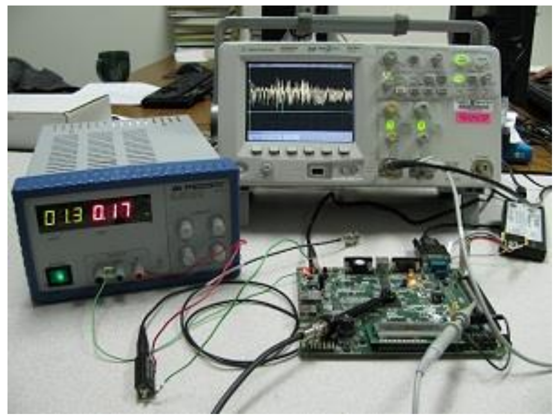
Leveraging Inexact Computing in Post-Quantum Cryptography



Davide Bellizia, ***François-Xavier Standaert***

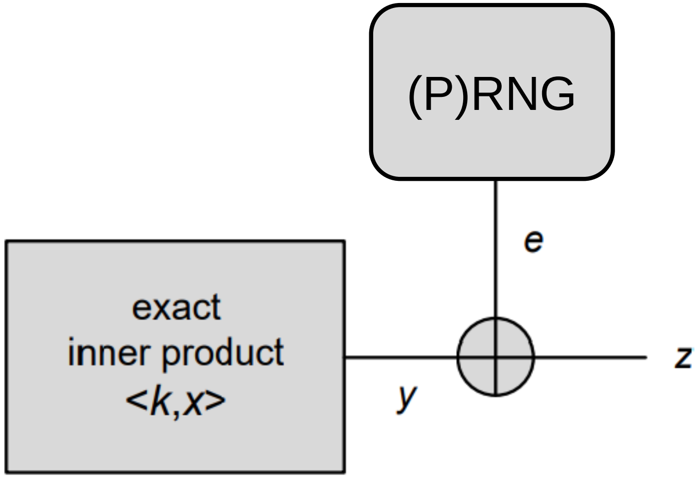
UCLouvain (Belgium)

DFT 2021, Virtual

- $D_k^\varepsilon = \{\ (x, \langle x, k \rangle \oplus e) :\ x \leftarrow \{0,1\}^n;\ e \leftarrow \mathrm{Ber}_\varepsilon\}$
- LPN problem: recover k thanks to samples from D_k^ε
- Assumed to be hard, quite versatile problem
- Extension in $\mathbb{Z}_q$ (LWE) popular for PQ crypto

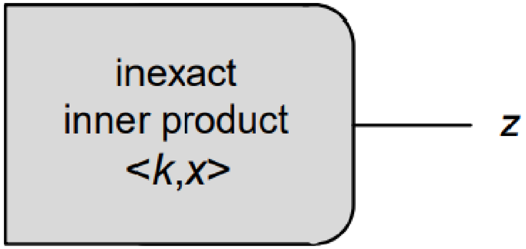
- $D_k^\varepsilon = \{\,(x, \langle x, k\rangle \oplus e) :\ x \leftarrow \{0,1\}^n;\ e \leftarrow \mathrm{Ber}_\varepsilon\}$
- LPN problem: recover k thanks to samples from D_k^ε
- Assumed to be hard, quite versatile problem
- Extension in $\mathbb{Z}_q$ (LWE) popular for PQ crypto

- Standard implementation with a (P)RNG



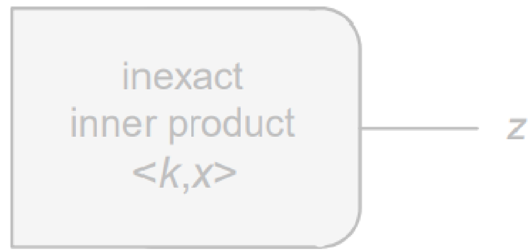
- May be expensive
  - E.g., require a block cipher
- Especially against leakage
  - Single probe on e makes LPN easy to solve

- Physical Inner Product ($\tilde{\varepsilon}$-PIP) ≈ device that directly outputs $\langle x, k \rangle$ with error probability $\tilde{\varepsilon}$



  - E.g., thanks to frequency/voltage overscaling, jitter, …

- Physical Inner Product ($\tilde{\varepsilon}$-PIP) ≈ device that directly outputs $\langle x, k\rangle$ with error probability $\tilde{\varepsilon}$



  - E.g., thanks to frequency/voltage overscaling, jitter, …

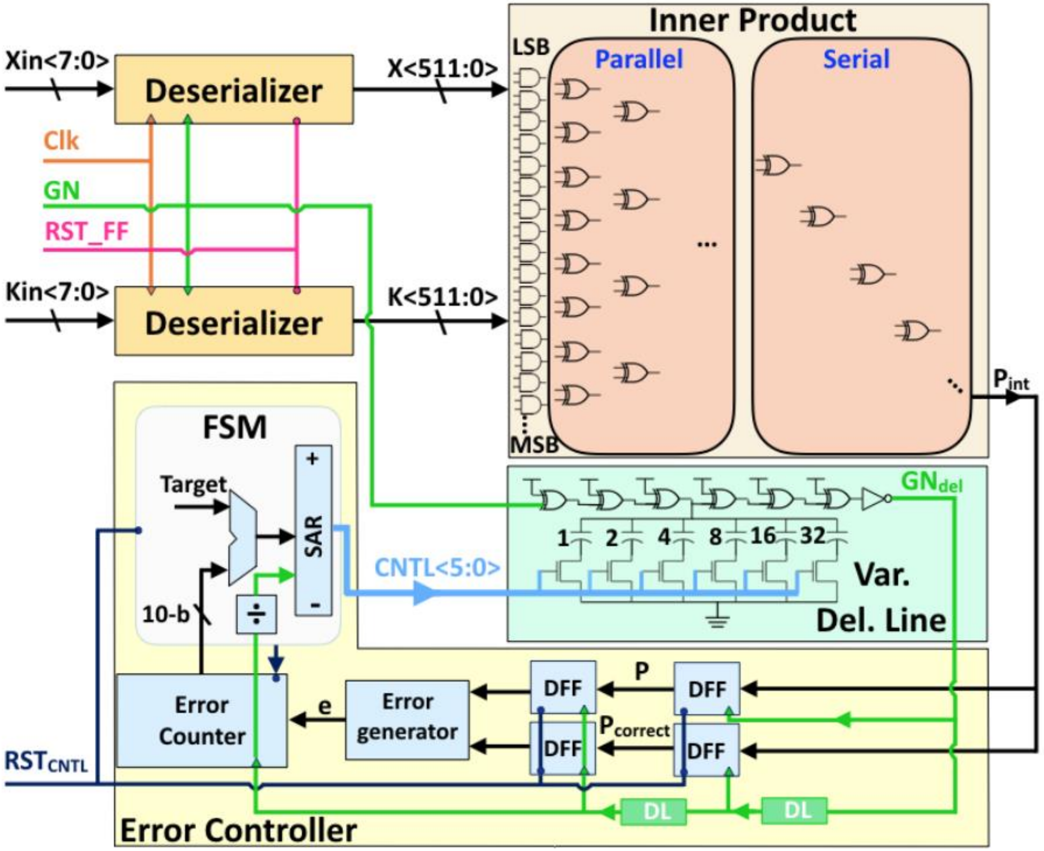
\+ Conceptually appealing (don't explicitly compute e)

\+ May lead to performance gains (e.g., in energy)

– Physical assumption (rather than mathematical one)

⇒ *Can the error probability be controlled accurately?*

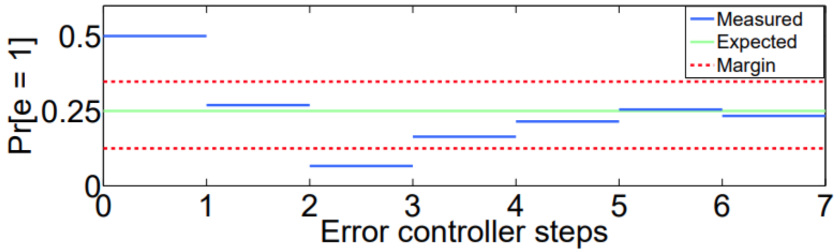
⇒ *Is the physical distribution close enough to Bernoulli?*



- Parallel then serial inner product architecture
  - Glitch-induced deterministic errors in serial part
  - Error control harder when serial part depth ↗

- Example: 512-bit LPPN co-processor
  - 64-bit parallel × 8-bit serial architecture

Step 1: error calibration

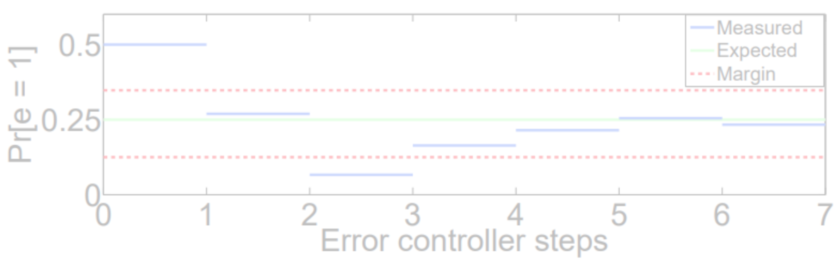


- Can be made quite accurate!
  - 6 bits of control
  - 1024 queries per bit
  - E.g., Target $\varepsilon = 0.25$

Feasibility & defaults #1: ASIC prototype

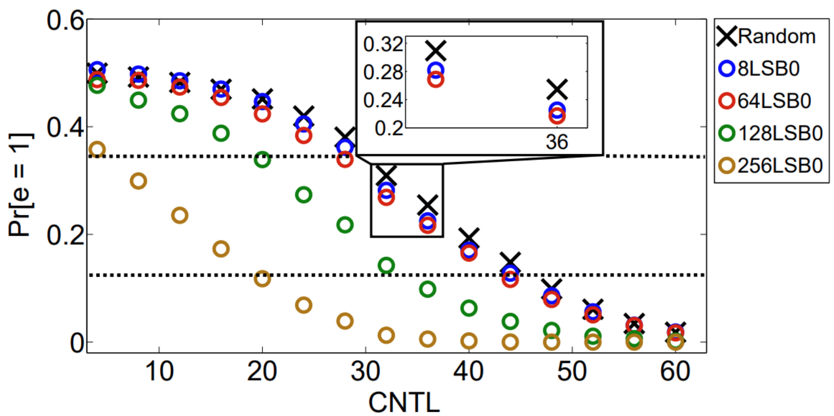
- Example: 512-bit LPPN co-processor
  - 64-bit parallel × 8-bit serial architecture

Step 1: error calibration



- Can be made quite accurate!
  - 6 bits of control
  - 1024 queries per bit
  - E.g., Target $\varepsilon = 0.25$

Step 2: samples generation



- Input-dependent Pr[error]
- E.g., by setting the bits of the serial part to all zeros / ones
- Mitigated by the parallel part

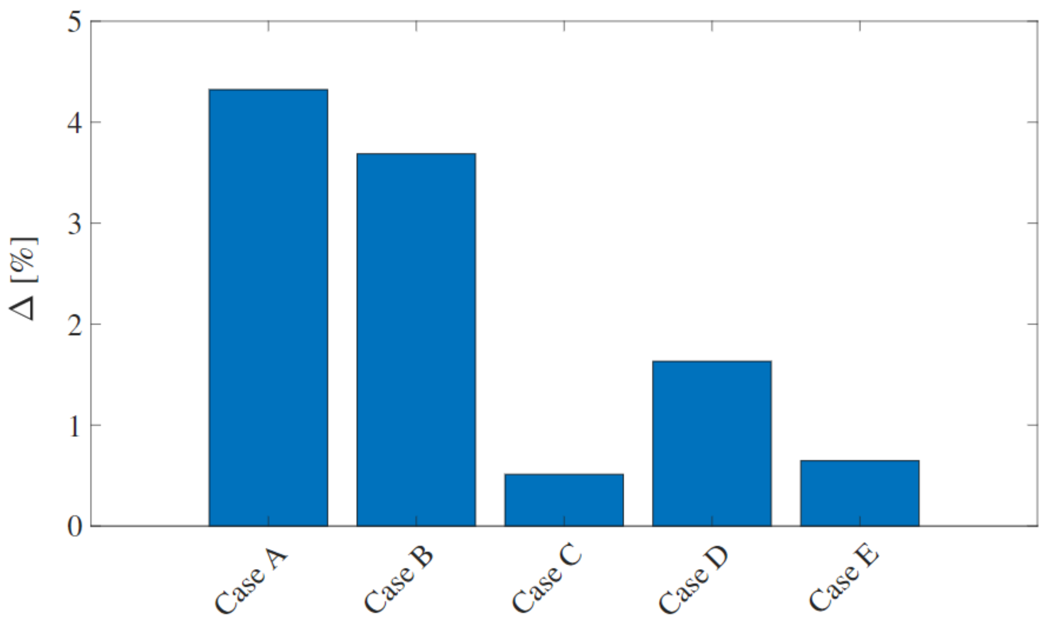
Defaults #2: output error dependencies

- Data-dependent Pr[error] extends to outputs
  - Cannot be made computationally hard to exploit
  - Cannot be completely cancelled by design

Defaults #2: output error dependencies

- Data-dependent Pr[error] extends to outputs
  - Cannot be made computationally hard to exploit
  - Cannot be completely cancelled by design

$$\Delta = \frac{\left|\widehat{\Pr}[e \leftarrow \varepsilon_0\text{–PIP}] - \widehat{\Pr}[\varepsilon_1\text{–PIP}]\right|}{2}$$



A. Parallel
B. Serial (min. size gates)
C. Parallel + jitter
D. B + power gating
E. D + bigger gates

- LPN-OD ≈ LPN with output-dependent errors

$$D_k^{\varepsilon_0,\varepsilon_1} = \left\{ \begin{matrix} (x, \langle x, k\rangle \oplus e) : \; x \leftarrow \{0,1\}^n; \\ e \leftarrow \mathrm{Ber}_{\varepsilon_0} \text{ if } \langle x, k\rangle = 0; e \leftarrow \mathrm{Ber}_{\varepsilon_1} \text{ if } \langle x, k\rangle = 1 \end{matrix} \right\}$$

Security reduction: LPN-OD ≈ LPN

- LPN-OD ≈ LPN with output-dependent errors

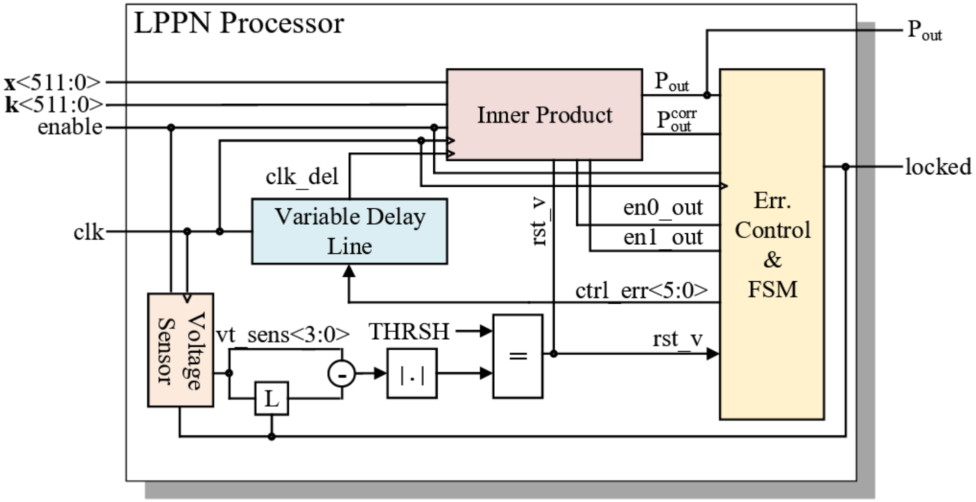
$$D_k^{\varepsilon_0,\varepsilon_1} = \left\{ \begin{array}{c} (x, \langle x,k\rangle \oplus e) : \ x \leftarrow \{0,1\}^n; \\ e \leftarrow \mathrm{Ber}_{\varepsilon_0} \text{ if } \langle x,k\rangle = 0; e \leftarrow \mathrm{Ber}_{\varepsilon_1} \text{ if } \langle x,k\rangle = 1 \end{array} \right\}$$

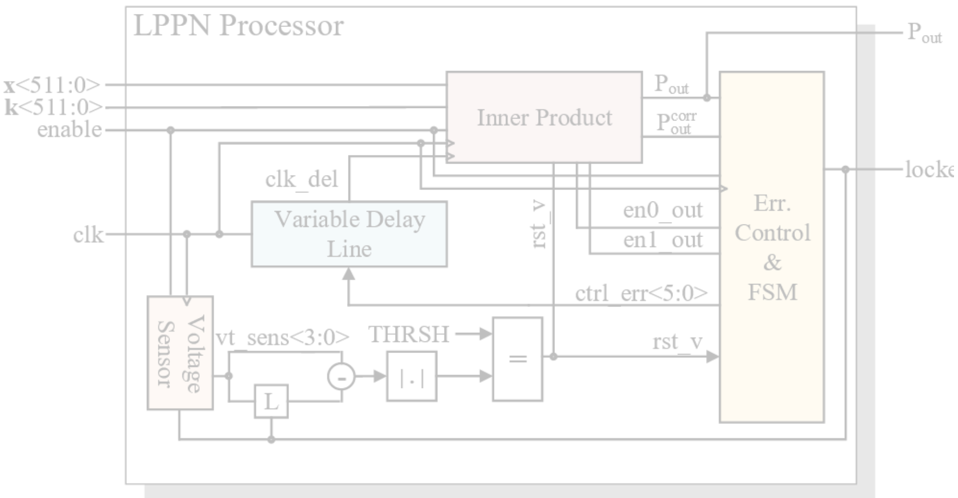
- **Theorem**: LPN-OD with $\varepsilon_0 = \varepsilon - \Delta$, $\varepsilon_1 = \varepsilon + \Delta$ is at least as hard as LPN with adapted security parameter

$$\varepsilon' = \frac{\varepsilon - \Delta}{1 - 2\Delta}$$

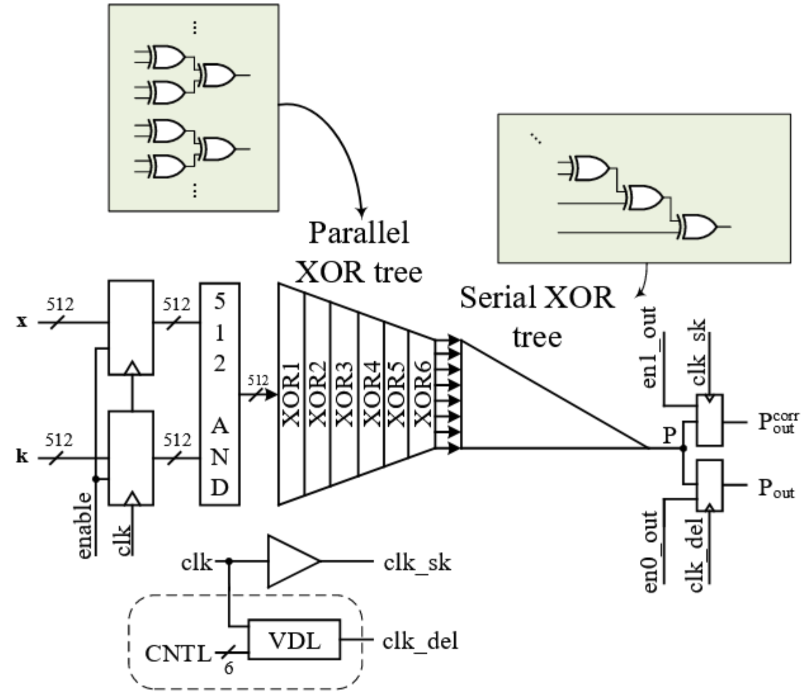
- Lower Δ (by design) ⇒ less security degradation

FPGA instance



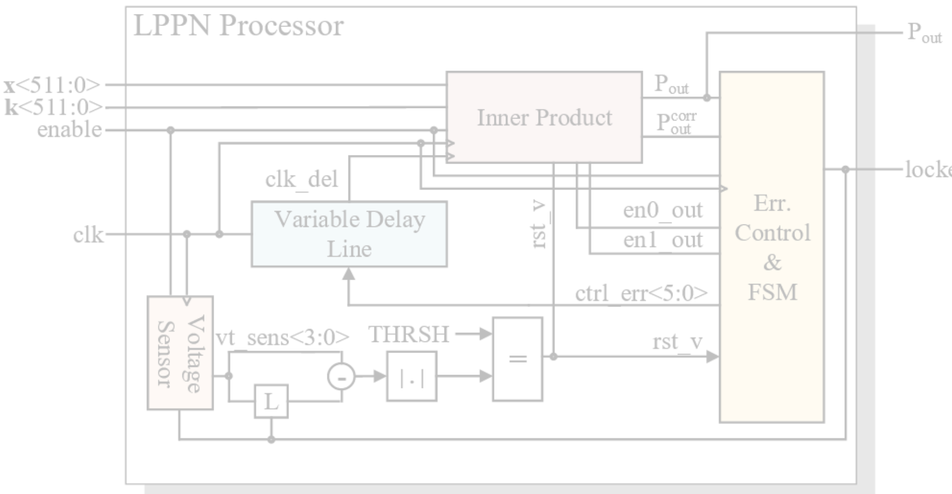


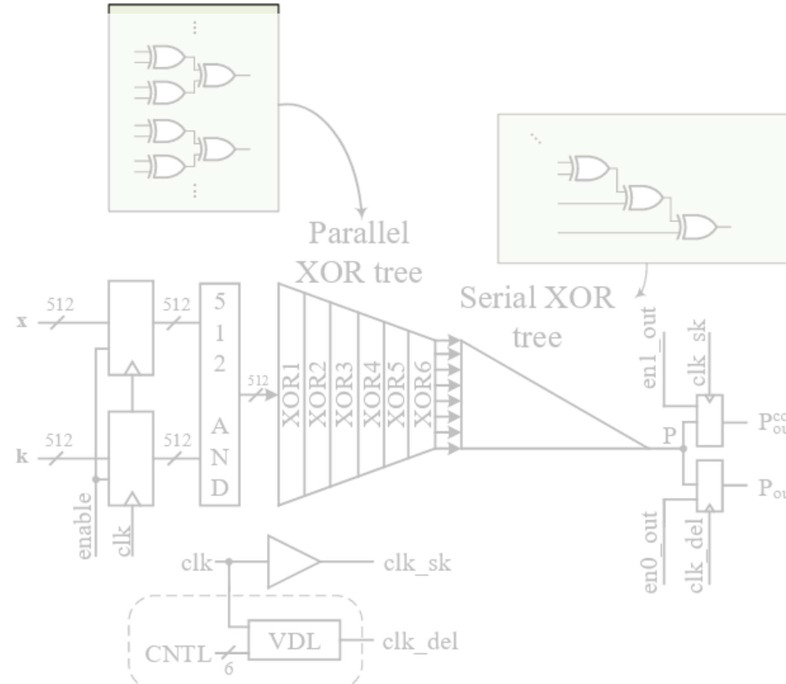
LPPN Processor
x<511:0>
k<511:0>
enable
clk
Inner Product
Pout
Pcorr out
clk_del
Variable Delay Line
rst_v
en0_out
en1_out
Err. Control & FSM
ctrl_err<5:0>
Voltage Sensor
vt_sens<3:0>
THRSH
L
|.|
=
rst_v
locke

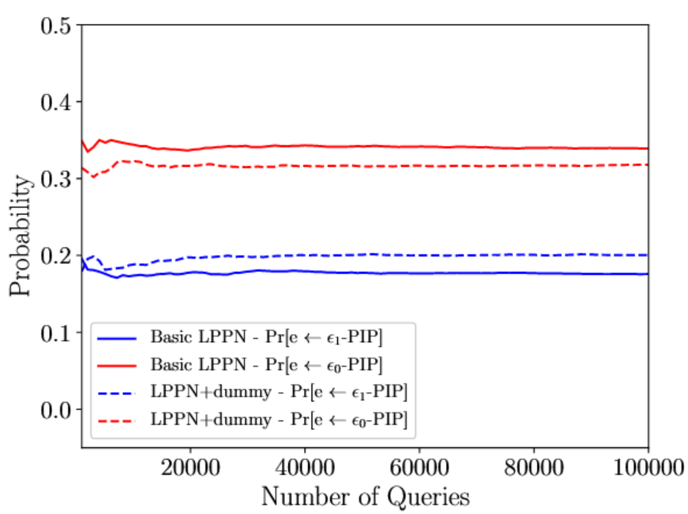


Parallel XOR tree
Serial XOR tree
x
k
512
512
512
512
512 AND
512
enable
clk
XOR1
XOR2
XOR3
XOR4
XOR5
XOR6
en1_out
clk_sk
P
Pcorr out
Pout
en0_out
clk_del
clk
clk_sk
CNTL
6
VDL
clk_del

FPGA instance







- Native Δ of $\approx$ 8.2%

$\Rightarrow$ As hard as LPN with $\varepsilon \approx 0.2$

- Reduced to 5.8% with design tweaks (dummy operations)

$\Rightarrow$ As hard as LPN with $\varepsilon \approx 0.22$

Secure implementation: masking

- Unprotected implementation: $y \leftarrow \varepsilon\text{–PIP}(x, k)$
- Masking: share $k = k_1 \oplus k_2 \oplus \ldots \oplus k_d$ and compute

  $$y \leftarrow \varepsilon\text{–PIP}(x, k_1) \oplus \langle x, k_2 \rangle \oplus \ldots \oplus \langle x, k_d \rangle$$

- Helps preventing side-channel attacks

- Unprotected implementation: $y \leftarrow \varepsilon\text{–PIP}(x, k)$
- Masking: share $k = k_1 \oplus k_2 \oplus \ldots \oplus k_d$ and compute

$$y \leftarrow \varepsilon\text{–PIP}(x, k_1) \oplus \langle x, k_2 \rangle \oplus \ldots \oplus \langle x, k_d \rangle$$

- Helps preventing side-channel attacks
- Also mitigates data-dependent errors since for $z = \varepsilon\text{–PIP}(x, k_1)$ the adversary only sees $\Pr[z|l] = \frac{1}{2} + \delta$, leading to LPN-OD with

$$\varepsilon_0' = \varepsilon_0 \cdot \left(\frac{1}{2} + \delta\right) + \varepsilon_1 \cdot \left(\frac{1}{2} - \delta\right)$$

$$\varepsilon_1' = \varepsilon_1 \cdot \left(\frac{1}{2} + \delta\right) + \varepsilon_0 \cdot \left(\frac{1}{2} - \delta\right)$$

- Conceptually appealing but provocative
  + Performance gains & side-channel security
  – Physical assumption (harder to assess)

- Interesting mix between physics & maths
  - Physics used to limit defaults by design
    - Many other implementations could be studied
  - Maths to prove security despite defaults

- Next step: from LP(P)N to LW(P)E and PQ crypto
  - May not be obvious with KEMS using FO-transform

THANKS

- D. Kamel, F.-X. Standaert, A. Duc, D. Flandre, F. Berti, *Learning with Physical Noise or Errors*, in IEEE Transactions on Dependable and Secure Computing, vol 17, num 5, pp 957-971, October 2020
- D. Kamel, D. Bellizia, F.-X. Standaert, D. Flandre, D. Bol, *Demonstrating an LPPN Processor*, in the proceedings of ASHES 2018, pp 18-23, Toronto, Canada, October 2018
- D. Kamel, D. Bellizia, O. Bronchain, F.-X. Standaert, *Side-channel Analysis of a Learning Parity with Physical Noise Processor*, in the Journal of Cryptographic Engineering, vol 11, num 2, pp 171-179, June 2021
- D. Bellizia, C. Hoffmann, D. Kamel, Hanlin Liu, P. Meaux, F.-X. Standaert, Yu Yu, *Learning Parity with Physical Noise: Imperfections, Reductions and FPGA Prototype*, in IACR Transactions on Cryptographic Hardware and Embedded Systems, vol 2021, num 3, pp 390-417