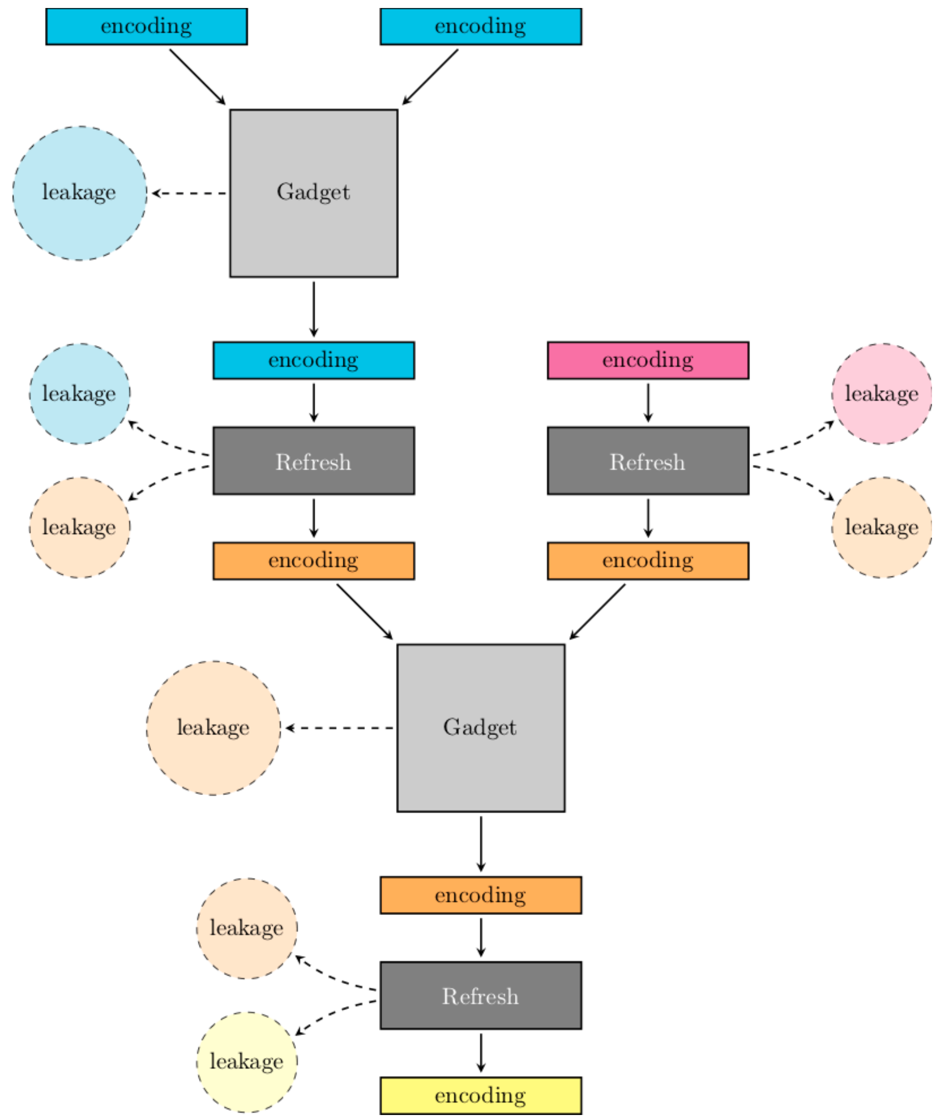
- Key properties:
  - uniformity
  - I/O separability

- DDF14 reduction:

  ε-random probing security

  $\Rightarrow \delta$-noisy leakage security

  with $\delta = O(\varepsilon) = O(1/\log n)$



Conclusion

- Noisy leakage model captures power and EM leakages
- A few constructions with provable security

	ISW / DFF14	ADF16	GJR18
Leakage rate	$O(1/n)$	$O(1)$	$\tilde{O}(1)$
Complexity blow-up	$O(n^2)$	$\tilde{O}(n^2)$	$\tilde{O}(n)$

Open questions

- What kind of δ do we get in practice for common hardware?
- Efficient implementations of ADF16 / GJR18?
- Can we get (quasi)linear complexity with smaller $\mathbb{F}$?
  With $\mathbb{F} = \mathrm{GF}(2^m)$? (*e.g.* to protect AES implementations)

Random probing security (more detail)

Recall: $a = (\omega^0, \omega^1, \ldots, \omega^{n-1}) \cdot (a_0, a_1, \ldots, a_{n-1})$

Lemma 1:

$$M \cdot \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix} \text{ reveals info on } a \quad \Leftrightarrow \quad (\omega^0, \omega^1, \ldots, \omega^{n-1}) \in \langle M \rangle$$

Proof sketch:

- W.l.g. M is full-rank
- $(\omega^0, \omega^1, \ldots, \omega^{n-1}) \notin \langle M \rangle \Rightarrow M \cdot (a_0, a_1, \ldots, a_{n-1})^{\mathsf{T}}$ uniform

Random probing security (more detail)

Lemma 2:

if $t < n$: $\mathrm{P}\left[(\omega^0, \omega^1, \ldots, \omega^{n-1}) \in \langle M \rangle\right] \leq n/|\mathbb{F}|$

Proof sketch:

- The set $\{\alpha \mid (\alpha^0, \alpha^1, \ldots, \alpha^{n-1}) \in \langle M \rangle\}$ has cardinality $< n$
- ω lies in this set with proba $< n/|\mathbb{F}|$

Conditions:

- $t < n \quad \Leftarrow \quad$ Chernoff with $\varepsilon = O(1/\log n)$
- $n/|\mathbb{F}| \leq 2^{-\lambda} \quad \Leftarrow \quad |p| = \log n + \lambda$