Physical Security of Code-based Cryptosystems based on the Syndrome Decoding Problem

Séminaire Creach Labs “Sécurité des composants” 2023

Brice Colombier

joint work with: Pierre-Louis Cayrel, Vlad Drăgoi, Vincent Grosso

June 29th 2023

LABORATOIRE HUBERT CURIEN

UMR • CNRS • 5516 • SAINT-ETIENNE

Context

- 2016 NIST called for proposals for **post-quantum cryptography** algorithms
- 2017 Round 1: 69 candidates,
- 2019 Round 2: 26 candidates,
- 2020 Round 3: 7 finalists (+8 alternate).
- 2022 **Round 4**
  - Selected: CRYSTALS-KYBER
  - Candidates: BIKE, **Classic McEliece** [1], HQC and ~~SIKE~~.

Research challenges

- *“More hardware implementations”*
- *“Side-channel attacks”*
- *“Side-channel resistant implementations”*

Dustin Moody (NIST), PKC 2022

[1] M. R. Albrecht, D. J. Bernstein, T. Chou, et al. **Classic McEliece: conservative code-based cryptography: cryptosystem specification**. Tech. rep. National Institute of Standards and Technology, 2022.

Classic McEliece

Classic McEliece

Classic McEliece is a **Key Encapsulation Mechanism**, based on the **Niederreiter cryptosystem** [2].

- `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 establishes a **shared secret**.

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

  Generate a random vector $\mathbf{e} \in \mathbb{F}_2^n$ of Hamming weight t
  Compute $\mathbf{s} = \mathbf{H}_{\text{pub}}\mathbf{e}$
  Compute the hash: $k_{\text{session}} = \mathrm{H}(1, \mathbf{e}, \mathbf{s})$

[2] H. Niederreiter. "Knapsack-Type Cryptosystems and Algebraic Coding Theory". In: **Problems of Control and Information Theory** 15.2 (1986), pp. 159–166.

Security

The security of the Niederreiter cryptosystem relies on the **syndrome decoding problem**.

Syndrome decoding problem

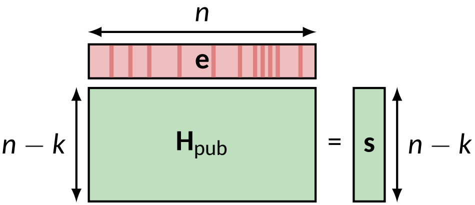
Input: a binary matrix $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$
a binary vector $\mathbf{s} \in \mathbb{F}_2^{n-k}$
a scalar $t \in \mathbb{N}^+$

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 a **hard** problem [3].

[3] E. R. Berlekamp, R. J. McEliece, and H. C. A. van Tilborg. "On the inherent intractability of certain coding problems (Corresp.)". In: **IEEE Transactions on Information Theory** 24.3 (1978), pp. 384–386.

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 **huge**! Up to 1.7 MB.

Hardware implementations

Implementations on embedded systems are now feasible : [4] [5] [6]
Reference hardware target : Arm® Cortex®-M4

Several **strategies** 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)

[4] S. Heyse. "Low-Reiter: Niederreiter Encryption Scheme for Embedded Microcontrollers". In: **International Workshop on Post-Quantum Cryptography**. Vol. 6061. Darmstadt, Germany: Springer, May 2010, pp. 165–181.

[5] J. Roth, E. G. Karatsiolis, and J. Krämer. "Classic McEliece Implementation with Low Memory Footprint". In: **CARDIS**. vol. 12609. Virtual Event: Springer, Nov. 2020, pp. 34–49.

[6] M. Chen and T. Chou. "Classic McEliece on the ARM Cortex-M4". In: **IACR TCHES** 2021.3 (2021), pp. 125–148.

A “modified” syndrome decoding problem

Syndrome decoding problem

Binary syndrome decoding problem (Binary SDP)

Input: a binary 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 : $\mathbf{Hx} = \mathbf{s}$

Syndrome decoding problem

Binary syndrome decoding problem (Binary SDP)

Input: a binary 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 : $\mathbf{Hx} = \mathbf{s}$

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

Input: a binary 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 : $\mathbf{Hx} = \mathbf{s}$

Syndrome decoding problem

Binary syndrome decoding problem (Binary SDP)

Input: a binary 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 : $\mathbf{Hx} = \mathbf{s}$

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

Input: a binary matrix $\mathbf{H} \in \{0,1\}^{(n-k)\times n}$
a ~~binary~~ vector $\mathbf{s} \in \mathbb{N}^{n-k}$ ← How do we get this integer syndrome?
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 : $\mathbf{Hx} = \mathbf{s}$

Physical attack #1: Fault injection

Syndrome computation

We target the **syndrome computation**: $\mathbf{s} = \mathbf{H}_{\text{pub}}\mathbf{e}$

Matrix-vector multiplication performed over $\mathbb{F}_2$

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

```
function MAT_VEC_MULT_SCHOOLBOOK(matrix, vector)
  for row ← 0 to n − k − 1 do
    syndrome[row] = 0                                          ▷ Initialisation
  for row ← 0 to n − k − 1 do
    for col ← 0 to n − 1 do
      syndrome[row] ^= matrix[row][col] & vector[col]          ▷ Multiplication and addition
  return syndrome
```

Laser fault injection attack on the schoolbook matrix-vector multiplication

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** [7].

[7] A. Menu, J.-M. Dutertre, J.-B. Rigaud, et al. “Single-bit Laser Fault Model in NOR Flash Memories: Analysis and Exploitation”. In: **FDTC**. Milan, Italy: IEEE, Sept. 2020, pp. 41–48.

Laser fault injection attack on the schoolbook matrix-vector multiplication

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 $\oplus$ Rn	0	1	0	0	0	0	0	0	0	1	Rm			Rdn		
EORS: R1 = R0 $\oplus$ 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** [7].

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

[7] A. Menu, J.-M. Dutertre, J.-B. Rigaud, et al. “Single-bit Laser Fault Model in NOR Flash Memories: Analysis and Exploitation”. In: **FDTC**. Milan, Italy: IEEE, Sept. 2020, pp. 41–48.

Laser fault injection attack on the schoolbook matrix-vector multiplication

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 $\oplus$ Rn	0	1	0	0	0	0	0	0	0	1	Rm			Rdn		
EORS: R1 = R0 $\oplus$ 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** [7].

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

Outcome: switching from $\mathbb{F}_2$ to $\mathbb{N}$

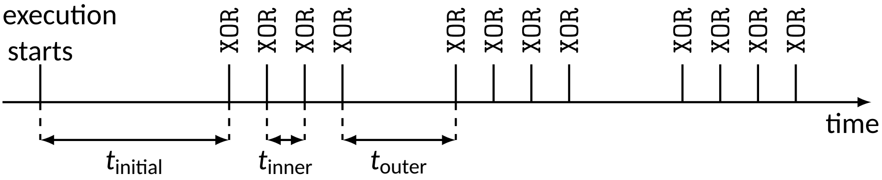
The exclusive-OR (addition over $\mathbb{F}_2$) is turned into an **addition with carry** (addition over $\mathbb{N}$)

[7] A. Menu, J.-M. Dutertre, J.-B. Rigaud, et al. "Single-bit Laser Fault Model in NOR Flash Memories: Analysis and Exploitation". In: **FDTC**. Milan, Italy: IEEE, Sept. 2020, pp. 41–48.

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 a **faulty syndrome s** $\in \mathbb{N}^{n-k}$ [8]

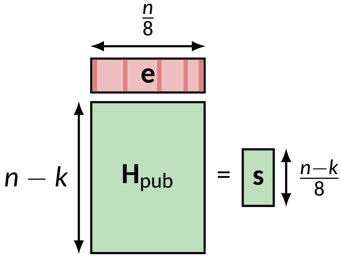
[8] P.-L. Cayrel, B. Colombier, V. Dragoi, et al. "Message-Recovery Laser Fault Injection Attack on the Classic McEliece Cryptosystem". In: **EUROCRYPT**. vol. 12697. Zagreb, Croatia: Springer, Oct. 2021, pp. 438–467

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 Packed matrix-vector multiplication

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



Physical attack #2: Side-channel analysis

Side-channel analysis to obtain the integer syndrome

Algorithm Packed matrix-vector multiplication

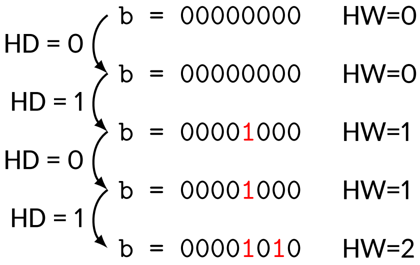
...
for `col` $\leftarrow$ 0 to $(n/8 - 1)$ **do**
    `b ^= matrix[row][col] & vector[col]`
...

```
b = 00000000

b = 00000000

b = 00001000

b = 00001000

b = 00001010
```

Side-channel analysis to obtain the integer syndrome

Algorithm Packed matrix-vector multiplication

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



Side-channel analysis to obtain the integer syndrome

Algorithm Packed matrix-vector multiplication

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

Integer syndrome from Hamming distances or Hamming weights

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

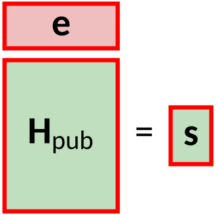
$$= \sum_{i=1}^{\frac{n}{8}-1} \left| \mathrm{HW}(\mathbf{b}_{j,i}) - \mathrm{HW}(\mathbf{b}_{j,i-1}) \right| \text{ if } \mathrm{HD}(\mathbf{b}_{j,i}, \mathbf{b}_{j,i-1}) \leq 1$$

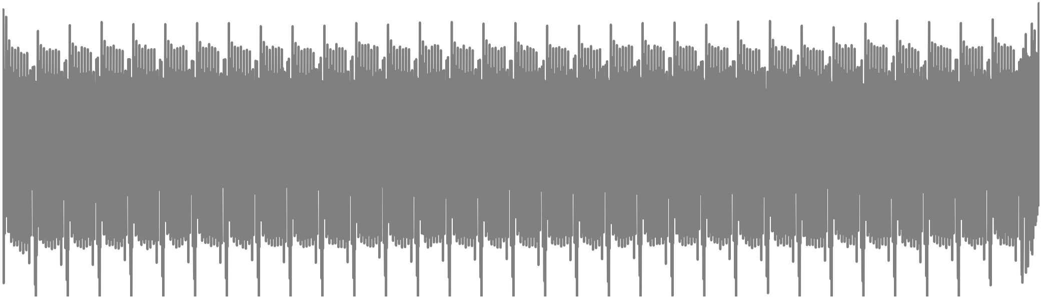
	b = 00001000	HW=1
HD = 2		
	b = 00000100	HW=1

Happens if:
$\mathrm{HW}(\texttt{mat[r][c]} \,\&\, \texttt{vec[c]}) > 1$
Unlikely, since $\mathrm{HW}(\mathbf{e}) = t$ is low.

Side-channel analysis for Hamming weight recovery

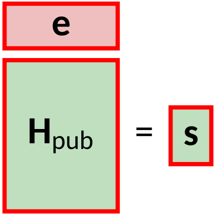
$$\mathbf{s} = \mathbf{H}_{\text{pub}}\mathbf{e}$$





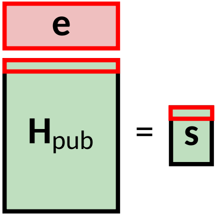
Side-channel analysis for Hamming weight recovery

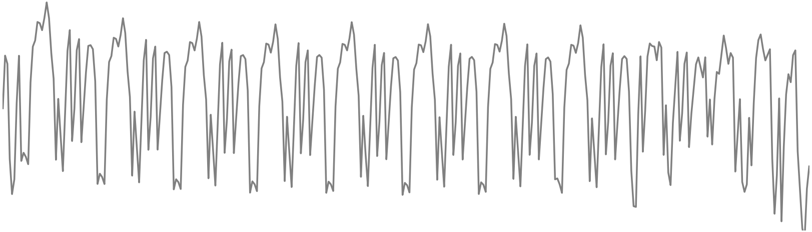
$$\mathbf{s} = \mathbf{H}_{\text{pub}}\mathbf{e}$$



Side-channel analysis for Hamming weight recovery

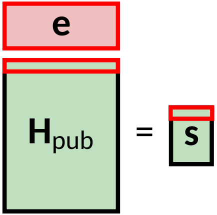
$$\mathbf{s}_j = \mathbf{H}_{pub_{[j,]}}\mathbf{e}$$

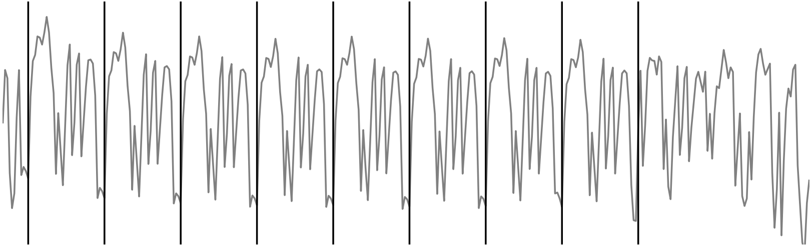




Side-channel analysis for Hamming weight recovery

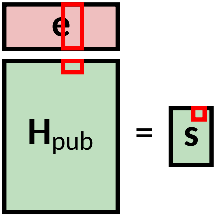
$$\mathbf{s}_j = \mathbf{H}_{pub_{[j,]}}\mathbf{e}$$



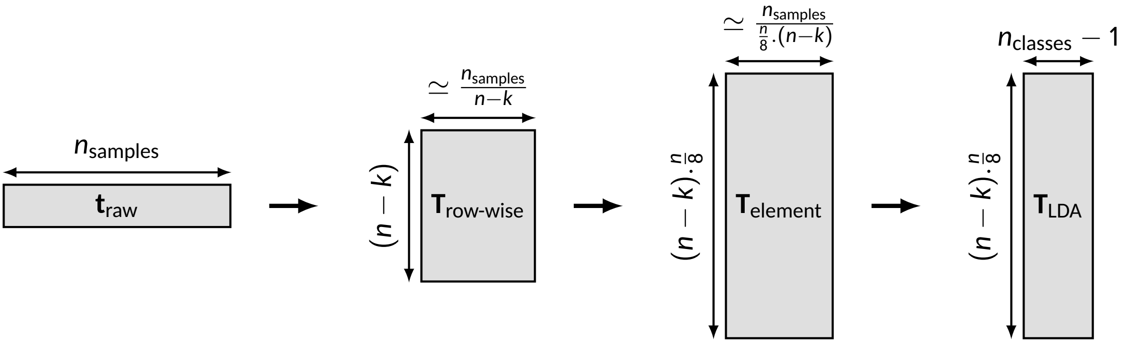


Side-channel analysis for Hamming weight recovery

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



Trace(s) reshaping process



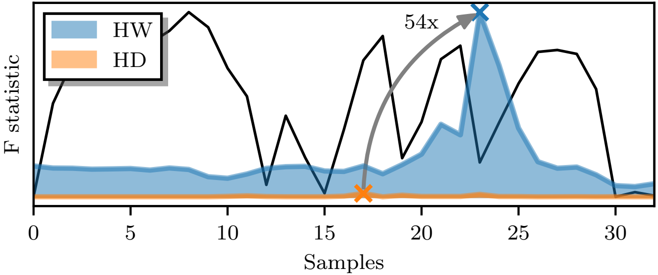
Training phase

- Linear Discriminant Analysis (LDA) for **dimensionality reduction**,
- From a **single** trace, we get $(n-k) \times \frac{n}{8}$ **training samples** $n = 8192$ → more than 1.7×10^6
- Fed to a **single** Random Forest 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. [9]

[9] B. Colombier, V. Dragoi, P. Cayrel, et al. "Profiled Side-Channel Attack on Cryptosystems Based on the Binary Syndrome Decoding Problem". In: **IEEE TIFS** 17 (2022), pp. 3407–3420

Exploiting the integer syndrome

Exploiting the integer syndrome

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

$\mathbb{N}$ 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 $\mathbf{A} \in \mathcal{M}_{m,n}(\mathbb{N})$
We have the following optimization problem:

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

Exploiting the integer syndrome

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

$\mathbb{N}$ 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 $\mathbf{A} \in \mathcal{M}_{m,n}(\mathbb{N})$
We have the following optimization problem:

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

Can be solved by **integer linear programming**.
With `Scipy.optimize.linprog`:

- $n = 8192$: ≈ 5 min...

Does not handle errors in **s** well...

Exploiting the integer syndrome

Option 2 (*Quantitative Group Testing* [10]): which columns of $\mathbf{H}_{\text{pub}}$ "contributed" to the syndrome.

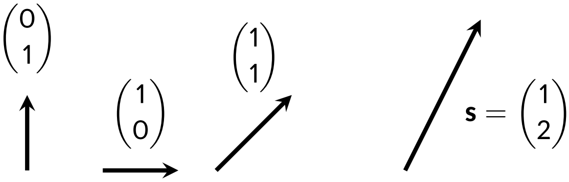
[10] U. Feige and A. Lellouche. "Quantitative Group Testing and the rank of random matrices". In: **CoRR** abs/2006.09074 (2020). arXiv: 2006.09074.

Exploiting the integer syndrome

Option 2 (*Quantitative Group Testing* [10]): which columns of $\mathbf{H}_{\text{pub}}$ "contributed" to the syndrome.

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

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



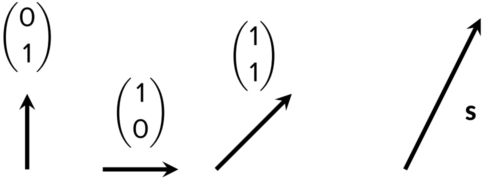
[10] U. Feige and A. Lellouche. "Quantitative Group Testing and the rank of random matrices". In: **CoRR** abs/2006.09074 (2020). arXiv: 2006.09074.

Exploiting the integer syndrome

Option 2 (*Quantitative Group Testing* [10]): which columns of $\mathbf{H}_{\text{pub}}$ "contributed" to the syndrome.

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

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



$$\mathbf{s} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

Score function

The dot product can be used to compute a "score" for every column:

$$\psi(i) = \mathbf{H}_{pub[,i]} \cdot \mathbf{s} + \bar{\mathbf{H}}_{pub[,i]} \cdot \bar{\mathbf{s}} \qquad \text{with } \bar{\mathbf{H}} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \qquad \text{and } \bar{\mathbf{s}} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

- $\psi(0) = 1 \times 0 + 2 \times 1 + 1 \times 1 + 0 \times 0 = 3$
- $\psi(1) = 1$
- $\psi(2) = 3$

[10] U. Feige and A. Lellouche. "Quantitative Group Testing and the rank of random matrices". In: **CoRR** abs/2006.09074 (2020). arXiv: 2006.09074.

Score function : advantages

The **score** of the columns of $\mathbf{H}_{\text{pub}}$ identifies which columns were **involved** in the computation.

From that we can derive the support of the secret vector $\mathbf{e}$.

Computational complexity

- Computing the dot product of two vectors is **very fast**,
- Overall cost for all columns of $\mathbf{H}_{\text{pub}}$: $\mathcal{O}((n-k)\times n) = \mathcal{O}(n^2)$
- $n = 8192 : \approx 0.2\,\text{s}$

Conclusion

Conclusion

Evaluation of post-quantum cryptography algorithms is a long process.

Work is needed in the following areas:

- Efficient implementations,
- Physical security of implementations,
- Protected implementations.

Bring together mathematicians, computer scientists, electrical engineers: SESAM team at LabHC.