Message-recovery Laser Fault Injection Attack on the *Classic McEliece* Cryptosystem

Pierre-Louis Cayrel Brice Colombier Vlad-Florin Drăgoi Alexandre Menu
Lilian Bossuet

Eurocrypt 2021
21/10/2021

LABORATOIRE HUBERT CURIEN UMR • CNRS • 5516 • SAINT-ETIENNE
UGA Université Grenoble Alpes
MINES Saint-Étienne Une école de l'IMT

Introduction

Most public key cryptosystems base their security on the hardness of number theoretic problems. In 1997, Peter Shor showed that quantum algorithms can solve these problems in **polynomial time**[1].

In 2016, NIST started a process[2] for cryptography standards that are **quantum resistant**.

One of the four finalists of Round 3 in the Key Encapsulation Mechanism category (announced July 22, 2020) is *Classic McEliece*[3], based on error-correcting codes.

[1] P. W. Shor. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer". In: *SIAMJournal on Computing* 26.5 (1997), pp. 1484–1509.

[2] https://csrc.nist.gov/Projects/post-quantum-cryptography/

[3] M. R. Albrecht et al. *Classic McEliece*. Tech. rep. https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions. National Institute of Standards and Technology, 2020.

McEliece and Niederreiter PKE schemes

Classic McEliece is based on the Niederreiter[4] cryptosystem:

- KeyGen$(n, k, t) = (\text{pk}, \text{sk})$
  - $\boldsymbol{H}$-parity-check of $\mathcal{C}$[5]
  - Sample $\boldsymbol{S} \in \mathrm{GL}_{n-k}(\mathbb{F}_2)$
  - $\boldsymbol{H}_{\text{pub}} = \boldsymbol{SHP}$
  - $\text{pk} = (\boldsymbol{H}_{\text{pub}}, t)$
  - $\text{sk} = (\boldsymbol{S}, \boldsymbol{H}, \boldsymbol{P})$

- Encrypt$(\boldsymbol{m}, \text{pk}) = \boldsymbol{s}$
  - Encode $\boldsymbol{m} \to \boldsymbol{e}$ ($\boldsymbol{e}$ is a vector of Hamming weight t: $\mathrm{wt}(\boldsymbol{e}) = t$)
  - $\boldsymbol{s} = \boldsymbol{H}_{\text{pub}}\boldsymbol{e}$

[4] H. Niederreiter. "Knapsack-type cryptosystems and algebraic coding theory". In: *Problems of Control and Information Theory* 15.2 (1986), pp. 159–166.

[5] $\mathcal{C}$ is a $[n, k]$ linear code that admits an efficient decoding algorithm able to correct up to t errors

Syndrome decoding problem

Syndrome decoding

Definition (Binary-SDP)

Input: a matrix $\boldsymbol{H} \in \mathcal{M}_{n-k,n}(\mathbb{F}_2)$, a vector $\boldsymbol{s} \in \mathbb{F}_2^{n-k}$ and $t \in \mathbb{N}^+$

Output: a vector $\boldsymbol{x} \in \mathbb{F}_2^n$, with $\mathrm{wt}(\boldsymbol{x}) \leq t$, such that $\boldsymbol{H}\boldsymbol{x} = \boldsymbol{s}$.

Known to be an NP-hard problem[6].

[6] 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.

Syndrome decoding

Definition (Binary-SDP)

Input: a matrix $\boldsymbol{H} \in \mathcal{M}_{n-k,n}(\mathbb{F}_2)$, a vector $\boldsymbol{s} \in \mathbb{F}_2^{n-k}$ and $t \in \mathbb{N}^+$

Output: a vector $\boldsymbol{x} \in \mathbb{F}_2^n$, with $\mathrm{wt}(\boldsymbol{x}) \leq t$, such that $\boldsymbol{H}\boldsymbol{x} = \boldsymbol{s}$.

Known to be an NP-hard problem[6].

Definition (ILP problem)

Let $n, m \in \mathbb{N}^+, \boldsymbol{b} \in \mathbb{N}^n, \boldsymbol{c} \in \mathbb{N}^m$ and $\boldsymbol{A} \in \mathcal{M}_{m,n}(\mathbb{N})$.
The ILP problem is defined as the optimization problem

$$\min\{\boldsymbol{b}^T x | \boldsymbol{A}\boldsymbol{x} = \boldsymbol{c}, \boldsymbol{x} \in \mathbb{N}^n, \boldsymbol{x} \geq 0\}.$$

[6] 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.

$\mathbb{N}$-SDP

Definition ($\mathbb{N}$-SDP)

Input: a matrix $\boldsymbol{H} \in \mathcal{M}_{n-k,n}(\mathbb{N})$ with $h_{i,j} \in \{0,1\}$ for all i, j,
a vector $\boldsymbol{s} \in \mathbb{N}^{n-k}$ and $t \in \mathbb{N}^*$ with $t \neq 0$.

Output: a vector $\boldsymbol{x} \in \mathbb{N}^n$ with $x_i \in \{0,1\}$ and $\mathrm{wt}(\boldsymbol{x}) \leq t$, such that $\boldsymbol{Hx} = \boldsymbol{s}$.

Theorem

Let us suppose that there exists a unique vector $\boldsymbol{x}^* \in \{0,1\}^n$ with $\mathrm{wt}(\boldsymbol{x}^*) = t$, solution to the $\mathbb{N}$-SDP. Then $\boldsymbol{x}^*$ is the optimum solution of an ILP problem.

The $\mathbb{N}$-SDP can be efficiently solved by a linear programming solver.

ILP solver for $\mathbb{N}$-SDP

Input: $\boldsymbol{H}, \boldsymbol{s}, t$

Output: $\boldsymbol{x}$ solution to $\mathbb{N}$-SDP or ERROR

1: Set $\boldsymbol{b} = (1, \ldots, 1)^T$ ▷ Hamming weight constraint
2: Solve $\min\{\boldsymbol{b}^T\boldsymbol{x} \mid \boldsymbol{Hx} = \boldsymbol{s}, 0 \preceq \boldsymbol{x} \preceq 1, \boldsymbol{x} \in \mathbb{R}^n\}$ ▷ Using the LP solver
3: Round the solution $\boldsymbol{x}^*$ to $\boldsymbol{x}^* \in \{0, 1\}^n$
4: **if** $\boldsymbol{Hx}^* = \boldsymbol{s}$ and $\mathrm{wt}(\boldsymbol{x}) \leq t$ **then**
5:     **return** $\boldsymbol{x}^*$
6: **else**
7:     **return** ERROR
8: **end if**

$\boldsymbol{H}$ and t are parameters of the cryptosystem.

Next step

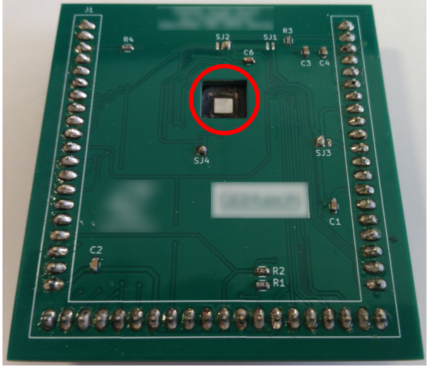
To place ourselves in the $\mathbb{N}$-SDP framework, we must now obtain $\boldsymbol{s} \in \mathbb{N}^{n-k}$.

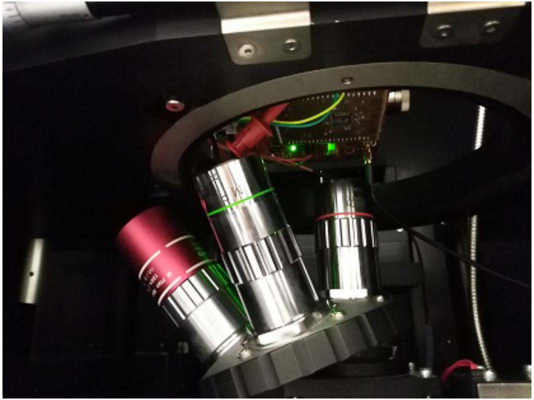
Laser fault injection

Laser fault injection

Laser fault injection was proposed in 2002 by Skorobogatov[7].

Principle: shine an **infrared** laser on the backside of an integrated circuit.





[7] S. P. Skorobogatov and R. J. Anderson. "Optical Fault Induction Attacks". In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Vol. 2523. Redwood Shores, CA, USA: Springer, 2002, pp. 2–12.

Laser fault injection in Flash memory

In a recent line of work, laser fault injection is done in **Flash memory**[8,9,10].

Fault model:

- **mono bit** precision,
- **bit-set** only ($0 \rightarrow 1$),
- **transient** : only **fetched** data is affected, stored data remains **unchanged**.

---

[8] D. S. V. Kumar et al. "An In-Depth and Black-Box Characterization of the Effects of Laser Pulses on ATmega328P". In: *International Conference on Smart Card Research and Advanced Applications*. Vol. 11389. Montpellier, France, Nov. 2018, pp. 156–170.

[9] B. Colombier et al. "Laser-induced Single-bit Faults in Flash Memory: Instructions Corruption on a 32-bit Microcontroller". In: *IEEE International Symposium on Hardware Oriented Security and Trust*. McLean, VA, USA, May 2019, pp. 1–10.

[10] K. Garb and J. Obermaier. "Temporary Laser Fault Injection into Flash Memory: Calibration, Enhanced Attacks, and Countermeasures". In: *International Symposium on On-Line Testing and Robust System Design*. Napoli, Italy: IEEE, July 2020, pp. 1–7.

Capabilities

It is possible to **corrupt the instructions**[11] which are fetched from Flash memory before being executed by the microcontroller.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

EORS instruction: R_d = R_m ^ R_n

0	1	0	0	0	0	0	0	0	1	R_m	R_{dn}

ADCS instruction: R_d = R_m + R_n

0	1	0	0	0	0	0	1	0	1	R_m	R_{dn}

We can turn addition in $\mathbb{F}_2$ into addition in $\mathbb{N}$.

[11] The opcodes for the instructions are available in the ARMv7-M Architecture Reference Manual `https://developer.arm.com/documentation/ddi0403/ee/`

Target function

We target the **matrix-vector multiplication** for **syndrome computation**: $\boldsymbol{s} = \boldsymbol{H}_{\text{pub}}\boldsymbol{e}$.

```
function MAT_VEC_MULT(matrix, error_vector)
    for row ← 0 to (n − k − 1) do
        syndrome[row] = 0
    for row ← 0 to (n − k − 1) do
        for col ← 0 to (n − 1) do
            syndrome[row] ^= matrix[row][col] & error_vector[col]
    return syndrome
```

Outcome

After performing $(n-k) \times n$ faults during **one** encryption, we get $\boldsymbol{s}^* \in \mathbb{N}$.

Experimental results

Experiments

Scipy LP solver[12] and cryptographic parameters of the *Classic McEliece* submission.

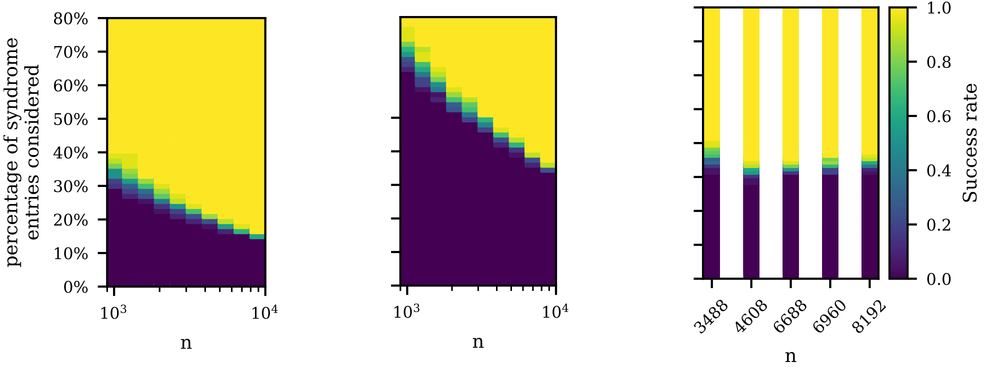
n	3488	4608	6688	6960	8192
k	2720	3360	5024	5413	6528
t	64	96	128	119	128
Equivalent bit-level security	128	196	256	256	256

- $\boldsymbol{H}_{\text{pub}}$ has $n-k$ rows and n columns, parity check matrix of a **random linear code**,
- $\boldsymbol{e}$ is of size n and of Hamming weight t.

[12] `https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.linprog.html`

Required number of faulty syndrome entries

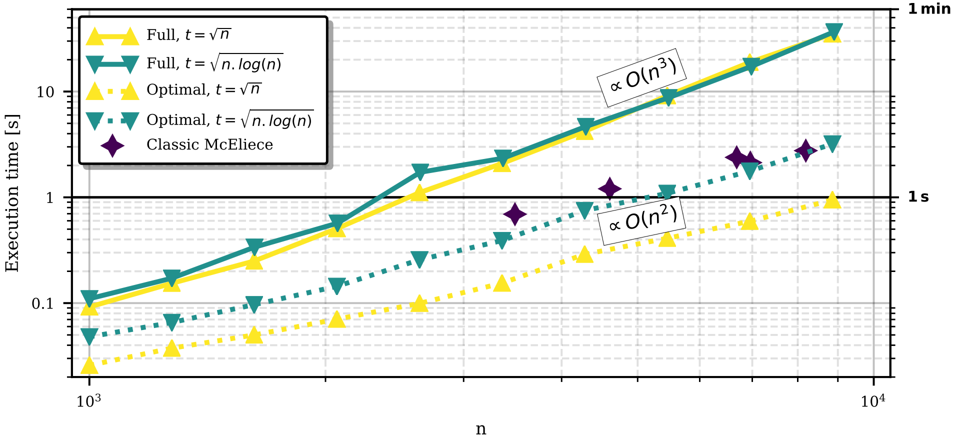
A **fraction** of the faulty syndrome entries is enough to solve the problem.



(a) $t = \sqrt{n}$ (b) $t = \sqrt{n \log(n)}$ (c) *Classic McEliece*

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

Execution time



On a desktop computer (6 cores at 2.8 GHz and 32 GB of RAM):

- *Classic McEliece* with **128**-bit security: **less than 1 second**,
- *Classic McEliece* with **256**-bit security: **less than 3 seconds**.

Conclusion

We presented a **message-recovery attack**:

Given the public matrix ($\boldsymbol{H}_{\mathrm{pub}}$) and a syndrome ($\boldsymbol{s}$), we aim at recovering the private error vector $\boldsymbol{e}$ such that $\boldsymbol{s} = \boldsymbol{H}_{\mathrm{pub}}\boldsymbol{e}$.

1. Perform laser fault injection during the encryption process to modify the instructions, turning the `XOR` into an `ADD`,
2. Obtain a faulty syndrome $\boldsymbol{s}^* \in \mathbb{N}^n$,
3. Solve the resulting problem using an Integer Linear Programming solver,
4. Recover the private error vector in polynomial time.

Perspectives

- Export the idea to key recovery attacks,
- Better study the attack complexity in the "Full" and "Optimal" cases,
- Extend our attack to any code-based cryptosystem for which the implementation is vulnerable to laser fault injection,
- Inspect other fault injection techniques to corrupt the instructions,
- Examine other types of corruption of instructions (`XOR`->Shift, ...) and try to define new challenging theoretical problems.

Backup slides

Micro-controller implementations of code-based cryptosystems

Existing work done in the 2010's[13,14].

Recent work too, since ARM Cortex-M4 became the de facto standard for this kind of embedded system implementations[15,16]

[13] S. Heyse. "Low-Reiter: Niederreiter Encryption Scheme for Embedded Microcontrollers". In: *International Workshop on Post-Quantum Cryptography*. Ed. by N. Sendrier. Vol. 6061. Lecture Notes in Computer Science. Darmstadt, Germany: Springer, May 2010, pp. 165–181.

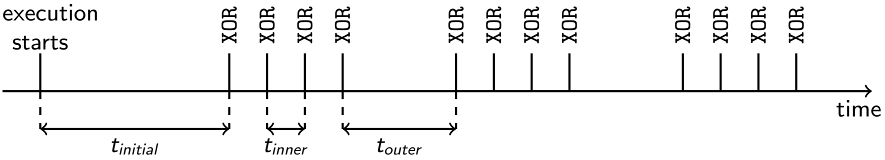
[14] T. Eisenbarth, T. Güneysu, S. Heyse, and C. Paar. "MicroEliece: McEliece for Embedded Devices". In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Ed. by C. Clavier and K. Gaj. Vol. 5747. Lecture Notes in Computer Science. Lausanne, Switzerland: Springer, Sept. 2009, pp. 49–64.

[15] M.-S. Chen and T. Chou. "Classic McEliece on the ARM Cortex-M4". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2021.3 (2021), pp. 125–148.

[16] J. Roth, E. G. Karatsiolis, and J. Krämer. "Classic McEliece Implementation with Low Memory Footprint". In: *International Conference on Smart Card Research and Advanced Application*. Ed. by P.-Y. Liardet and N. Mentens. Vol. 12609. Lecture Notes in Computer Science. Virtual Event: Springer, Nov. 2020, pp. 34–49.

Fault injection timing

Constant-time code makes fault injection much easier.



3 delays to tune:

- $t_{initial}$
- t_{inner}
- t_{outer}