Set-swapping Attack on the Classic McEliece Cryptosystem

PQ-TLS project – Axis 3

Brice Colombier

June 5, 2024

Laboratoire Hubert Curien
UMR ▪ CNRS ▪ 5516 ▪ Saint-Étienne

Joint work

Joint work with:

- Pierre-Louis Cayrel (SESAM team, LabHC, Saint-Étienne)
- Vlad-Florin Dragoi (Univ. Arad, Romania)
- Vincent Grosso (SESAM team, LabHC, Saint-Étienne)
- Alexandre Menu (SAS team, EMSE, Gardanne)
- Lilian Bossuet (SESAM team, LabHC, Saint-Étienne)

Agenda

1. Classic McEliece
   - Encapsulation

2. Syndrome decoding problem (which is NP-complete)
   - How to make it “easier” to solve
     and actually solve it

3. Practical aspects
   - How to make it happen by way of physical attacks

Classic McEliece encapsulation

Classic McEliece encapsulation

Classic McEliece is a **Key Encapsulation Mechanism**

- `KeyGen() ->` ($\mathbf{H}_{\text{pub}}$, k_{priv})
- `Encap(`$\mathbf{H}_{\text{pub}}$`) ->` ($\mathbf{s}$, k_{session})
- `Decap(`$\mathbf{s}$, k_{priv}`) ->` (k_{session})

The Encapsulation procedure (Niederreiter encryption [Nie86]) establishes a **shared secret**.

- `Encap(`$\mathbf{H}_{\text{pub}}$`) ->` ($\mathbf{s}$, k_{session})

  Generate a random vector $\mathbf{e} \in \mathbb{F}_2^{\mathbf{n}}$ of Hamming weight $\mathbf{t}$ (($\mathbf{n}$; $\mathbf{t}$): security parameters)

  Compute $\mathbf{s} = \mathbf{H}_{\text{pub}}\mathbf{e}$

  Compute the hash: $k_{\text{session}} = \text{H}(1, \mathbf{e}, \mathbf{s})$

[Nie86] H. Niederreiter. "Knapsack-Type Cryptosystems and Algebraic Coding Theory". In: **Problems of Control and Information Theory** (1986)

Classic McEliece encapsulation

Classic McEliece is a **Key Encapsulation Mechanism**

- `KeyGen() ->` $(\mathbf{H}_{\text{pub}}, \text{k}_{\text{priv}})$
- `Encap(`$\mathbf{H}_{\text{pub}}$`) ->` $(\mathbf{s}, \text{k}_{\text{session}})$
- `Decap(`$\mathbf{s}, \text{k}_{\text{priv}}$`) ->` $(\text{k}_{\text{session}})$

The Encapsulation procedure (Niederreiter encryption [Nie86]) establishes a **shared secret**.

- `Encap(`$\mathbf{H}_{\text{pub}}$`) ->` $(\mathbf{s}, \text{k}_{\text{session}})$

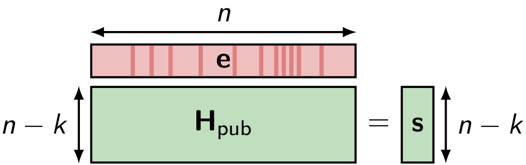
  Generate a random vector $\mathbf{e} \in \mathbb{F}_2^{\mathbf{n}}$ of Hamming weight $\mathbf{t}$ (($\mathbf{n}; \mathbf{t}$): security parameters)

  Compute $\mathbf{s} = \mathbf{H}_{\text{pub}}\mathbf{e}$

  Compute the hash: $\text{k}_{\text{session}} = \text{H}(1, \mathbf{e}, \mathbf{s})$

[Nie86] H. Niederreiter. "Knapsack-Type Cryptosystems and Algebraic Coding Theory". In: **Problems of Control and Information Theory** (1986)

Classic McEliece parameters



n	k	$(n-k)$	t
3488	2720	768	64
4608	3360	1248	96
6688	5024	1664	128
6960	5413	1547	119
8192	6528	1664	128

The public key $\mathbf{H}_{\text{pub}}$ is **very large**.

Hardware implementations

Embedded/hardware implementations are now feasible: [RKK20] [CC21] [Che+22] [NM24]
Several **strategies** exist to store the (very large) keys:

- Streaming the public key from somewhere else,
- Use a structured code,
- Use a very large microcontroller.

New threats

That makes them vulnerable to **physical attacks** (fault injection & side-channel analysis)

[RKK20] Johannes Roth, Evangelos G. Karatsiolis, and Juliane Krämer. "Classic McEliece Implementation with Low Memory Footprint". In: **CARDIS**. 2020

[CC21] Ming-Shing Chen and Tung Chou. "Classic McEliece on the ARM Cortex-M4". In: **IACR TCHES** (2021)

[Che+22] Po-Jen Chen et al. "Complete and Improved FPGA Implementation of Classic McEliece". In: **IACR TCHES** (2022)

[NM24] Cyrius Nugier and Vincent Migliore. "Acceleration of a Classic McEliece Postquantum Cryptosystem With Cache Processing". In: **IEEE Micro** (2024)

Syndrome decoding problem

Syndrome decoding problem

Syndrome decoding problem

Input: a binary parity-check matrix $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$
a binary vector $\mathbf{s} \in \mathbb{F}_2^{n-k}$
a scalar $t \in \mathbb{N}^+$

Output: a binary vector $\mathbf{x} \in \mathbb{F}_2^n$ with a Hamming weight $\mathrm{HW}(\mathbf{x}) \leq t$ such that: $\mathbf{Hx} = \mathbf{s}$

Known to be an NP-complete problem [BMT78].

[BMT78] Elwyn R. Berlekamp, Robert J. McEliece, and Henk C. A. van Tilborg. "On the inherent intractability of certain coding problems (Corresp.)". In: **IEEE Transactions on Information Theory** (1978)

Syndrome decoding problem

Binary syndrome decoding problem (Binary SDP)

Input: a binary parity-check matrix $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$
a binary vector $\mathbf{s} \in \mathbb{F}_2^{n-k}$
a scalar $t \in \mathbb{N}^+$

Output: a binary vector $\mathbf{x} \in \mathbb{F}_2^n$ with a Hamming weight $\mathrm{HW}(\mathbf{x}) \leq t$ such that: $\mathbf{Hx} = \mathbf{s}$

Integer syndrome decoding problem ($\mathbb{N}$-SDP)

Input: a binary parity-check matrix $\mathbf{H} \in \{0,1\}^{(n-k)\times n}$
a ~~binary~~ vector $\mathbf{s} \in \mathbb{N}^{n-k}$
a scalar $t \in \mathbb{N}^+$

Output: a binary vector $\mathbf{x} \in \{0,1\}^n$ with a Hamming weight $\mathrm{HW}(\mathbf{x}) \leq t$ such that:
$\mathbf{Hx} = \mathbf{s}$

$\mathbb{N}$-SDP as an optimisation problem

Option 1: Consider $\mathbf{H}_{\text{pub}}\mathbf{e} = \mathbf{s}$ as an **optimization problem** and solve it.

Integer syndrome decoding problem ($\mathbb{N}$-SDP)

Input: a matrix $\mathbf{H}_{\text{pub}} \in \mathcal{M}_{n-k,n}(\mathbb{N})$ with $h_{i,j} \in \{0,1\}$ for all i, j
a vector $\mathbf{s} \in \mathbb{N}^{n-k}$
a scalar $t \in \mathbb{N}^{+}$

Output: a vector $\mathbf{e}$ in $\mathbb{N}^n$ with $x_i \in \{0,1\}$ for all i
and with a Hamming weight $\text{HW}(\mathbf{x}) \leq t$ such that: $\mathbf{H}_{\text{pub}}\mathbf{e} = \mathbf{s}$

ILP problem

Let $\mathbf{b} \in \mathbb{N}^n$, $\mathbf{c} \in \mathbb{N}^m$ and $\mathbf{A} \in \mathcal{M}_{m,n}(\mathbb{N})$ then:

$$\min\{\mathbf{b}^T\mathbf{x} \mid \mathbf{A}\mathbf{x} = \mathbf{c}, \mathbf{x} \in \mathbb{N}^n, \mathbf{x} \geq 0\}$$

with $\mathbf{b} = (1, 1, ..., 1)$ and $\mathbf{x} \in \{0,1\}^n$

$\mathbb{N}$-SDP as an optimisation problem

Option 1: Consider $\mathbf{H}_{\text{pub}}\mathbf{e} = \mathbf{s}$ as an **optimization problem** and solve it.

Integer syndrome decoding problem ($\mathbb{N}$-SDP)

Input: a matrix $\mathbf{H}_{\text{pub}} \in \mathcal{M}_{n-k,n}(\mathbb{N})$ with $h_{i,j} \in \{0,1\}$ for all i, j
a vector $\mathbf{s} \in \mathbb{N}^{n-k}$
a scalar $t \in \mathbb{N}^{+}$

Output: a vector $\mathbf{e}$ in $\mathbb{N}^n$ with $x_i \in \{0,1\}$ for all i
and with a Hamming weight $\text{HW}(\mathbf{x}) \leq t$ such that: $\mathbf{H}_{\text{pub}}\mathbf{e} = \mathbf{s}$

ILP problem

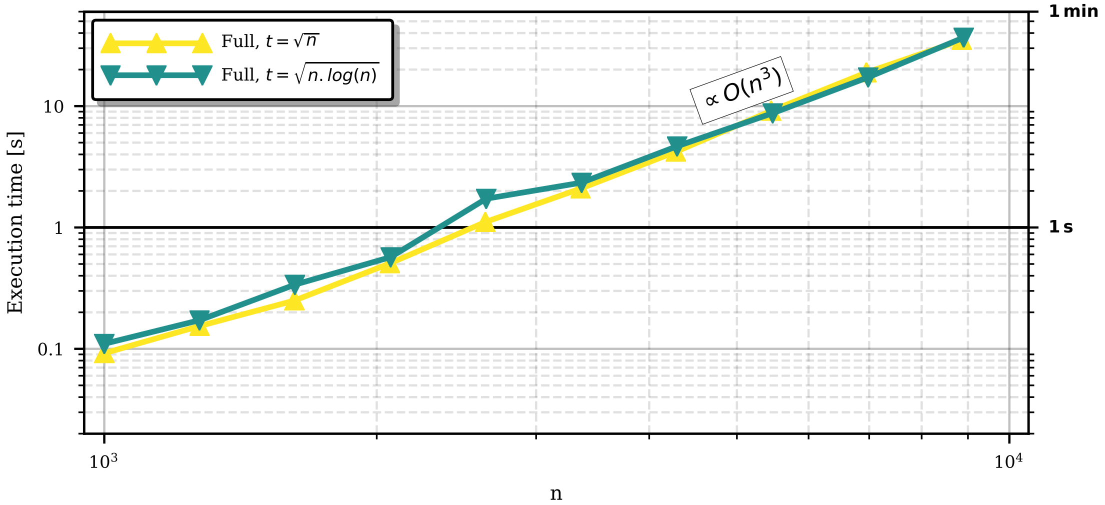
Let $\mathbf{b} \in \mathbb{N}^n$, $\mathbf{c} \in \mathbb{N}^m$ and $\mathbf{A} \in \mathcal{M}_{m,n}(\mathbb{N})$ then:

$$\min\{\mathbf{b}^T\mathbf{x} \mid \mathbf{A}\mathbf{x} = \mathbf{c}, \mathbf{x} \in \mathbb{N}^n, \mathbf{x} \geq 0\}$$

with $\mathbf{b} = (1, 1, ..., 1)$ and $\mathbf{x} \in \{0,1\}^n$

Solved by **integer linear programming**
(using `Scipy.optimize.linprog` for example)

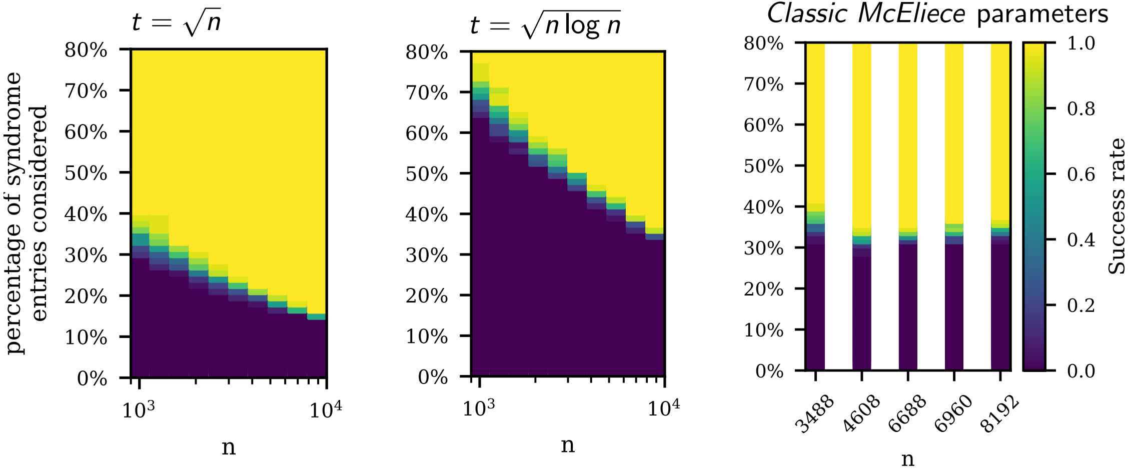
Integer Linear Programming



For *Classic McEliece*: $3488 < n < 8192$

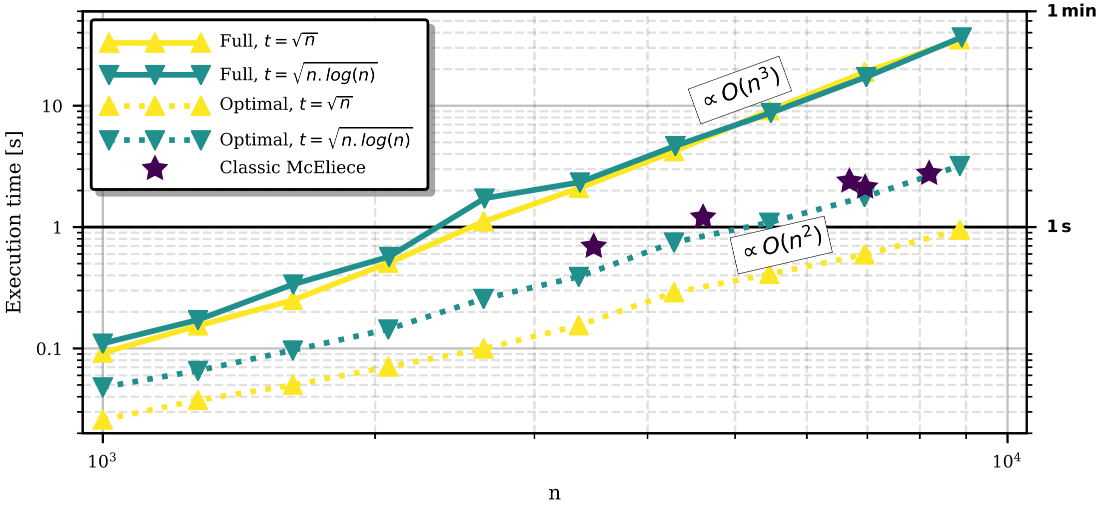
Required fraction of faulty syndrome entries

We observed that only a **fraction** of the faulty syndrome entries is enough to solve the problem.



For *Classic McEliece*, **less than** 40 % faulty syndrome entries is enough.

Experimental results



When considering the **optimal fraction**, time complexity drops from $\mathcal{O}(n^3)$ to $\mathcal{O}(n^2)$.
The largest parameters can be attacked in **a few seconds** on a desktop computer.

$\mathbb{N}$-SDP as an optimization problem: summary

Considering the $\mathbb{N}$-SDP as an optimization problem [Cay+21]

- 👍 easy to **express**,
- 👍 allows to use a **generic ILP solver**,
- 👍 is reasonably **efficient**,
- 👎 does not tolerate **errors** in the integer syndrome.

[Cay+21] Pierre-Louis Cayrel et al. "Message-Recovery Laser Fault Injection Attack on the Classic McEliece Cryptosystem". In: **EUROCRYPT**. 2021

Option 2: Reframe $\mathbf{H}_{\text{pub}}\mathbf{e} = \mathbf{s}$ as the Quantitative Group Testing problem [FL20]

Abstract

Given a random Bernoulli matrix $A \in \{0,1\}^{m \times n}$, an integer $0 < k < n$ and the vector $y := Ax$, where $x \in \{0,1\}^n$ is of Hamming weight k, the objective in the *Quantitative Group Testing* (QGT) problem is to recover x.

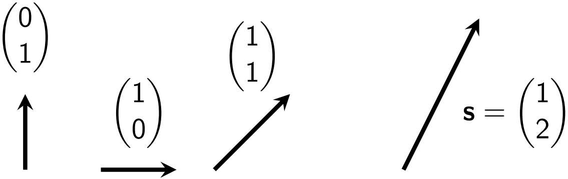
We want to find **which columns** of $\mathbf{H}_{\text{pub}}$ **contributed the most** to $\mathbf{s}$.

[FL20] Uriel Feige and Amir Lellouche. "Quantitative Group Testing and the rank of random matrices". In: **CoRR** (2020). arXiv: 2006.09074

The score function

Example: $t = 2 = \mathrm{HW}(\mathbf{e})$

$$\mathbf{H}_{\text{pub}}\mathbf{e} = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix} . \mathbf{e} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$



The dot product [FL20] can be used to compute a **score** for a column:

Score function

$$\psi_i(\mathbf{s}) = \mathbf{H}_{pub[,i]} \cdot \mathbf{s} + \bar{\mathbf{H}}_{pub[,i]} \cdot \bar{\mathbf{s}} \qquad \text{with } \bar{\mathbf{H}} = 1 - \mathbf{H} \qquad \text{and } \bar{\mathbf{s}} = t - \mathbf{s}$$

$$\psi_0(\mathbf{s}) = 3 \qquad \psi_1(\mathbf{s}) = 1 \qquad \psi_2(\mathbf{s}) = 3$$

[FL20] Uriel Feige and Amir Lellouche. "Quantitative Group Testing and the rank of random matrices". In: **CoRR** (2020). arXiv: 2006.09074

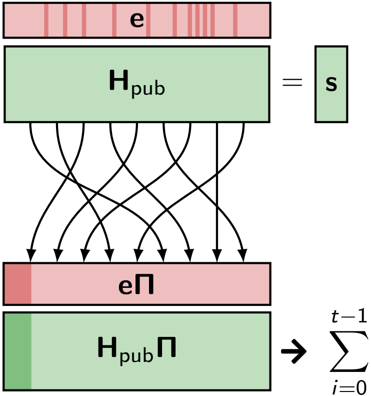
From the score to the support

Algorithm 1 Permutation from score

---

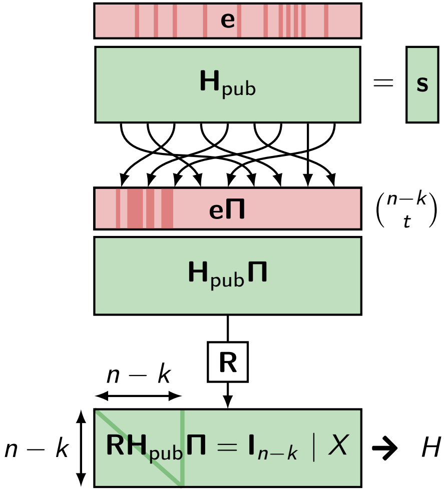
for i $\leftarrow 0$ to $n-1$ **do**
  Compute $\psi_i(\mathbf{s})$
$\mathbf{\Pi} \leftarrow$ sort $\psi(\mathbf{s})$ in descending order
Return $\mathbf{\Pi}$

Best-case scenario: t-threshold decoder



$$\rightarrow \sum_{i=0}^{t-1} (\mathbf{H}_{\text{pub}}\mathbf{\Pi})_{[,:t]} = \mathbf{s} \quad ✓$$

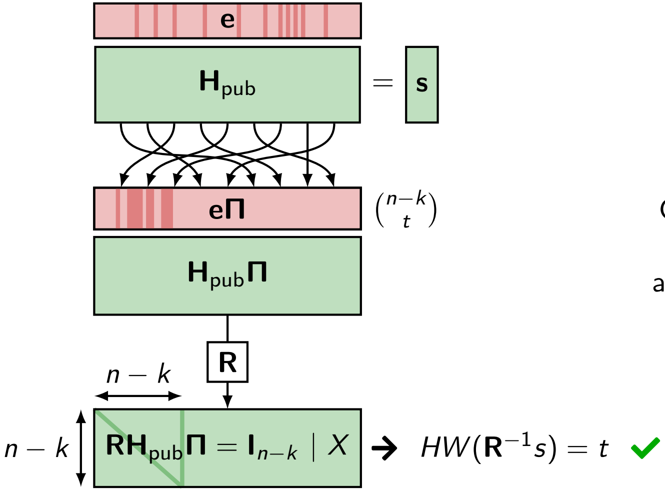
Rank-threshold score decoder: Information Set Decoding *à la* Prange [Pra62]



$\rightarrow HW(\mathbf{R}^{-1}s) = t$ ✓

[Pra62] Eugene Prange. "The Use of Information Sets in Decoding Cyclic Codes". In: **IRE Transactions on Information Theory** (1962)

Rank-threshold score decoder: Information Set Decoding *à la* Prange [Pra62]



Can be improved by allowing δ ones in the last k positions of $e\mathbf{\Pi}$ and use more advanced ISD variants.

[Pra62] Eugene Prange. "The Use of Information Sets in Decoding Cyclic Codes". In: **IRE Transactions on Information Theory** (1962)

Solving ℕ-SDP with the score function

Solving ℕ-SDP with the score function [Col+22]

- 👍 is computationally efficient
- 👍 tolerates some errors in the integer syndrome
- 👍 gets more efficient with larger cryptographic parameters
- 👎 does not cope so well with high noise levels

[Col+22] Brice Colombier et al. "Profiled Side-Channel Attack on Cryptosystems Based on the Binary Syndrome Decoding Problem". In: **IEEE TIFS** (2022)

Practical aspects: physical attacks

Objective

$\mathbb{N}$-SDP framework: compute $\boxed{\mathbf{s} = \mathbf{H}_{\text{pub}}\mathbf{e}}$ over $\mathbb{N}$ instead of $\mathbb{F}_2$

Algorithm 2 Schoolbook matrix-vector multiplication over $\mathbb{F}_2$

```
function MAT_VEC_MULT_SCHOOLBOOK(mat, vec)
  for row ← 0 to n − k − 1 do
    syn[row] = 0                                          ▷ Initialization
  for row ← 0 to n − k − 1 do
    for col ← 0 to n − 1 do
      syn[row] ^= mat[row][col] & vec[col]    ▷ multiply-accumulate
  return syn
```

Objective

$\mathbb{N}$-SDP framework: compute $\boxed{\mathbf{s} = \mathbf{H}_{\text{pub}}\mathbf{e}}$ over $\mathbb{N}$ instead of $\mathbb{F}_2$

Algorithm 2 Schoolbook matrix-vector multiplication over $\mathbb{F}_2$

```
function MAT_VEC_MULT_SCHOOLBOOK(mat, vec)
  for row ← 0 to n − k − 1 do
    syn[row] = 0                                    ▷ Initialization
  for row ← 0 to n − k − 1 do
    for col ← 0 to n − 1 do
      syn[row] ^= mat[row][col] & vec[col]          ▷ multiply-accumulate
  return syn
```

Option 1: laser fault injection attack

Targeting the XOR operation, considering the Thumb instruction set.

bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
`EORS: Rd = Rm` ⊕ `Rn`	0	1	0	0	0	0	0	0	0	1	`Rm`			`Rdn`		
`EORS: R1 = R0` ⊕ `R1`	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1

Laser fault injection in flash memory : **mono-bit, bit-set fault model** [Col+19][Men+20].

`ADCS: R1 = R0 + R1`	0	1	0	0	0	0	0	**1**	0	1	0	0	0	0	0	1

[Col+19] Brice Colombier et al. "Laser-induced Single-bit Faults in Flash Memory: Instructions Corruption on a 32-bit Microcontroller". In: **IEEE HOST**. 2019

[Men+20] Alexandre Menu et al. "Single-bit Laser Fault Model in NOR Flash Memories: Analysis and Exploitation". In: **FDTC**. 2020

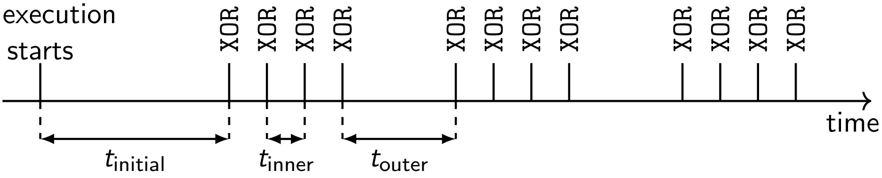
Multiple faults

Three independent delays must be tuned to fault the full matrix-vector multiplication:

t_{initial} **initial** delay before the multiplication starts

t_{inner} delay in the **inner** `for` loop

t_{outer} delay in the **outer** `for` loop



Outcome

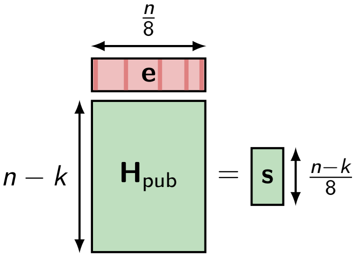
After $n.(n-k)$ faults, we get an **integer syndrome s** $\in \mathbb{N}^{n-k}$

Packed matrix-vector multiplication

Objection: the schoolbook matrix-vector multiplication algorithm is **highly inefficient**!
Each **machine word** stores only **one bit**: a **lot** of memory is wasted.

Algorithm 3 Packed matrix-vector multiplication

```
function MAT_VEC_MULT_PACKED(mat, vec)
  for row ← 0 to ((n − k)/8 − 1) do
    syn[row] = 0                                  ▷ Initialisation
  for row ← 0 to (n − k − 1) do
    b = 0
    for col ← 0 to (n/8 − 1) do
      b ^= mat[row][col] & vec[col]
    b ^= b >> 4
    b ^= b >> 2                                   ▷ Exclusive-OR folding
    b ^= b >> 1
    b &= 1                                        ▷ LSB extraction
    syn[row/8] |= b << (row % 8)                  ▷ Packing
  return syn
```



Option 2: side-channel analysis

Algorithm 4 Packed matrix-vector multiplication

1: ...
2: **for** `col` $\leftarrow$ 0 to $(n/8-1)$ **do**
3: `b ^= mat[row][col] & vec[col]`
4: ...

```
b = 00000000
b = 00000000
b = 00001000
b = 00001000
b = 00001010
```

Option 2: side-channel analysis

Algorithm 4 Packed matrix-vector multiplication

```
...
for col ← 0 to (n/8 − 1) do
    b ^= mat[row][col] & vec[col]
...
```

b = 00000000 HW=0

HD = 0

b = 00000000 HW=0

HD = 1

b = 00001000 HW=1

HD = 0

b = 00001000 HW=1

HD = 1

b = 00001010 HW=2

Option 2: side-channel analysis

Algorithm 4 Packed matrix-vector multiplication

```
1: ...
2: for col ← 0 to (n/8 − 1) do
3:   b ^= mat[row][col] & vec[col]
4: ...
```

	b = 00000000	HW=0
HD = 0		
	b = 00000000	HW=0
HD = 1		
	b = 00001000	HW=1
HD = 0		
	b = 00001000	HW=1
HD = 1		
	b = 00001010	HW=2

Integer syndrome from Hamming distances or Hamming weights

$$s_j = \sum_{i=1}^{\frac{n}{8}-1} \mathrm{HD}(\mathbf{b}_{j,i}, \mathbf{b}_{j,i-1})$$

$$= \sum_{i=1}^{\frac{n}{8}-1} \left| \mathrm{HW}(\mathbf{b}_{j,i}) - \mathrm{HW}(\mathbf{b}_{j,i-1}) \right| \quad \text{if } \mathrm{HD}(\mathbf{b}_{j,i}, \mathbf{b}_{j,i-1}) \leq 1$$

	b = 00001000	HW=1
HD = 2		
	b = 00000100	HW=1

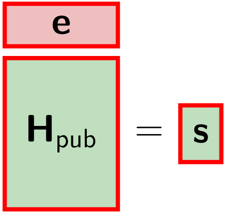
Happens if:

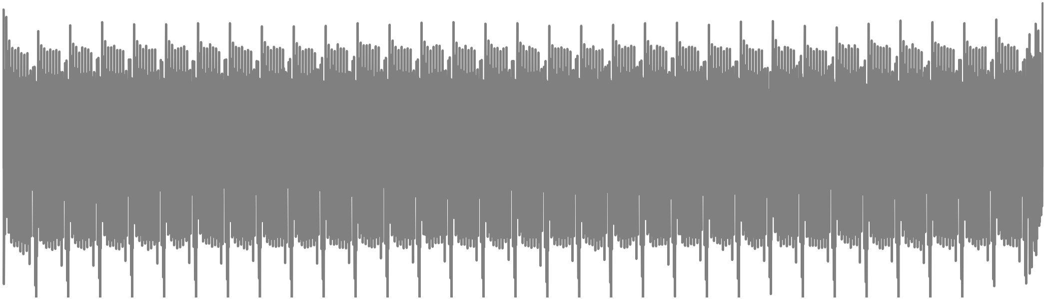
HW(`mat[r][c] & vec[c]`) > 1

Unlikely since $\mathrm{HW}(\mathbf{e}) = t$ is low.

Side-channel analysis for Hamming weight recovery

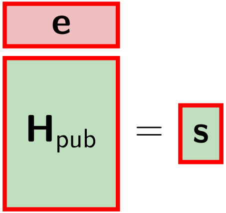
$$\mathbf{s} = \mathbf{H}_{\text{pub}}\mathbf{e}$$





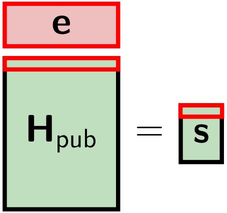
Side-channel analysis for Hamming weight recovery

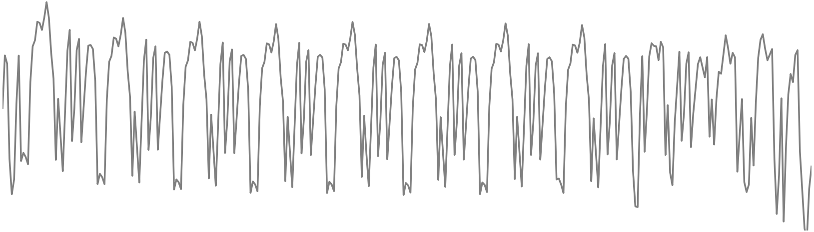
$$\mathbf{s} = \mathbf{H}_{\text{pub}}\mathbf{e}$$



Side-channel analysis for Hamming weight recovery

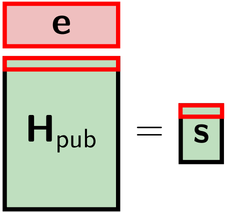
$$\mathbf{s}_j = \mathbf{H}_{pub_{[j,]}}\mathbf{e}$$

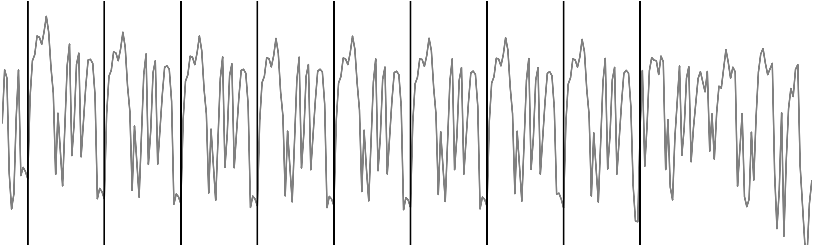




Side-channel analysis for Hamming weight recovery

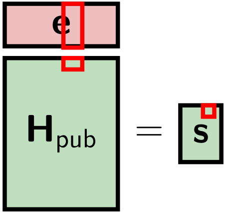
$$\mathbf{s}_j = \mathbf{H}_{pub_{[j,]}}\mathbf{e}$$



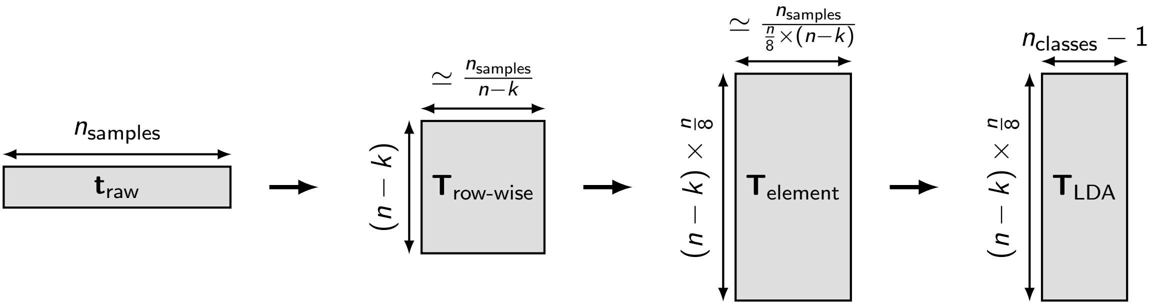


Side-channel analysis for Hamming weight recovery

$$b \mathbin{\hat{}}= \mathbf{H}_{pub_{[j,i]}} \mathbf{e}_i$$



Trace reshaping process



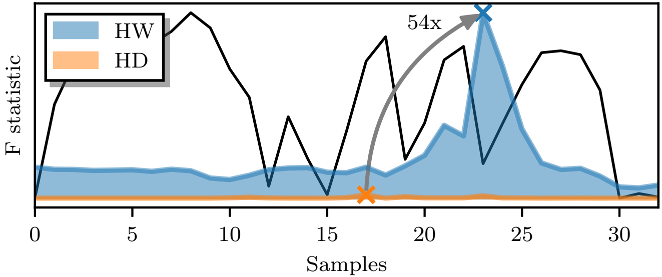
Training phase

- Linear Discriminant Analysis (LDA) for **dimensionality reduction**,
- **One** trace gives $(n-k)\times\frac{n}{8}$ **training samples** $n=8192$ → more than 1.7×10^6
- Fed to a **single** RF classifier (`sklearn.ensemble.RandomForestClassifier`)

Random Forest classifier

Random Forest classifier training:

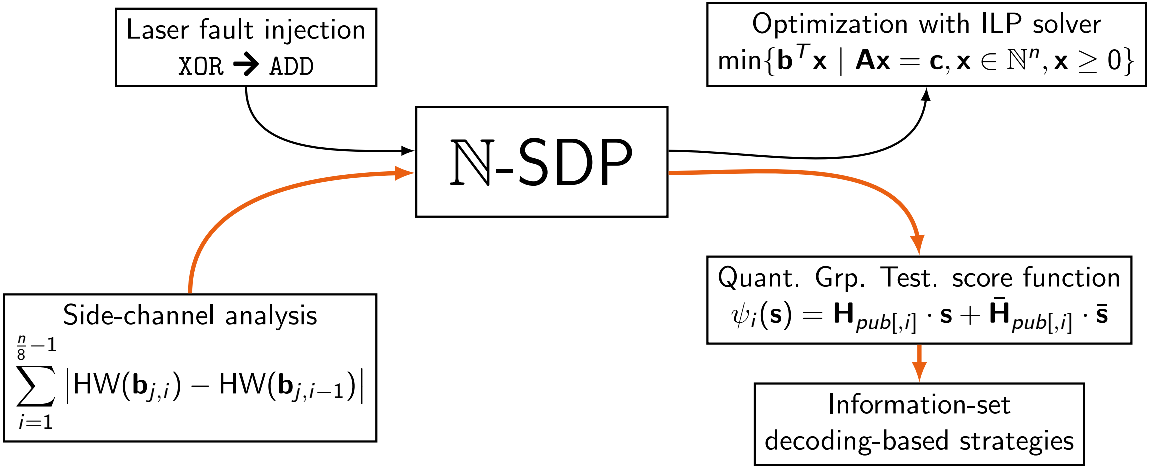
- Hamming weight:
  - $> 99.5\,\%$ test accuracy,
- Hamming distance:
  - $\approx 80\,\%$ test accuracy.



Outcome

- We can recover the **Hamming weight** very accurately,
- but **not the Hamming distance**...
- We can compute a *slightly innacurate* integer syndrome.[7]

[7] Brice Colombier et al. "Profiled Side-Channel Attack on Cryptosystems Based on the Binary Syndrome Decoding Problem". In: **IEEE TIFS** (2022)


Laser fault injection
XOR → ADD
Optimization with ILP solver
$\min\{\mathbf{b}^T\mathbf{x} \mid \mathbf{A}\mathbf{x} = \mathbf{c}, \mathbf{x} \in \mathbb{N}^n, \mathbf{x} \geq 0\}$
$\mathbb{N}$-SDP
Side-channel analysis
$\sum_{i=1}^{\frac{n}{8}-1} \left|\mathrm{HW}(\mathbf{b}_{j,i}) - \mathrm{HW}(\mathbf{b}_{j,i-1})\right|$
Quant. Grp. Test. score function
$\psi_i(\mathbf{s}) = \mathbf{H}_{pub[,i]} \cdot \mathbf{s} + \bar{\mathbf{H}}_{pub[,i]} \cdot \bar{\mathbf{s}}$
Information-set
decoding-based strategies

Dealing with errors in the integer syndrome

The integer syndrome, derived from HW side-channel leakage, is often incorrect [Gro+23]:

- **double-cancellation** errors : **same** Hamming weight but **different** value

HD = 2

```
b = 00001000    HW=1
b = 00000100    HW=1
```

only gets **worse** when the register size grows (32, 64)...

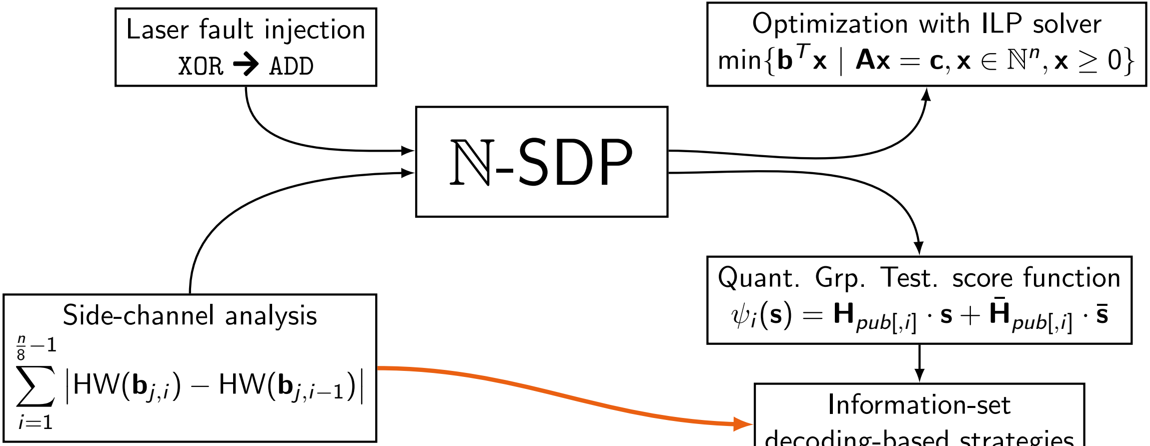
- classifier inaccuracy for **high noise-levels** [Dra+22].

[Gro+23] Vincent Grosso et al. "Punctured Syndrome Decoding Problem - Efficient Side-Channel Attacks Against Classic McEliece". In: **COSADE**. 2023

[Dra+22] Vlad-Florin Dragoi et al. "Integer Syndrome Decoding in the Presence of Noise". In: **IEEE ITW**. 2022

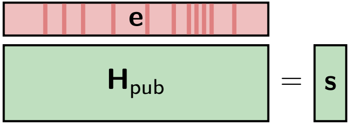
Back to SDP

Back to SDP

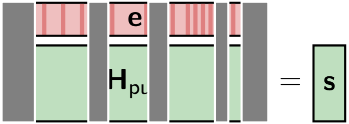

Laser fault injection
XOR → ADD
$\mathbb{N}$-SDP
Optimization with ILP solver
$\min\{\mathbf{b}^T\mathbf{x} \mid \mathbf{A}\mathbf{x} = \mathbf{c}, \mathbf{x} \in \mathbb{N}^n, \mathbf{x} \geq 0\}$
Side-channel analysis
$\sum_{i=1}^{\frac{n}{8}-1} \left| \mathrm{HW}(\mathbf{b}_{j,i}) - \mathrm{HW}(\mathbf{b}_{j,i-1}) \right|$
Quant. Grp. Test. score function
$\psi_i(\mathbf{s}) = \mathbf{H}_{pub[,i]} \cdot \mathbf{s} + \bar{\mathbf{H}}_{pub[,i]} \cdot \bar{\mathbf{s}}$
Information-set
decoding-based strategies

Back to SDP: punctured syndrome decoding problem

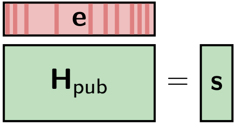
SDP



Punctured SDP [Gro+23]



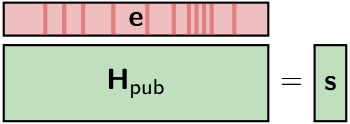
Removing columns associated with an all-zero word in **e**.
(can be detected by side-channel analysis)



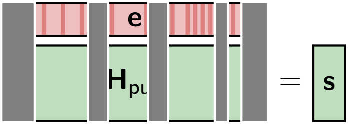
[Gro+23] Vincent Grosso et al. "Punctured Syndrome Decoding Problem - Efficient Side-Channel Attacks Against Classic McEliece". In: **COSADE**. 2023

Back to SDP: punctured syndrome decoding problem

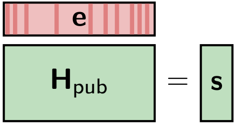
SDP



Punctured SDP [Gro+23]



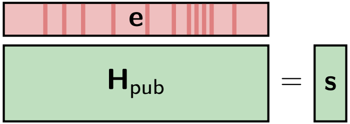
Removing columns associated with an all-zero word in **e**.
(can be detected by side-channel analysis)

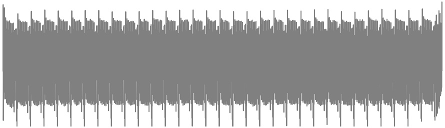


- reduces the code **size**
  - ISD strategies more applicable
- not for large **registers** (32, 64)
  - not enough all-zero words in **e**

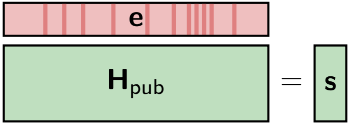
[Gro+23] Vincent Grosso et al. "Punctured Syndrome Decoding Problem - Efficient Side-Channel Attacks Against Classic McEliece". In: **COSADE**. 2023

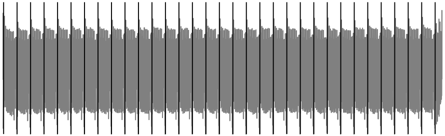
Back to SDP: *t*-test attack



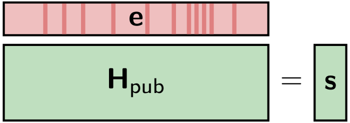


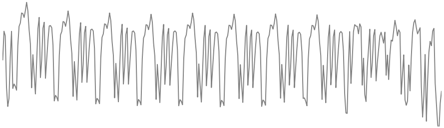
Back to SDP: t-test attack

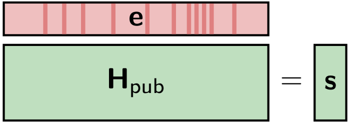


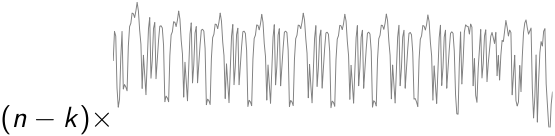


Back to SDP: t-test attack



$(n-k)\times$ 



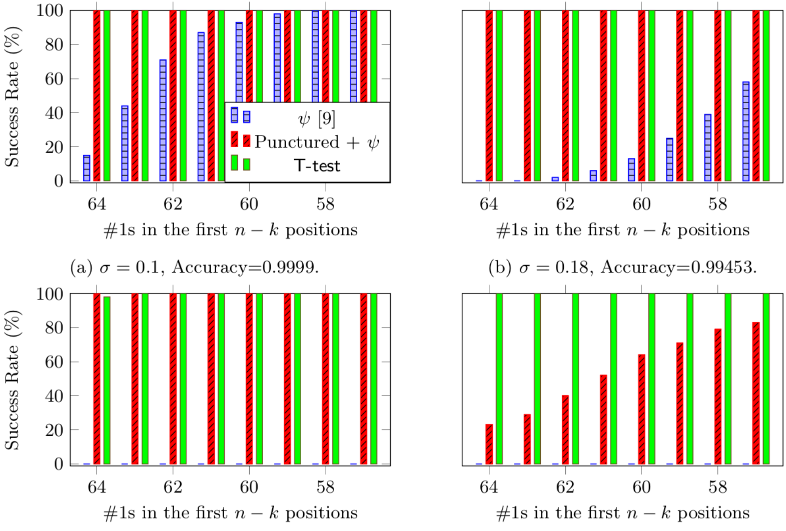


Identify the top t columns of $\mathbf{H}_{\text{pub}}$ that **best explain** the observed power consumption.

Algorithm 5 t-test attack

for `i` $\leftarrow 0$ to $n-1$ **do**
  for every sample **do**
    G_0 := `subtraces[sample]` where $H[:, i] = 0$
    G_1 := `subtraces[sample]` where $H[:, i] = 1$
    t-test(G_0, G_1)
  `t_vals[i]` = max(t-tests)
Return indexes of top t values in `t_vals`

Experimental results



(a) $\sigma = 0.1$, Accuracy=0.9999.

(b) $\sigma = 0.18$, Accuracy=0.99453.

(c) $\sigma = 0.26$, Accuracy=0.94553.

(d) $\sigma = 0.30$, Accuracy=0.90442.

Fig. 3: Success rate of the three methods for 8-bit words and different noise levels.

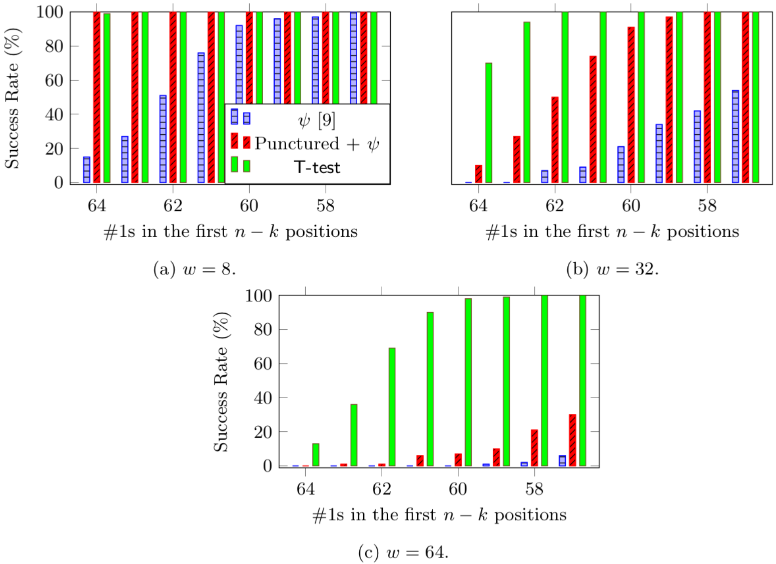
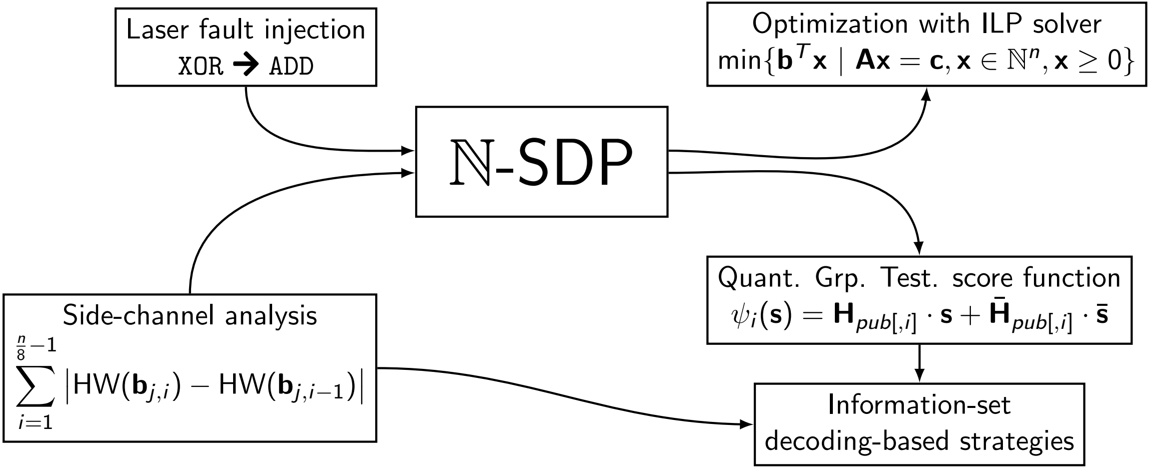


Fig. 5: Comparison of the three methods for different register sizes at noise level $\sigma = 0.16$.

Conclusion

Conclusion


Laser fault injection
XOR → ADD
$\mathbb{N}$-SDP
Optimization with ILP solver
$\min\{\mathbf{b}^T\mathbf{x} \mid \mathbf{A}\mathbf{x} = \mathbf{c}, \mathbf{x} \in \mathbb{N}^n, \mathbf{x} \geq 0\}$
Side-channel analysis
$\sum_{i=1}^{\frac{n}{8}-1} \left|\mathrm{HW}(\mathbf{b}_{j,i}) - \mathrm{HW}(\mathbf{b}_{j,i-1})\right|$
Quant. Grp. Test. score function
$\psi_i(\mathbf{s}) = \mathbf{H}_{pub[,i]} \cdot \mathbf{s} + \bar{\mathbf{H}}_{pub[,i]} \cdot \bar{\mathbf{s}}$
Information-set
decoding-based strategies

Future works and perspectives

Future works:

- Study the ISD **enumeration** step starting with the **initial permutation**
- Better understand the “**noise**” on the integer syndrome, and remove it?
- Target **hardware** implementations and exploit **Hamming distance** leakage

Perspectives:

- Recover **long-term secrets** too
- Swap the sets on **other cryptosystems**!

Future works and perspectives

Future works:

- Study the ISD **enumeration** step starting with the **initial permutation**
- Better understand the "**noise**" on the integer syndrome, and remove it?
- Target **hardware** implementations and exploit **Hamming distance** leakage

Perspectives:

- Recover **long-term secrets** too
- Swap the sets on **other cryptosystems**!

— **Questions ?** —