Algebraic Key-Recovery Side-Channel Attack on *Classic McEliece*

Michaël Bulois[1], Pierre-Louis Cayrel[2], Vlad-Florin Drăgoi[3,4], and Vincent Grosso[2]

[1] Université Jean Monnet, CNRS, Centrale Lyon, INSA Lyon, Universite Claude Bernard Lyon 1, ICJ UMR 5208, 42023 Saint-Etienne, France
michael.bulois@univ-st-etienne.fr
[2] Université Jean Monnet Saint-Etienne, CNRS, Institut d Optique Graduate School, Laboratoire Hubert Curien UMR 5516, 42023 Saint-Etienne, France,
{pierre.louis.cayrel; vincent.grosso}@univ-st-etienne.fr
[3] Faculty of Exact Sciences, Aurel Vlaicu University, Arad, Romania,
[4] LITIS, University of Rouen Normandie, Saint-Etienne du Rouvray, France
vlad.dragoi@uav.ro

Abstract. The ongoing standardization of post-quantum cryptography (PQC) by NIST has renewed attention to code-based cryptosystems, notably *Classic McEliece*, due to their long-standing resistance to cryptanalysis and strong security against quantum adversaries.
However, implementation-level vulnerabilities such as side-channel leakage remain a critical threat.
In this work, we present a novel algebraic side-channel attack targeting the decapsulation phase of *Classic McEliece*, specifically when implemented using the reference decoding algorithm based on Berlekamp–Massey and matrix-vector multiplications. Our approach exploits Hamming weight leakage observed during syndrome computations and combines linear algebra techniques over finite fields with bit-level side-channel information. The proposed method significantly improves over previous attacks in terms of computational efficiency and robustness to noise, remaining effective even under realistic leakage inaccuracies. We provide both a theoretical analysis and a probabilistic model characterizing the success rate in noisy conditions.
Our results also contribute to the theoretical understanding of algebraic distinguishability of field elements from side-channel leakage, partially validating previously stated conjectures. Experimental simulations confirm the practicality of the attack, including for large parameter sets associated with high security levels.
Importantly, the attack specifically exploits leakage arising during the matrix–vector multiplication used to compute the double syndrome in the *Classic McEliece* reference implementation. The attack does not apply if the double syndrome is computed by other means (for example, using transposed additive FFTs), or if a different decoding algorithm is used that does not require computing the double syndrome at all. This highlights that informed algorithmic choices can serve as effective countermeasures and underscores the urgent need for rigorous side-channel

evaluations and secure-by-design implementations of code-based cryptosystems.

Keywords: · Post-quantum cryptography · Code-based cryptography · Classic McEliece · Side-channel attacks · Linear Algebra

1 Introduction

The NIST Post-Quantum Cryptography (PQC) standardization initiative, launched in 2017, has stimulated intense research into quantum-resistant public-key cryptosystems. In 2024, the lattice-based scheme CRYSTALS-Kyber [SAB+20], now standardized under the name ML-KEM, was selected as the primary algorithm for key encapsulation mechanisms (KEMs). For digital signatures, NIST retained Falcon and CRYSTALS-Dilithium—also lattice-based—alongside SPHINCS+, a hash-based scheme, to diversify cryptographic assumptions.

To further expand this diversity, NIST also evaluated code-based candidates. Among these, HQC [AAB+22] was selected for standardization. Conversely, *Classic McEliece* [ABC+22], despite its long-standing resistance to cryptanalysis, was excluded from immediate standardization by NIST. According to NIST, its ongoing parallel evaluation by the International Organization for Standardization (ISO) raised concerns about potential standard fragmentation. Its inclusion may be reconsidered following the conclusion of the ISO process. In the meantime, attention remains focused on practical concerns such as implementation efficiency and resistance to physical attacks.

In this work, we focus on *Classic McEliece*, a code-based KEM whose security relies on the hardness of decoding random linear codes—specifically, Goppa codes. Despite its name, *Classic McEliece* is built upon the Niederreiter cryptosystem [Nie86], and has withstood decades of cryptanalytic scrutiny.

However, as with many public-key schemes, *Classic McEliece* is vulnerable to side-channel attacks, which exploit physical information leaked during execution, such as timing, power consumption, or electromagnetic emissions. Code-based cryptosystems have been known to exhibit such leakage since well before the PQC standardization effort [HMP10,MSSS11,AHPT11,CEvMS16]. Over time, various countermeasures—e.g., masking, constant-time execution—have been proposed to mitigate these threats.

Following the PQC standardization efforts, software/hardware implementations of *Classic McEliece* have been developed [CC21,CCD+22,SKE+23,FLZG24]. These implementations aim to balance performance and security across diverse platforms, ranging from constrained embedded systems to high-throughput environments. Ensuring their resilience to physical attacks remains essential for secure deployment.

Scope and Limitations. This work targets a specific class of side-channel vulnerabilities in *Classic McEliece* implementations that rely on Berlekamp–Massey decoding, as found in the official reference software. In this setting, the attack

exploits leakage arising during the matrix–vector multiplication used to compute the double syndrome. Consequently, the attack does not apply if the double syndrome is computed by other means (for example, using transposed additive FFTs [CC21], which is also used in the official optimized Classic McEliece implementations), or if a different decoding algorithm is used that does not require computing the double syndrome at all. For example the public syndrome could be decoded using Patterson algorithm, hence avoiding the re-computation of a double-syndrome. This underlines the critical role of implementation choices in achieving side-channel resistance.

1.1 Notations

We aim to remain as close as possible to the notations introduced in the Round-4 *Classic McEliece* design document [ABC+22].

Integers are denoted by lowercase letters. Sets are represented using calligraphic capital letters, e.g., $\mathcal{A}$, and we denote the cardinality of a set $\mathcal{A}$ by $\#\mathcal{A}$. A set of consecutive integers is denoted by $[\![n; m]\!] = \{i \in \mathbb{Z} \mid n \leq i \leq m\}$.

We denote by $\mathbb{F}_q$ the finite field of order q. Elements of $\mathbb{F}_q$ are represented by lowercase Greek letters, e.g., $\alpha \in \mathbb{F}_q$. Polynomials over $\mathbb{F}_q$ are denoted using bold Greek letters.

Bold lowercase letters are used to denote vectors, e.g., $\boldsymbol{v}$, while bold uppercase letters denote matrices, e.g., $\boldsymbol{H}$. For a matrix $\boldsymbol{H}$, we denote by h_{ij} the entry located in the i^{th} row and j^{th} column. We denote the i^{th} row and j^{th} column of $\boldsymbol{H}$ by $\boldsymbol{H}[i,:]$ and $\boldsymbol{H}[:,j]$, respectively. A submatrix of $\boldsymbol{H}$ indexed by a set of rows $\mathcal{I}$ and columns $\mathcal{J}$ is written $\boldsymbol{H}[\mathcal{I}, \mathcal{J}]$.

The support of a vector $\boldsymbol{v}$ is defined as $\mathrm{supp}(\boldsymbol{v}) = \{i \mid v_i \neq 0\}$. The notion of support can also be extended to lists of integers or other indexed, compatible data structures.

Definition 1 (Finite field representation). *Let m be an integer and $\boldsymbol{\zeta} \in \mathbb{F}_2[X]$ be an irreducible polynomial of degree m. An element $\alpha \in \mathbb{F}_{2^m}$ is represented as an element $\boldsymbol{\alpha} = (\alpha_0, \ldots, \alpha_{m-1})$ of $\mathbb{F}_2^m$ via a fixed isomorphism $\mathbb{F}_{2^m} \cong \mathbb{F}_2[X]/(\boldsymbol{\zeta})$: $\alpha = \sum_{i=0}^{m-1} \alpha_i x^i$ where $x = \overline{X}$ is the class of X in $\mathbb{F}_2[X]/(\boldsymbol{\zeta})$ thus yielding an identification $\mathbb{F}_{2^m} \cong (\mathbb{F}_2)^m$.*

In particular, $\mathbb{F}_{2^m}$ is an $\mathbb{F}_2$-vector space with a distinguished basis $B_{can} = B_{can,\boldsymbol{\zeta}} = (x^i)_{i \in [\![0,m-1]\!]}$.

Definition 2 (Hamming weight). *Let $\boldsymbol{c} \in \mathbb{F}_q^n$. Its Hamming weight is $\mathsf{wt}(\boldsymbol{c}) = \#\mathrm{supp}(\boldsymbol{c})$. For $\alpha \in \mathbb{F}_{2^m}$ we define its Hamming weight by $\mathsf{wt}(\alpha) \stackrel{\mathsf{def}}{=} \mathsf{wt}(\boldsymbol{\alpha})$, where $\boldsymbol{\alpha} \in (\mathbb{F}_2)^m$ is the binary extension of α.*

Definition 3 (Binary Hamming weight). *Let $\mathsf{wt}_2 : \mathbb{F}_{2^m} \to \mathbb{F}_2$ be defined by $\mathsf{wt}_2(\alpha_0, \ldots, \alpha_{m-1}) = \sum_{i=0}^{m-1} \alpha_i$ where the sum is understood in $\mathbb{F}_2$. It is a linear form of $\mathbb{F}_{2^m}$, i.e. an element of the dual $\mathbb{F}_{2^m}^*$.*

The binary Hamming weight appears to be a natural candidate as a linear form that can be derived from side-channel information.

Definition 4 (Code). *A code $\mathscr{C}$ of length n is a subset of $\mathbb{F}_q^n$. An element $\boldsymbol{c} = (c_0, \ldots, c_{n-1}) \in \mathscr{C}$ is called a codeword. If $\mathscr{C}$ is a vector subspace of dimension k of the vector space $\mathbb{F}_q^n$, then $\mathscr{C}$ is a linear code of length n and dimension k. Such a linear code is referred to as an $[n, k]_q$ code.*

Definition 5 (Parity-check matrix). *Let $\mathscr{C}$ be an $[n, k]_q$ linear code, and let $\boldsymbol{H}$ be an $(n - k) \times n$ matrix such that*

$$\forall \boldsymbol{c} \in \mathscr{C}, \quad \boldsymbol{H}\boldsymbol{c}^T = 0.$$

Then, $\boldsymbol{H}$ is a parity-check matrix for $\mathscr{C}$. Let $\boldsymbol{v} \in \mathbb{F}_q^n$. The syndrome of the vector $\boldsymbol{v}$ with respect to $\boldsymbol{H}$ is defined as $\boldsymbol{s} = \boldsymbol{H}\boldsymbol{v}^T$.

Definition 6 (Goppa code). *Let $\mathcal{L} = \{\alpha_0, \ldots, \alpha_{n-1}\}$, where $\alpha_i \in \mathbb{F}_q$ and $\alpha_i \neq \alpha_j$ for $i \neq j$. Let $\boldsymbol{\gamma}(x) \in \mathbb{F}_q[x]$ with $\deg(\boldsymbol{\gamma}) = t$, such that $\forall \alpha \in \mathcal{L}, \boldsymbol{\gamma}(\alpha) \neq 0$. The set*

$$\mathscr{G}(\boldsymbol{\gamma}, \mathcal{L}) = \left\{ \boldsymbol{c} = (c_0, c_1, \ldots, c_{n-1}) \in \mathbb{F}_2^n : \sum_{i=0}^{n-1} \frac{c_i}{x - \alpha_i} \equiv 0 \text{ in } \mathbb{F}_q[x]/(\boldsymbol{\gamma}(x)) \right\}$$

defines a Goppa code with parameters $\mathcal{L}$ and $\boldsymbol{\gamma}(x)$. Additionally, let $\boldsymbol{H}$ be the $t \times n$ matrix defined as

$$\boldsymbol{H} = \begin{pmatrix} \boldsymbol{\gamma}(\alpha_0)^{-1} & \ldots & \boldsymbol{\gamma}(\alpha_{n-1})^{-1} \\ \vdots & \ddots & \vdots \\ \alpha_0^{t-1}\boldsymbol{\gamma}(\alpha_0)^{-1} & \ldots & \alpha_{n-1}^{t-1}\boldsymbol{\gamma}(\alpha_{n-1})^{-1} \end{pmatrix}.$$

Its expansion over $\mathbb{F}_2$, an $mt \times n$ matrix, serves as a parity-check matrix for $\mathscr{G}(\boldsymbol{\gamma}, \mathcal{L})$.

1.2 The *Classic McEliece* KEM

Classic McEliece [ABC+22] is a Key Encapsulation Mechanism (KEM) based on the Niederreiter cryptosystem using Goppa codes.

In Table 1, we present the parameter sets for *Classic McEliece* and *toyeliece*, a smaller configuration intended for research and experimentation. In the ISO proposal,[5] the authors refined the parameter list and restricted it to the case $m = 13$. In particular, the irreducible polynomial $\boldsymbol{\zeta}$ used to define the extension field $\mathbb{F}_{2^m}$ is given by

$$\boldsymbol{\zeta}(x) = x^{13} + x^4 + x^3 + x + 1.$$

Classic McEliece [ABC+22] consists of three main algorithms:

[5] `https://classic.mceliece.org/iso-mceliece-20230419.pdf`

Table 1: *Classic McEliece* [ABC+22] and `toyeliece` [BCM+25] parameter sets.

Parameter Set	toyeliece 51220	mceliece 348864	mceliece 460896	mceliece 6688128	mceliece 6960119	mceliece 8192128
m	9	12	13	13	13	13
n	512	3488	4608	6688	6960	8192
t	20	64	96	128	119	128

KeyGen : The key generation process selects a random permuted support $\mathcal{L}$, consisting of n distinct elements from $\mathbb{F}_{2^m}$, and an irreducible monic polynomial $\boldsymbol{\gamma}$ of degree t. Together, $(\boldsymbol{\gamma}, \mathcal{L})$ form the secret key sk, which is used to compute the private parity-check matrix $\boldsymbol{H}_{\text{priv}}$. This matrix is then reduced to systematic form, yielding the public matrix $\boldsymbol{H}_{\text{pub}} = (\mathbf{I}_{mt} \,|\, \mathbf{T})$.

Encap : A random error vector $\boldsymbol{e} \in \mathbb{F}_2^n$ of Hamming weight t is sampled, and the ciphertext is computed as $\boldsymbol{z} = \boldsymbol{H}_{\text{pub}}\boldsymbol{e}^T$. The session key K is derived by hashing $\boldsymbol{e}$ and $\boldsymbol{z}$.

Decap : The error vector $\boldsymbol{e}$ is recovered from the ciphertext $\boldsymbol{z}$ using knowledge of $\boldsymbol{\gamma}$ and $\mathcal{L}$. The session key K is then derived by hashing $\boldsymbol{e}$ and $\boldsymbol{z}$.

In this article, we focus on the decapsulation algorithm, presented in Algorithm 1. In this step, the owner of the private key computes a syndrome polynomial using the Vandermonde structure of the Goppa code to decode the ciphertext $\boldsymbol{z}$ and derive the session key K.

Algorithm 1 The decapsulation algorithm of the *Classic McEliece* KEM

Input: Ciphertext $\boldsymbol{z}$ and private key $\mathsf{sk} = (\boldsymbol{\gamma}, \mathcal{L})$
Output: Session key K

1: Compute the padded vector $\boldsymbol{v} = (\boldsymbol{z}, 0, \ldots, 0)$ of length n
2: Construct the matrix:

$$\boldsymbol{H}_{\text{priv}_{\boldsymbol{\gamma}^2}} = \begin{pmatrix} \boldsymbol{\gamma}(\alpha_0)^{-2} & \cdots & \boldsymbol{\gamma}(\alpha_{n-1})^{-2} \\ \vdots & \ddots & \vdots \\ \alpha_0^{2t-1}\boldsymbol{\gamma}(\alpha_0)^{-2} & \cdots & \alpha_{n-1}^{2t-1}\boldsymbol{\gamma}(\alpha_{n-1})^{-2} \end{pmatrix}$$

3: Compute the syndrome: $\boldsymbol{s} = \boldsymbol{H}_{\text{priv}_{\boldsymbol{\gamma}^2}}\boldsymbol{v}^T$
4: Use the Berlekamp–Massey algorithm to compute the error locator polynomial $\sigma(x)$
5: Evaluate $\sigma(\alpha_0), \ldots, \sigma(\alpha_{n-1})$ for $\alpha_i \in \mathcal{L}$ and recover the error vector $\boldsymbol{e}$
6: Compute $K = \mathsf{hash}(1 \| \boldsymbol{e} \| \boldsymbol{z})$
7: **return** K

In the remainder of this work, we focus on Step 3 of the above algorithm, namely the syndrome computation. To simplify notation, we define $\boldsymbol{\gamma}(\alpha_i)^{-2} = \beta_i$ for $0 \leq i < n$.

1.3 Related Work

Side-Channel Scenario. Key-recovery side-channel attacks targeting both the reference and optimized implementations of *Classic McEliece* have been recently demonstrated [GJJ22,SCD+23,BCM+25,DCV+25]. Among these, several attacks focus on leakage during the syndrome computation step of the decapsulation process, each employing different strategies to address classifier accuracy and leakage modeling.

As shown in [DCV+25], the syndrome computation (see Step 3 of Algorithm 1) leaks side-channel information—specifically, power consumption traces—that enable an adversary to observe the following matrix:

$$\boldsymbol{H}_{\mathsf{wt}} = \begin{pmatrix} \mathsf{wt}(\beta_0) & \cdots & \mathsf{wt}(\beta_{n-1}) \\ \mathsf{wt}(\alpha_0\beta_0) & \cdots & \mathsf{wt}(\alpha_{n-1}\beta_{n-1}) \\ \vdots & \ddots & \vdots \\ \mathsf{wt}(\alpha_0^{2t-1}\beta_0) & \cdots & \mathsf{wt}(\alpha_{n-1}^{2t-1}\beta_{n-1}) \end{pmatrix}$$

Experimental results demonstrate that this leakage can be used to recover a sufficient number of pairs (α_i, β_i). The authors then formulate the following conjecture:

Conjecture 1 ([DCV+25]). For almost all degree-m monic irreducible polynomials $\boldsymbol{\zeta} \in \mathbb{F}_2[x]$, the extension field $\mathbb{F}_{2^m} \cong \mathbb{F}_2[X]/(\boldsymbol{\zeta})$ is such that almost all pairs $(\alpha, \beta) \in \mathbb{F}_{2^m}^* \times \mathbb{F}_{2^m}^*$ can be uniquely determined from $\boldsymbol{H}_{\mathsf{wt}}$, provided that t is sufficiently large.

In the same work [DCV+25], the conjecture was originally formulated based on a smaller subset of the information. Specifically, the authors claimed that fewer than $2t$ rows (denoted $d_{m,t} < 2t$) are sufficient to distinguish the pairs (α_i, β_i).

We also consider a noisy setting, in which several entries of $\boldsymbol{H}_{\mathsf{wt}}$ are incorrect—typically deviating from the correct value by either $+1$ or -1. The probability that a given value is correct is referred to as the *accuracy*, commonly denoted by a. In the experiments conducted in [DCV+25] using the ChipWhisperer platform [OC14], the observed accuracy for Hamming weight measurements exceeds 0.96.

Exploitation of the Leakage. In [DCV+25], a side-channel attack targets the syndrome computation step to recover both $\boldsymbol{\gamma}$ and $\mathcal{L}$. While the authors demonstrate that their method can tolerate some classifier inaccuracies, the attack still requires relatively high accuracy (≥ 0.945), i.e., low-noise measurement environments. In practice, such accuracy may not always be achievable in noisier setups.

In [VCC+25], the authors investigate the robustness of side-channel attacks on *Classic McEliece* in the presence of side-channel estimation errors. They propose a simple yet effective algorithm capable of recovering a list of Hamming weights from noisy measurements, where each Hamming weight is affected by

an error of at most ± 1. Their algorithm achieves a high success rate in reconstructing the corresponding field elements, and notably, it requires only t out of n values (or equivalently mt values). However, the method is restricted to scenarios where the errors are bounded by ± 1.

In both studies, exhaustive search techniques are employed, resulting in high computational complexity. Table 2 summarizes the different attacks and the corresponding attack scenarios.

Table 2: Complexity comparison for the side-channel key-recovery attacks targeting the syndrome computation in the *Classic McEliece* KEM.

Article	**Scenario**	**Type of attack**	Accuracy
[DCV+25]	Erroneous Hamming weights $\Rightarrow (\alpha, \boldsymbol{\gamma}(\alpha))$ not recovered	Exhaustive search precomputation	$a > 0.945$
[VCC+25]	Error ± 1 on Hamming weights $\Rightarrow (\alpha, \boldsymbol{\gamma}(\alpha))$ recovered	Exhaustive search and decoding	$a > 0.81$
This article	Error ± 1 (or even higher[6]) on Hamming weights $\Rightarrow (\alpha, \boldsymbol{\gamma}(\alpha))$ recovered	$\mathcal{O}(m^3 t)$ complexity and decoding	$a > 0.74$

1.4 The Berlekamp-Massey Algorithm

The Berlekamp-Massey algorithm [Mas69] is an efficient method for determining the shortest linear feedback shift register (LFSR) that generates a given binary sequence. Originally developed for decoding BCH and Reed-Solomon codes, it has found widespread applications in coding theory and cryptanalysis.

In the context of this work, we use the algorithm to recover the minimal polynomial $\boldsymbol{\chi}(Y)$ that characterizes a sequence of Hamming weights modulo 2. This polynomial plays a central role in reconstructing secret key elements in our side-channel attacks on the *Classic McEliece* cryptosystem. By leveraging algebraic properties, the algorithm iteratively updates $\boldsymbol{\chi}(Y)$ based on discrepancies observed in the sequence, ensuring an optimal reconstruction. For completeness, the Berlekamp-Massey algorithm is included in the Section A (Algorithm 5).

1.5 Contributions

In this work, we revisit side-channel key-recovery attacks targeting the reference implementation of *Classic McEliece* during the decapsulation phase. Building on the same threat model as in [VCC+25], we propose a novel attack that significantly improves upon previous approaches [VCC+25,DCV+25], both in terms of

[2] see Remark 5

computational efficiency and resilience to estimation errors in the side-channel leakage.

Our contribution is threefold:

- **Improved robustness and efficiency.** We introduce a new algebraic attack that tolerates realistic noise levels in the leakage and exhibits much lower computational complexity, especially for high values of the extension degree m. Unlike prior approaches, our method scales well and remains practical for large parameter sets relevant to post-quantum security.
- **Algebraic framework for noisy leakage.** Our analysis offers a new perspective on the effectiveness of previous attacks by establishing an algebraic link between the Hamming weights of intermediate sensitive variables and the corresponding secret field elements. This connection holds even under noisy side-channel observations. The framework we develop generalizes to any $\mathbb{F}_2$-linear leakage function, although our results suggest that the parity of the Hamming weight is the most natural and realistic leakage function in practice.
- **Theoretical insights and generalization.** Our results deepen the understanding of Conjecture 1 from [DCV+25], showing that distinguishing field elements based on side-channel information remains feasible even when the exploited leakage is strictly less informative than in previous attacks. In particular, we explain why primitive elements of $\mathbb{F}_{2^m}$ can be efficiently distinguished in the noise-free case, and provide evidence supporting this behavior under realistic assumptions.

At the core of our approach is an efficient algorithm that operates on $\boldsymbol{H}_{\mathsf{wt}_2}$, the reduction modulo 2 of the matrix of Hamming weights observed during syndrome computation. This allows for accurate recovery of the pairs $(\alpha, \boldsymbol{\gamma}(\alpha))$ with high probability, even in the presence of noise, and with complexity orders of magnitude lower than previous methods.

1.6 Organization

This article is organized as follows.

In Section 2, we study secret reconstruction from Hamming weight leakage in a noise-free setting, covering algebraic preliminaries, the reconstruction algorithm, theoretical justification, and complexity analysis.

Section 3 extends this to noisy environments, presenting decoding algorithms and analyzing success probabilities.

Section 4 discusses applications to other code-based cryptosystems.

Finally, subsection 4.3 addresses implementation considerations and proposes countermeasures, highlighting decoding methods and general protections against side-channel leakage.

2 Recovering $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ from $\widetilde{\boldsymbol{H}}_{\mathsf{wt}}$: algebraic method

2.1 Algebraic preliminaries

Let $\mathbb{K}$ be a field, V a vector space, and h a linear endomorphism of V (i.e., a $\mathbb{K}$-linear map $h : V \to V$). In the context of this paper, we have $\mathbb{K} = \mathbb{F}_2$ and $V = \mathbb{F}_{2^m}$ or its dual space $\mathbb{F}_{2^m}^* = \{\varphi : V \to \mathbb{F}_2 \mid \varphi \text{ linear}\}$.

The following hypothesis on the endomorphism h will be frequently used in what follows:

$$\text{The characteristic polynomial } \boldsymbol{\chi}_h \text{ of } h \text{ is irreducible.} \tag{H_1}$$

Lemma 1. *Assuming hypothesis* (H_1)*, there are no nontrivial* h*-stable subspaces of* V*. In other words, if* $W \subseteq V$ *satisfies* $h(W) \subseteq W$*, then either* $W = \{0\}$ *or* $W = V$*.*

Proof. Let $W \subseteq V$ be an h-stable subspace and choose a basis B of V obtained by extending a basis of W. The matrix of h relative to B then has a block upper-triangular form:

$$\begin{pmatrix} h_{|W} & * \\ 0 & h_{V/W} \end{pmatrix}.$$

Consequently, the characteristic polynomial factors as

$$\boldsymbol{\chi}_h = \boldsymbol{\chi}_{h_{|W}} \times \boldsymbol{\chi}_{h_{V/W}}.$$

Since $\boldsymbol{\chi}_h$ is irreducible by hypothesis, one of these factors must be trivial. Hence, either $\dim W = 0$ (i.e., $W = \{0\}$) or $\dim V/W = 0$ (i.e., $W = V$). □

In the following, the notation h^i denotes the i-th power of the endomorphism h, i.e., $h^2(\boldsymbol{v}) = h(h(\boldsymbol{v}))$.

Corollary 1. *Assume that* h *satisfies* (H_1)*. Then, for any nonzero vector* $\boldsymbol{v} \in V \setminus \{0\}$*, the family*

$$(\boldsymbol{v}, h(\boldsymbol{v}), h^2(\boldsymbol{v}), \ldots, h^{\dim V - 1}(\boldsymbol{v}))$$

forms a basis of V*.*

Moreover, if $\dim V \geq 2$*, then for any integer* $i \in \mathbb{Z}$*, the family*

$$(h^i(\boldsymbol{v}), h^{i+1}(\boldsymbol{v}), \ldots, h^{i+\dim V - 1}(\boldsymbol{v}))$$

is also a basis of V*.*

Proof. Consider the subspace

$$W = \langle h^i(\boldsymbol{v}) \mid i \in \mathbb{N} \rangle,$$

the smallest h-stable subspace of V containing $\boldsymbol{v}$. Since V is finite-dimensional, there exists a minimal integer n such that

$$h^n(\boldsymbol{v}) \in \langle \boldsymbol{v}, h(\boldsymbol{v}), \ldots, h^{n-1}(\boldsymbol{v}) \rangle.$$

Thus,

$$W = \langle \boldsymbol{v}, h(\boldsymbol{v}), \dots, h^{n-1}(\boldsymbol{v}) \rangle.$$

By Lemma 1, the only h-stable subspaces of V are $\{0\}$ and V itself, and since $\boldsymbol{v} \neq 0$, it follows that $W = V$. Hence,

$$(\boldsymbol{v}, h(\boldsymbol{v}), \dots, h^{n-1}(\boldsymbol{v}))$$

is a spanning set of V. By minimality of n, these vectors are linearly independent, so $n = \dim V$, proving the first assertion.

Now assume $\dim V \geq 2$. If $\ker(h)$ contains a nonzero vector $\mathbf{w}$, then $\mathbb{K}\mathbf{w}$ is a nontrivial h-stable subspace of dimension 1, contradicting Lemma 1. Hence, $\ker(h) = \{0\}$, so h is invertible. Therefore, $h^i(\boldsymbol{v})$ is well-defined and nonzero for all $i \in \mathbb{Z}$. The second assertion then follows directly from the first by applying powers of h or h^{-1}.

2.2 Linear algebra related to the sequence $(\mathsf{wt}_2(\alpha^i\beta))_i$

Let $\alpha, \beta \in \mathbb{F}_{2^m}$. For any integer $i \in \mathbb{N}$, the element $\alpha^i\beta \in \mathbb{F}_{2^m}$ has a Hamming weight modulo 2 denoted by $\mathsf{wt}_2(\alpha^i\beta) \in \mathbb{F}_2$. In this subsection, we investigate the algebraic properties of the sequence

$$\mathcal{W}_{\alpha,\beta} := \big(\mathsf{wt}_2(\alpha^i\beta)\big)_{i\in\mathbb{N}},$$

which takes values in $\mathbb{F}_2$.

Our analysis relies on the following fundamental observations:

- The map $\mathsf{wt}_2 : \mathbb{F}_{2^m} \to \mathbb{F}_2$ is an $\mathbb{F}_2$-linear form,
- Multiplication by α defines a linear endomorphism $h_\alpha : \mathbb{F}_{2^m} \to \mathbb{F}_{2^m}$,
- This endomorphism induces a dual endomorphism $h_\alpha^* : \mathbb{F}_{2^m}^* \to \mathbb{F}_{2^m}^*$ on the dual space given by

$$h_\alpha^*(\varphi)(x) = \varphi(\alpha x), \quad \forall \varphi \in \mathbb{F}_{2^m}^*,\ x \in \mathbb{F}_{2^m}.$$

Remark 1. If B is a basis of $\mathbb{F}_{2^m}$ and B^* its dual basis, then the matrix representation of h_α^* in B^* satisfies

$$\mathrm{Mat}_{B^*}(h_\alpha^*) = \mathrm{Mat}_B(h_\alpha)^T.$$

In particular, the irreducibility condition (H_1) on h_α is equivalent to the same condition on h_α^*.

Lemma 2. *The hypothesis (H_1) on h_α is equivalent to the condition $\mathbb{F}_2[\alpha] = \mathbb{F}_{2^m}$.*

Proof. If $\mathbb{F}_2[\alpha] = \mathbb{F}_{2^m}$, then α is a root of an irreducible polynomial $\boldsymbol{\chi}_\alpha \in \mathbb{F}_2[Y]$ of degree m. In the basis $(1, \alpha, \alpha^2, \dots, \alpha^{m-1})$, the matrix of the endomorphism h_α is the companion matrix of $\boldsymbol{\chi}_\alpha$. In particular, the characteristic polynomial $\boldsymbol{\chi}_{h_\alpha}$ of h_α coincides with $\boldsymbol{\chi}_\alpha$, which is irreducible.

Conversely, if $\mathbb{F}_2[\alpha] \neq \mathbb{F}_{2^m}$, then the set $W := \mathbb{F}_2[\alpha]$ is a nontrivial h_α-stable subspace of $\mathbb{F}_{2^m}$. The result then follows from Lemma 1. □

From now on, we assume hypothesis (H_1) for α with $m \geq 2$, and denote by

$$\boldsymbol{\chi}_\alpha = Y^m + \sum_{\ell=0}^{m-1} c_\ell Y^\ell \in \mathbb{F}_2[Y]$$

the irreducible monic polynomial of degree m with α as a root. As noted in the proof, it coincides with the characteristic polynomial of h_α (and of h_α^*).

Remark 2. The sequence $\mathcal{W}_{\alpha,\beta}$ can thus be viewed as the output of a linear feedback shift register (LFSR) over $\mathbb{F}_2$ with feedback polynomial $\boldsymbol{\chi}_\alpha$ (or $\boldsymbol{\chi}_{\alpha^{-1}}$, depending on convention). This directly follows from the recurrence relation

$$\alpha^{i+m} = \sum_{\ell=0}^{m-1} c_\ell \alpha^{i+\ell}$$

and the linearity of wt_2.

We also introduce the family $\varphi_\alpha[i] := (h_\alpha^*)^i(\mathsf{wt}_2) \in \mathbb{F}_{2^m}^*$ for $i \in \mathbb{N}$, and note that by construction,

$$\varphi_\alpha[i](\beta) = \mathsf{wt}_2(\alpha^i \beta), \tag{1}$$

which corresponds to the i-th term of the sequence $\mathcal{W}_{\alpha,\beta}$.

From Corollary 1, we deduce that

$$B_\alpha^* := (\varphi_\alpha[0], \ldots, \varphi_\alpha[m-1])$$

is a basis of $\mathbb{F}_{2^m}^*$.

Let B_α be the basis of $\mathbb{F}_{2^m}$ dual to B_α^*. By definition, the coordinates of an element $\delta \in \mathbb{F}_{2^m}$ in this basis are given by

$$Mat_{B_\alpha}(\delta) = \big(\varphi_\alpha[0](\delta), \ldots, \varphi_\alpha[m-1](\delta)\big)^T. \tag{2}$$

When applied to $\delta = \beta$, equations (2) and (1) imply that the coordinates of β in the basis B_α form the initial state (seed) of the LFSR described in Remark 2.

To relate the coordinates of an element $\delta \in \mathbb{F}_{2^m}$ in the two bases B_α and B_{can} (see Definition 1), we use the change-of-basis formula:

$$Mat_{B_\alpha}(\delta) = Mat_{B_\alpha}(B_{can}) \times Mat_{B_{can}}(\delta). \tag{3}$$

2.3 Noise-free setting

Thanks to the results from subsection 2.2, we now have all the necessary tools to build an algorithm that reconstructs pairs (α_k, β_k) from the sequence of Hamming weights $\boldsymbol{H}_{\mathsf{wt}}[:k]$.

More precisely, let $\alpha, \beta \in \mathbb{F}_{2^m}$ such that $\mathbb{F}_2[\alpha] = \mathbb{F}_{2^m}$ and $\beta \neq 0$. For any $i \in [\![0, 2t]\!]$, define

$$w_i = \mathsf{wt}(\alpha^i \beta) \in [\![0, m]\!] \quad \text{and} \quad \bar{w}_i = \mathsf{wt}_2(\alpha^i \beta) \in \mathbb{F}_2,$$

so that $(\bar{w}_i)_i$ are the first $2t$ terms of the sequence $\mathcal{W}_{\alpha,\beta}$.

Then:

- algorithms, such as Berlekamp-Massey, allow us to recover the minimal polynomial generating a linearly recurrent sequence. By Remark 2, this lets us reconstruct the polynomial $\boldsymbol{\chi}_\alpha$ from the sequence $(\bar{w}_i)_{i\in[\![0,2m]\!]}$. The condition $\mathbb{F}_2[\alpha] = \mathbb{F}_{2^m}$ (equivalently (H_1) on h_α) corresponds to $\deg \boldsymbol{\chi}_\alpha = m$, which can be tested. If this test fails (which occurs only for very few choices of α), we abandon the reconstruction for this specific pair α, β.
- Knowing that α is a root of $\boldsymbol{\chi}_\alpha$ leaves exactly m possible candidates for α. Denote the roots of $\boldsymbol{\chi}_\alpha$ by $\alpha^{(0)}, \ldots, \alpha^{(m-1)}$.
- For each $\alpha^{(\ell)}$, we can compute

$$C_\ell = Mat_{B_{\alpha^{(\ell)}}}(B_{can}).$$

  Indeed, by equation (2), we have

$$C_\ell = \left(\mathsf{wt}_2((\alpha^{(\ell)})^i x^j)\right)_{i,j\in[\![0,m-1]\!]}.$$

  Since $\boldsymbol{\chi}_{\alpha^{(\ell)}} = \boldsymbol{\chi}_\alpha$ is irreducible of degree m, we have $\mathbb{F}_2[\alpha^{(\ell)}] = \mathbb{F}_{2^m}$, so $B_{\alpha^{(\ell)}}$ forms a basis (as shown for B_α in subsection 2.2). Therefore, C_ℓ is invertible.
- Whenever $\alpha = \alpha^{(\ell)}$, combining Equations (3), (2) and (1) yields

$$Mat_{B_{can}}(\beta) = C_\ell^{-1} \times (\bar{w}_0, \ldots, \bar{w}_{m-1})^T.$$

  For an arbitrary ℓ, we can always define and compute an element $\beta^{(\ell)} \in \mathbb{F}_{2^m}$ by setting

$$Mat_{B_{can}}(\beta^{(\ell)}) = C_\ell^{-1} \times (\bar{w}_0, \ldots, \bar{w}_{m-1})^T.$$

  Note that the different pairs $(\alpha^{(\ell)}, \beta^{(\ell)})$ thus obtained generate identical sequences $\mathcal{W}_{\alpha^{(\ell)},\beta^{(\ell)}}$ since these are outputs of an LFSR with the same minimal polynomial $\boldsymbol{\chi}_\alpha$ and the same initial state (seed)

$$Mat_{B_{\alpha^{(\ell)}}}(\beta^{(\ell)})^T = \left(C_\ell Mat_{B_{can}}(\beta^{(\ell)})\right)^T = (\bar{w}_0, \ldots, \bar{w}_{m-1})^T.$$

- However, we claim that the original Hamming weight sequences $\left(\mathsf{wt}((\alpha^{(\ell)})^i \beta^{(\ell)})\right)_i$ very likely distinguish the m candidate pairs. This allows us to identify a unique matching pair, which must then be (α, β). Supporting evidence for this claim is provided in Lemma 3.

The central algorithm of this subsection can thus be summarized in the following pseudo-code.

Algorithm 2 Constructive algorithm to find pairs.

Input: The matrix $\boldsymbol{H}_{\mathsf{wt}}$
Output: t good pairs $(\alpha_k, \beta_k)_{k\in\mathcal{I}}$ with $\#\mathcal{I} = t$

1: $\mathcal{I} = \emptyset$, $k = -1$ ▷ $\mathcal{I}$ is the set of good pairs, k the index of a column
2: **while** $\#\mathcal{I} \neq t$ **do**
3: $k = k + 1$ ▷ browse $\boldsymbol{H}_{\mathsf{wt}_2} = \boldsymbol{H}_{\mathsf{wt}} \pmod 2$ column-wise
4: Construct $\mathcal{W} = (\mathsf{wt}_2(\alpha_k^i \beta_k))_{i\in[\![0,2m-1]\!]}$ from column k of $\boldsymbol{H}_{\mathsf{wt}_2}$
5: $\boldsymbol{\chi} \leftarrow \mathrm{BM}(\mathcal{W})$ ▷ Apply Berlekamp-Massey's algorithm to $\mathcal{W}$ and store its generating polynomial
6: **if** $\deg(\boldsymbol{\chi}) \neq m$ **then** ▷ α_k does not satisfy (H_1)
7: **go to** 3 ▷ go to line 3 to take the next column
8: **else**
9: Compute the roots of $\boldsymbol{\chi}$: $\alpha^{(0)}, \ldots, \alpha^{(m-1)}$
10: $\mathcal{J} = \emptyset$ ▷ $\mathcal{J}$ stores the $(\alpha^{(\ell)}, \beta^{(\ell)})$ compatible with $\boldsymbol{H}_{\mathsf{wt}}[:k]$
11: **for** $\ell \leftarrow 0$ to $m-1$ **do** ▷ browse the roots of $\boldsymbol{\chi}$
12: Compute

$$C_\ell = \begin{pmatrix} \mathsf{wt}_2((\alpha^{(\ell)})^0 x^0) & \ldots & \mathsf{wt}_2((\alpha^{(\ell)})^0 x^{m-1}) \\ \mathsf{wt}_2((\alpha^{(\ell)})^1 x^0) & \ldots & \mathsf{wt}_2((\alpha^{(\ell)})^1 x^{m-1}) \\ \vdots & \ddots & \vdots \\ \mathsf{wt}_2((\alpha^{(\ell)})^{m-1} x^0) & \ldots & \mathsf{wt}_2((\alpha^{(\ell)})^{m-1} x^{m-1}) \end{pmatrix}, \quad W = \begin{pmatrix} \mathsf{wt}_2(\alpha_k^0 \beta_k) \\ \mathsf{wt}_2(\alpha_k^1 \beta_k) \\ \vdots \\ \mathsf{wt}_2(\alpha_k^{m-1} \beta_k) \end{pmatrix}$$

13: Compute $\beta^{(\ell)} = C_\ell^{-1} \times W$
14: **if** $(\mathsf{wt}((\alpha_k^{(\ell)})^i (\beta_k^{(\ell)})))_i = (\mathsf{wt}(\alpha_k^i \beta_k))_i$ **then** ▷ use of $\boldsymbol{H}_{\mathsf{wt}}$
15: $\mathcal{J} \leftarrow$ Append $(\alpha^{(\ell)}, \beta^{(\ell)})$
16: **if** $\#\mathcal{J} = 1$ **then** ▷ a single match among the m pairs of candidates
17: $\alpha_k, \beta_k = \mathcal{J}[0]$
18: $\mathcal{I} \leftarrow$ Append (α_k, β_k, k) ▷ (α_k, β_k, k) indicates that (α_k, β_k) generates column k

Remark 3. The first part of the algorithm could also be applied to various linear leakages other than wt_2. For instance,

- The "trace" operator is also $\mathbb{F}_2$-linear, but it might be difficult to obtain via physical attacks.
- Taking every single coordinate in $(\mathbb{F}_2)^m$ is $\mathbb{F}_2$-linear. This is particularly true for least significant bit leakage, which in some devices can have a greater impact on power consumption than other bits.

Note that with such individual leakage (and without data on the full Hamming weight wt), we cannot distinguish the m roots of a given polynomial $\boldsymbol{\chi}$ as done in line 14 and afterwards.

However, if one has access to two such leakages, each given wrong candidate $\alpha^{(\ell)}$ might give rise to different candidates $\beta^{(\ell)}$, $\beta^{(\ell)\prime}$ for each leakage, thus allowing to eliminate these candidates.

2.4 Theoretical sufficiency of the knowledge of $\mathbf{wt}(\boldsymbol{\alpha}^i\boldsymbol{\beta})_i$

We now provide evidence of why the condition in line 16 should often be satisfied.

Given an irreducible $\mathbb{F}_2$-polynomial $\boldsymbol{\zeta}$ of degree m (from which we construct $\mathbb{F}_{2^m}$ and its associated Hamming distance), we say that there is a *collision* between the distinct pairs (α, β) and (α', β') if the entire sequences $(\mathsf{wt}(\alpha^i\beta))_{i\in\mathbb{N}}$ and $(\mathsf{wt}((\alpha')^i\beta'))_{i\in\mathbb{N}}$ coincide. With the following lemma, we provide a proof for Conjecture 1.

Lemma 3. *Assume that $\boldsymbol{\zeta}$ is primitive. Then there is no collision with a pair (α, β) such that α is primitive.*

Remark 4.
- The condition that $\boldsymbol{\zeta}$ is primitive is necessary. Indeed, computations show that for some defining polynomials (such as $\sum_{i=0}^{12} X^i$ for $m = 12$) there are typically clusters of m collisions. From the discussion above Algorithm 2, these clusters must be made of the $(\alpha^{(\ell)}, \beta^{(\ell)})$ with $\ell \in [\![0, m-1]\!]$. However, for *Classic McEliece*, the primitivity condition on $\boldsymbol{\zeta}$ is always satisfied.
- The condition that α is primitive is not too restrictive. In the worst case for *Classic McEliece*, $m = 12$, among the 4020 elements α such that $\mathbb{F}_2[\alpha] = \mathbb{F}_{2^m}$, 1728 are primitive. For the ISO parameters ($m = 13$), except for $\alpha = 0$ and $\alpha = 1$, all elements are primitive.
- The lemma only tells us that the Hamming weight sequences discriminate the candidates at some integer index which may be beyond $2t$, the length of the acquired sequence. But, since two different sequences necessarily differ at some point, there is no reason for the bifurcation not to occur early on.
- Effective computations show that the Hamming weight sequence indeed discriminates the candidates early on.

Proof. of Lemma 3 We can assume $m \geqslant 2$.

Note that if α is primitive, then α satisfies hypothesis (H_1). It follows from the discussion preceding Algorithm 2 that, whenever there is a collision with a pair (α', β') then $\alpha \neq \alpha'$ and $\boldsymbol{\chi}_\alpha = \boldsymbol{\chi}_{\alpha'}$. Since the different roots of a given irreducible polynomial are related by iterations of the Frobenius automorphism $Fr(\delta) = \delta^2$, there exists some $i_0 \in [\![1, m-1]\!]$ such that $\alpha' = \alpha^{2^{i_0}}$.

Since α is primitive and $\beta \neq 0$, we have $\mathbb{F}_{2^m} \setminus \{0\} = \{\alpha^i\beta | i \in [\![0, 2^m - 2]\!]\}$. We are interested in the indices $t_0, \ldots, t_{m-1}$ satisfying $\alpha^{t_\ell}\beta = x^\ell$ ($\ell \in [\![0, m-1]\!]$). Since α is primitive, those indices are well-defined modulo $2^m - 1$. Moreover, $x = \frac{x^\ell}{x^{\ell-1}} = \frac{\alpha^{t_\ell}\beta}{\alpha^{t_{\ell-1}}\beta} = \alpha^{t_\ell - t_{\ell-1}}$ so $t_0, \ldots, t_{m-1}$ forms an arithmetic progression in $\mathbb{Z}/(2^m - 1)\mathbb{Z}$ with a common difference of $r := t_1 - t_0$. Since $\boldsymbol{\zeta}$ is primitive, x is a generator of $(\mathbb{F}_{2^m} \setminus \{0\}, \times)$ so r is invertible in $\mathbb{Z}/(2^m - 1)\mathbb{Z}$.

For (α', β'), the same holds, but we will only need to note that the indices $t'_0, \ldots, t'_{m-1}$ defined by $(\alpha')^{t'_\ell}\beta = x^\ell$ satisfy the equality $x = (\alpha')^{t'_\ell - t'_{\ell-1}}$.

We have chosen the indices t_ℓ since they are precisely the indices for which $\mathsf{wt}(\alpha^t\beta) = 1$. From the collision hypothesis, there exists a permutation σ of $[\![0, m-1]\!]$ such that for all ℓ, we have $t'_\ell = t_{\sigma(\ell)}$. From the relation between

α and α', we deduce that $\alpha^{2^{i_0}(t_{\sigma(\ell)}-t_{\sigma(\ell-1)})} = (\alpha')^{t'_\ell - t'_{\ell-1}} = x$. In particular, $(t_{\sigma(\ell)}2^{i_0})_\ell$ also forms an arithmetic progression in $\mathbb{Z}/(2^m-1)\mathbb{Z}$ with a common difference of r. This can also be reformulated as follows: $(t_{\sigma(\ell)})_\ell$ forms an arithmetic progression in $\mathbb{Z}/(2^m-1)\mathbb{Z}$ with a common difference of $r2^{m-i_0}$.

Let $c = \sigma(1) - \sigma(0) \in [\![-m+1, m-1]\!]$, then $rc = t_{\sigma(1)} - t_{\sigma(0)} = r2^{m-i_0}$. Similarly let $c' = \sigma^{-1}(1) - \sigma^{-1}(0) \in [\![-m+1, m-1]\!]$, then $rc' = 2^{i_0}(t_{\sigma(\sigma^{-1}(1))} - t_{\sigma(\sigma^{-1}(0))}) = 2^{i_0}r$. Combining these two equalities, we get the relation $rcc' = r2^{m-i_0}c' = 2^m r = r$. Since r is invertible, this yields $cc' = 1$ in $\mathbb{Z}/(2^m-1)\mathbb{Z}$.

In $\mathbb{Z}$, $-(m-1)^2 \leqslant cc' \leqslant (m-1)^2$. Since $(m-1)^2 < 2^m - 2$ $(m \geqslant 2)$, the only possibility to get $cc' = 1$ in $\mathbb{Z}/(2^m-1)\mathbb{Z}$ is that $cc' = 1$ in $\mathbb{Z}$. So $c, c' = \pm 1$. This contradicts the equality $c' = 2^{i_0}$ in $\mathbb{Z}/(2^m-1)\mathbb{Z}$ (deduced from $rc' = 2^{i_0}r$) since $i_0 \in [\![1, m-1]\!]$. □

2.5 Complexity

We analyze here the expected running time of Algorithm 2, which remains practical even for large values of m. The analysis assumes constant-time arithmetic over $\mathbb{F}_{2^m}$ and constant-time evaluation of $\mathsf{wt}_2(\cdot)$, which is a reasonable assumption given that m is relatively small (e.g., $m \leq 13$ in all *Classic McEliece* parameters).

The complexity analysis of our algorithm falls under the same hypothesis as [KM23] and [ABC+22] (Section 5 Selected parameter sets), more precisely, the code length $n \to \infty$ while $m = \mathcal{O}(\log_2(n))$ and $tm \approx \frac{n}{4}$.

Lemma 4. *The expected running time of Algorithm 2 is*

$$\mathcal{O}\left(\frac{(n\log_2(n))^2}{n_m}\right),$$

where n_m denotes the number of elements in the support $\mathcal{L}$ that satisfy the hypothesis (H_1). In particular, when $2^m - 1$ is a prime number, we have $n_m = n$ and the complexity simplifies to $\mathcal{O}(n(\log_2(n))^2)$.

Proof. Let us analyze the cost of a single iteration of the **while** loop in Algorithm 2, which attempts to recover a pair (α, β).

- **Berlekamp-Massey (Step 5):** Given a binary sequence of length $2m$, the Berlekamp-Massey algorithm runs in $\mathcal{O}(m^2)$ operations.
- **Root finding (Step 9):** We compute the m roots of a degree-m polynomial over $\mathbb{F}_{2^m}$. In characteristic 2, algorithms based on equal-degree factorization (e.g., via distinct-degree decomposition and root extraction) yield a cost of $\mathcal{O}(m^3)$ in practice.
- **Matrix construction (Step 12):** Computing the matrix $C_\ell \in \mathbb{F}_2^{m\times m}$ from the roots involves $\mathcal{O}(m^2)$ operations.
- **Linear system solving (Step 13):** Solving a linear system of size m over $\mathbb{F}_{2^m}$ has complexity $\mathcal{O}(m^c)$, where $c \leq 3$ is the exponent of matrix multiplication.

Overall, the complexity of a single iteration is dominated by the root-finding and linear system solving steps, and can be upper bounded by $\mathcal{O}(m^3)$.

Now, we estimate the expected number of iterations N_{it} needed to recover t valid (α, β) pairs. Let n_m be the number of support elements in $\mathcal{L}$ satisfying (H_1). At each iteration, an element from $\mathcal{L}$ is sampled uniformly at random. Assuming independence and no collisions (as suggested by Lemma 3), the probability that an iteration yields a useful pair is n_m/n.

To recover t correct pairs, the expected number of trials follows a geometric distribution with success probability n_m/n, giving:

$$N_{it} = t \cdot \frac{n}{n_m}.$$

Hence, the total complexity is $\mathcal{O}\left(\frac{(n \log_2(n))^2}{n_m}\right)$. In particular, if $2^m - 1$ is prime, then $\mathcal{L} = \mathbb{F}_{2^m}^*$, and $n = n_m$, so $N_{it} = t$ and the complexity simplifies to $\mathcal{O}(n(\log_2(n))^2)$.

□

Remark. The above estimate reflects practical costs under realistic parameters, with m fixed (typically $m \leq 13$). The asymptotic expression $\mathcal{O}\left(\frac{(n \log_2(n))^2}{n_m}\right)$ is therefore primarily indicative of behavior across increasing security levels. Our method remains efficient and usable even for extended variants of *Classic McEliece* with $m > 13$.

3 Improved error-correcting algorithm

3.1 Noisy setting

In [VCC+25], the authors proposed a more realistic model. Even in the noise-free scenario, such as the one in [DCV+25], the accuracy is not equal to 1.

As demonstrated by the DPA contest V3[7], the acquisition setting can drastically affect the success rate of an attack, or equivalently impact the accuracy of a distinguisher.

Indeed, in practice the accuracy of the Hamming weight distinguisher is less than 1, which is why they consider noisy estimations of the actual Hamming weight. The error on these estimations is defined by a categorical random variable. More precisely, an estimated weight sequence $\widetilde{\mathcal{W}}$ is a column vector of the noisy matrix $\widetilde{\boldsymbol{H}}_{\mathsf{wt}}$, defined as:

$$\widetilde{\boldsymbol{H}}_{\mathsf{wt}}[:, j] = \left(\mathsf{wt}(\alpha_j^i \beta_j) + \varepsilon_{i,j}\right)_{i \in [\![0; 2t-1]\!]},$$

with $\varepsilon_{i,j} \in \{0, 1, -1\}$. In other words, we can write $\widetilde{\mathcal{W}} = \mathcal{W} + \mathcal{E}$, where $\mathcal{W}$ is the correct Hamming weight sequence and $\mathcal{E}$ represents the error sequence. For a deeper understanding of the error model, we refer the reader to [GCCD23,VCC+25].

[7] `https://dpacontest.telecom-paris.fr/v3/index.php`

Notice that taking the mod 2 of the estimated weight sequence yields:

$$\widetilde{\mathcal{W}_2} = \mathcal{W}_2 + \mathcal{E}_2,$$

where $\mathcal{W}_2$ is the correct mod 2 weight sequence and $\mathcal{E}_2$ is the mod 2 error sequence with $e_i \in \{0, 1\}$. Typically: - $e_i = 1$ indicates an error at position i in $\mathcal{W}_2$ (i.e., $\varepsilon_i \in \{1, -1\}$), - $e_i = 0$ indicates an error-free position (i.e., $\varepsilon_i = 0$).

This leads us to the central question:

Can we recover the BM polynomial generating the sequence $\mathcal{W}_2$ from a noisy sequence $\widetilde{\mathcal{W}_2}$?

To answer this, we propose two solutions: Algorithm 3 and Algorithm 4. Before analyzing these methods in detail, we outline the main steps of our error correction strategy.

1. **Mod 2 error correction:**
   - Use Algorithm 3 or Algorithm 4 to determine a BM polynomial that is likely to be the original polynomial generating the LFSR $\mathcal{W}_2$.
   - Recover $\mathcal{E}_2$ and hence $\mathcal{W}_2$ using this polynomial.
2. **Recover potential pairs:**
   - Apply Algorithm 2 (from line 6 to line 13) to recover a list of candidate pairs (α, β) satisfying $\mathcal{W}_2(\alpha, \beta) = \mathcal{W}_2$, where:

$$\mathcal{W}_2(\alpha, \beta) \stackrel{\mathsf{def}}{=} \left(\mathsf{wt}_2(\alpha_k^i \beta_k)\right)_{i \in [\![0; 2t-1]\!]}.$$

3. **Compute valid pairs:** For each candidate (α, β), compute the sequence:

$$\mathcal{W}(\alpha, \beta) \stackrel{\mathsf{def}}{=} \left(\mathsf{wt}(\alpha_k^i \beta_k)\right)_{i \in [\![0; 2t-1]\!]},$$

   and retain the pairs satisfying the following conditions:
   - On the error-free positions, the Hamming weight must match exactly:

$$\mathcal{W}(\alpha, \beta)_{[\![0; 2t-1]\!] \setminus \mathrm{supp}(\mathcal{E}_2)} = \widetilde{\mathcal{W}}_{\mathrm{supp}(\mathcal{E}_2)^c}.$$

   - On error positions, the difference must be ± 1:

$$\left(\mathcal{W}(\alpha, \beta) - \widetilde{\mathcal{W}}\right)_{\mathrm{supp}(\mathcal{E}_2)} \in \{-1, +1\}^{\#\mathrm{supp}(\mathcal{E}_2)}.$$

   If no pair satisfies these conditions, then the BM polynomial found in Step 1 was likely not the correct one.

Remark 5. These techniques can be extended to other error models, where errors e_i on the Hamming weights lie in $[\![-c, c]\!]$ for some $c > 1$, with high probability for $e_i = 0$ and low probability that $|e_i| > 1$ (e.g., a discrete Gaussian-like model). This can be handled either by keeping Step 3 unchanged (thus rejecting sequences where $|e_i| > 1$), or by allowing a small number of discrepancies below a defined threshold.

Algorithm 3 Majority rule

Input: $\widetilde{\mathcal{W}_2}$ — noisy mod 2 Hamming weight sequence obtained from SCA
Output: Most probable $\boldsymbol{\chi}_k$ generating $\mathcal{W}_2$
1: $Polys \leftarrow [\,]$
2: **for** $i \leftarrow 0$ to $\lfloor \log_2(\frac{t}{m}) \rfloor$ **do**
3: $\quad s \leftarrow 2^i$
4: $\quad$ **for** $j \leftarrow 0$ to $2t - 2ms$ **do**
5: $\quad\quad w \leftarrow \widetilde{\mathcal{W}_2}[j : s : j + 2ms]$
6: $\quad\quad poly \leftarrow \mathrm{BM}(w)$
7: $\quad\quad$ **if** $\deg(poly) = m$ and $\mathrm{LSB}(poly) = 1$ **then**
8: $\quad\quad\quad Polys.\mathrm{append}(poly)$
return $\mathrm{argmax}_{p \in Polys}\ \#\{q \in Polys \mid q = p\}$ ▷ Most frequent polynomial

3.2 Algorithms for error correction

Algorithm 3 takes a noisy mod 2 Hamming weight sequence $\widetilde{\mathcal{W}_2}$ as input. It explores multiple sub-sequences of length $2m$ with various spacings (Line 2), applies the Berlekamp-Massey (BM) algorithm, and collects valid polynomials of degree m with least significant bit equal to 1 into the list $Polys$. The most frequent polynomial in this list is returned as the best candidate.

Algorithm 4 Sequence distance

Input: $\widetilde{\mathcal{W}_2}$ — noisy mod 2 Hamming weight sequence obtained from SCA
Output: Most probable BM polynomial $\boldsymbol{\chi}_k$ and corresponding denoised sequence $\mathcal{W}_2$
1: $poly\_saved \leftarrow 0$, $min\_error \leftarrow 2t + 1$
2: **for** $i \leftarrow 0$ to $2t - 2m$ **do**
3: $\quad w \leftarrow \widetilde{\mathcal{W}_2}[i : i + 2m]$
4: $\quad poly \leftarrow \mathrm{BM}(w)$
5: $\quad Seq \leftarrow \mathrm{LFSR}(poly, w, \mathrm{length} = 2t)$
6: $\quad error \leftarrow \mathrm{dist}(Seq, \widetilde{\mathcal{W}_2})$
7: $\quad$ **if** $error < min\_error$ **then**
8: $\quad\quad min\_error \leftarrow error$
9: $\quad\quad poly\_saved \leftarrow poly$
10: $\quad\quad seq\_saved \leftarrow Seq$
return $poly\_saved$, seq_saved

Algorithm 4 takes a different approach: for each position in the sequence $\widetilde{\mathcal{W}_2}$, it considers a window of size $2m$, computes the corresponding BM polynomial, and reconstructs a candidate sequence for $\mathcal{W}_2$ by generating values both in the forward and backward directions using a Linear Feedback Shift Register (LFSR). The forward sequence is obtained by applying the LFSR from the initial window, and the backward sequence is built similarly by reversing both the window and the polynomial. The concatenation of these two partial sequences yields a full-length candidate sequence.

This candidate is then compared to the noisy sequence $\widetilde{\mathcal{W}_2}$ using the Hamming distance. The algorithm selects the polynomial that minimizes this distance, under the assumption that the correct polynomial will generate a sequence closer to the original, with fewer mismatches.

Importantly, as long as there exists at least one error-free sub-sequence of length $2m$ in the original sequence $\mathcal{W}_2$, the Berlekamp-Massey algorithm will recover the correct connection polynomial for that segment. In practice, we observe that wrong polynomials typically result in sequences with significantly more discrepancies, making this distance-based selection reliable when the overall noise level remains moderate.

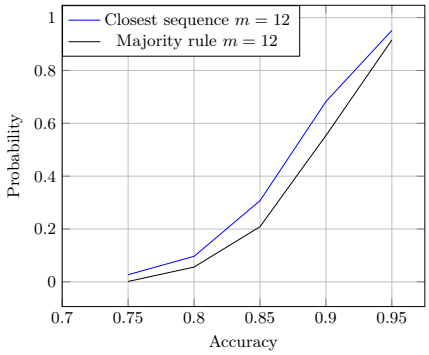


Fig. 1: Experimental success rate for polynomial recovery using majority vote rule and minimal distance.

Figure 1 shows the success rate of the two approaches in recovering the correct characteristic polynomial when the full-length sequence is affected by a number of random local bit flips. This noise model reflects typical side-channel measurement errors encountered when estimating the parity of Hamming weights. As illustrated in the figure, the distance-based approach (Algorithm 4) demonstrates greater resilience to noise. These empirical results can be compared with the theoretical predictions presented in Figure 2b.

During our experiments with the majority rule method, we observed that feeding Berlekamp-Massey with faulty sequences of length $2m$ often led to repeated recovery of the same incorrect polynomial. This negatively impacts the success rate of the attack, as it shows that the presence of a single error within a $2m$-length window can result in a persistent incorrect output. In contrast, the distance-based method appears more robust: a sequence of length $2t$ affected by errors still allows recovery of the correct polynomial as long as it contains one error-free sub-sequence of length $2m$.

It is important to emphasize that both methods rely on the assumption that a correctly recovered characteristic polynomial—based on what is believed to be an uncorrupted sub-sequence can be used to reconstruct the full *guessed*

sequence. This reconstructed sequence allows for the identification of error positions within the original noisy sequence. However, even when the error positions are known, determining the *direction* of the error (i.e., whether the Hamming weight was over- or underestimated) remains unresolved at this stage. This ambiguity is precisely addressed in Step 3 of the procedure described in the previous subsection.

A key empirical observation is the following: among 20,000 reconstruction attempts for (α, β) under a low accuracy scenario (0.74), only 392 yielded the correct pair. Crucially, among the remaining 19,608 incorrect predictions, **only one incorrect pair was not rejected** by Step 3. This highlights the effectiveness of the verification step: under reasonable accuracy conditions, the set $\mathcal{I}$ of candidate pairs (α, β) produced by Algorithm 2 contains the correct pair with high probability, and is extremely unlikely to contain false positives.

3.3 Success probability

Since, an unerroneous $2m$ sequence allows one to reconstruct the correct sequence, we shall focus our analysis on this problem. Hence, the first step is to determine the probability that the error vector $\boldsymbol{e}$ of a given Hamming weight admits a zero sub-block of length $2m$.

The proofs of the results of this section are postponed to Appendix B

Lemma 5. *Let $\boldsymbol{e} \in \mathbb{F}_2^{2t}$ with $\mathsf{wt}(\boldsymbol{e}) = l$. The probability that $\exists \mathcal{I} \subset [\![0; 2t-1]\!]$ s.t. $\boldsymbol{e}_{\mathcal{I}} = \mathbf{0}_{\#\mathcal{I}}$ and $\#\mathcal{I} \geq 2m$ equals*

$$\sum_{j=1}^{\lfloor \frac{2t-l}{2m} \rfloor} \frac{(-1)^{j+1} \binom{l+1}{j} \binom{2t-2mj}{l}}{\binom{2t}{l}} \tag{4}$$

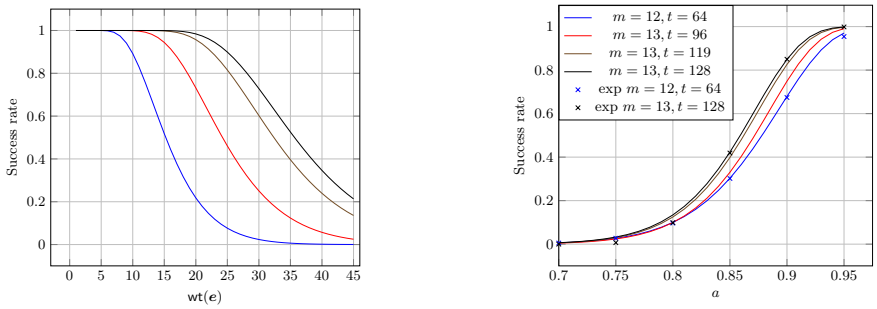
Now, let us see how the weight $\mathsf{wt}(\boldsymbol{e})$ depends on the accuracy of the classifier.

Remark 6. Recall that $\widetilde{\boldsymbol{H}}_{\mathsf{wt}}[, i] = (\mathsf{wt}(\beta_i) + \varepsilon_{0,i}, \ldots, \mathsf{wt}(\alpha_i^{2t-1}\beta_i) + \varepsilon_{2t-1,i})$ where $\varepsilon_{j,i}$ is a random variable that follows the categorical distribution, $Pr(\varepsilon_i = 0) = a$ and $Pr(\varepsilon_i = -1) = Pr(\varepsilon_i = 1) = \frac{1-a}{2}$. When moving down to $\mathbb{F}_2$ (where $1 = -1$) we obtain that $\boldsymbol{e}$ is a Bernoulli vector with $Pr(e_i = 0) = a$ and $Pr(e_i = 1) = 1 - a$. This implies that $\mathsf{wt}(\boldsymbol{e})$ is a discrete random variable that follows the Binomial distribution $\mathcal{B}(1-a, 2t)$.

Proposition 1. *The probability that Algorithm 4 output the correct sequence given a, the accuracy of the classifier, equals*

$$\sum_{l=0}^{2t} (1-a)^l a^{2t-l} \sum_{j=1}^{\lfloor \frac{2t-l}{2m} \rfloor} (-1)^{j+1} \binom{l+1}{j} \binom{2t-2mj}{l} \tag{5}$$

As demonstrated in Equation (2), the probability of successfully recovering a value of α is expressed as a function of both the distinguisher accuracy (see



(a) Pr(*success*) in function of $\mathsf{wt}(\boldsymbol{e})$.

(b) Pr(*success*) in function of a.

Fig. 2: Theoretical probability of success of Algorithm 4 for all *Classic McEliece* parameters in function of a) $\mathsf{wt}(\boldsymbol{e})$ and b) accuracy.

Figure 2b) and the error weight (see Figure 2a). The analysis reveals that the effectiveness of the attack is significantly influenced by the chosen security parameters. In particular, increasing these parameters tends to improve the success rate of the attack.

To assess the tightness and reliability of the theoretical bounds, extensive experiments were carried out. These confirmed that the theoretical predictions closely match the experimental outcomes. This correspondence is particularly valuable for evaluators: instead of performing full-scale attacks, one can estimate the distinguisher's accuracy and directly use the theoretical formula given in Equation (5) to predict the attack's success rate. This approach can substantially reduce evaluation time while maintaining meaningful security insights.

Bounding the accuracy Typically, it suffices to correctly recover t distinct pairs (α, β) in order to compute the polynomial $\gamma(x)$, which can then be used to reconstruct the full private key (see Section 4.1). Assume further that the extension field $\mathbb{F}_{2^m}$ is generated by a primitive polynomial—this is indeed the case for all ISO-specified parameter sets. As indicated by Lemma 3, this implies that no collisions are expected among the (α, β) pairs recovered via Algorithm 2. Consequently, we can lower bound the success probability of Algorithm 4 by t/n.

This observation allows us to derive a lower bound on the distinguisher accuracy a needed to ensure that $\Pr(\text{success}) \geq t/n$. For all *Classic McEliece* parameter sets (see Table 1), the ratio t/n is slightly below 0.021. Through numerical computation, we determined that this implies a lower bound of $a \geq 0.74$.

However, if one aims to employ more advanced techniques for key recovery—such as those described in [KM23,DCV+25]—then more than t correct α values are required. In particular, the method in [KM23] demands $mt+1$ correct values, leading to a stricter accuracy requirement of $a \geq 0.815$ for our algorithm. This same lower bound also applies in scenarios where $mt + \delta$ correct values are needed, as discussed in [DCV+25].

The source code for our experiments is available at `https://github.com/vingrosso/keyRecoveryClassicMcEliece`.

4 Applications

We now demonstrate two distinct contexts in which Algorithm 2 can serve as a foundational component for well-known key-recovery attacks.

4.1 Breaking Goppa codes with hints

Our approach fits within a broader framework—namely, the problem of breaking binary Goppa codes with additional side information (or *hints*). In this setting, the attacker is given the public binary Goppa code (i.e., a random basis for the private code, or equivalently, the matrix $\boldsymbol{H}_{\text{pub}}$), along with a set of helpful hints. The objective is to recover the private key, more specifically, the Goppa polynomial $\boldsymbol{\gamma}(x)$ and the private support $\mathcal{L} \subseteq \mathbb{F}_{2^m}$.

Although it is always possible to define a canonical support $\mathcal{L}_{\text{can}} = \{\alpha^i \mid i \in [\![0, 2^m - 2]\!]\}$, where α is a generator of $\mathbb{F}_{2^m}^*$, two main challenges arise:

1. The exact positions (indices) of the elements α^i within the actual support $\mathcal{L}$ are unknown.
2. It is unclear which specific elements of $\mathbb{F}_{2^m}$ have been excluded during the construction of $\mathcal{L}$.

Except for the recent work in [DCV+25], existing solutions to this problem [Sen00,KM23] do not explicitly analyze how these hints are generated, nor do they quantify the amount of side information required to recover the private key.

In Table 3, we outline how the results of our method can be used to generate the necessary hints for input to these established Goppa code equivalence solvers.

Algorithm 2 output	**Hints**	**Complexity**	**Article**
t pairs $(\alpha_i, \boldsymbol{\gamma}(\alpha_i))$ $a \geq 0.74$	$\boldsymbol{\gamma}(x)$	$\mathcal{O}(n^3 + 2^h n^2 \log n)$	[Sen00]
$mt + 1$ pairs $(\alpha_i, \boldsymbol{\gamma}(\alpha_i))$	$mt + 1$ elements α_i	$\mathcal{O}(n^5)$	[KM23]
$a \geq 0.815$	$t(m-2) + 1$ elements α_i and $\boldsymbol{\gamma}(x)$	$\mathcal{O}(n^4)$	
$mt + \delta$ pairs $(\alpha_i, \boldsymbol{\gamma}(\alpha_i))$ $a \geq 0.815$	$mt + \delta$ correct pairs $(\alpha_i, g(\alpha_i))$	$\mathcal{O}\left(n(mt)^{c-1}\right)$, $c \leq 3$	[DCV+25]

Table 3: The hints provided by Algorithm 2 as input to existing solvers for Solving the Goppa code equivalence problem together with the minimum accuracy.

4.2 Other McEliece variants

Alternant codes and Generalized Reed-Solomon codes are both algebraic codes that were used in a McEliece type cryptosystem. We can show that our method can be applied to any of these codes in order to recover their structure. The underlying idea is that all these codes admit as parity-check matrix a Vandermonde type matrix. To define these code we require a support $\mathcal{L} \stackrel{\mathsf{def}}{=} \{\alpha_0, \ldots, \alpha_{n-1}\} \subset \mathbb{F}_{2^m}$ of length $|\mathcal{L}| = n$ ($n \leq 2^m$) and a multiplier $\boldsymbol{u} \in \mathbb{F}_{2^m}^n$, with $u_i \neq 0$.

$$\boldsymbol{V}_k(\boldsymbol{v}, \mathcal{L}) \stackrel{\mathsf{def}}{=} \begin{pmatrix} v_0 & \ldots & v_{n-1} \\ \alpha_0 v_0 & \ldots & \alpha_{n-1} v_{n-1} \\ \vdots & \ddots & \vdots \\ \alpha_0^{k-1} v_0 & \ldots & \alpha_{n-1}^{k-1} v_{n-1} \end{pmatrix}.$$

The Generalized-Solomon code of dimension k and length n can be defined as follows

$$\mathbf{GRS}_k(\boldsymbol{u}, \mathcal{L}) \stackrel{\mathsf{def}}{=} \{(u_0 f(\alpha_0), \ldots, u_{n-1} f(\alpha_{n-1}) \mid f \in \mathbb{F}_{2^m}[x]\ ,\ \deg(f) < k\}\,.$$

The dual of a GRS code is still a GRS code, formally, we have $\mathbf{GRS}_k(\boldsymbol{u}, \mathcal{L})^\perp = \mathbf{GRS}_{n-k}(\boldsymbol{v}, \mathcal{L})$ where $v_i^{-1} = u_i \prod_{i \neq j}(\alpha_i - \alpha_j)$. This implies that $\boldsymbol{V}_{n-k}(\boldsymbol{v}, \mathcal{L})$ is a parity-check matrix of $\mathbf{GRS}_k(\boldsymbol{u}, \mathcal{L})$. A Reed-Solomon code is a GRS with multpiplier $\boldsymbol{u} = (1, \ldots, 1)$.

An alternate code of order r, denoted by $\mathbf{Alt}_r(\boldsymbol{u}, \mathcal{L})$, is a subfield-subcode, that can be defined using the parity-check matrix as follows

$$\mathbf{Alt}_r(\boldsymbol{u}, \mathcal{L}) = \{\boldsymbol{c} \in \mathbb{F}_2^n \mid \boldsymbol{V}_r(\boldsymbol{u}, \mathcal{L})\boldsymbol{c}^T = 0\}.$$

Its dimension is $k \geq n - rm$. Now we can see that $\mathscr{G}(\boldsymbol{\gamma}, \mathcal{L}) = \mathbf{Alt}_t(\boldsymbol{u}, \mathcal{L})$ with $u_i = \gamma(\alpha_i)^{-1}$, where $\gamma(\alpha_i) \neq 0, \forall \alpha_i \in \mathcal{L}$.

Remark 7. Since Algorithm 2 recovers pairs (α_i, β_i) using the Hamming weight matrix of the Vandermonde matrix $\boldsymbol{V}(\boldsymbol{v}, \mathcal{L})$, we have

- for Goppa codes we recover $\beta_i = \boldsymbol{\gamma}^{-1}(\alpha_i)$;
- for alternant codes we recover $\beta_i \in \mathbb{F}_{2^m}$;
- for GRS code we recover $\beta_i = (u_i(\prod_{i \neq j}(\alpha_i - \alpha_j))^{-1}$ for $u_i \in \mathbb{F}_{2^m}$;
- for Reed-Solomon codes we can recover $\beta_i = (\prod_{i \neq j}(\alpha_i - \alpha_j))^{-1}$.

4.3 Countermeasures

While our attack shows that decoding implementations based on Berlekamp–Massey and matrix–vector multiplications are vulnerable to side-channel leakage through Hamming weight observations, several effective countermeasures are available.

First, high-performance embedded implementations such as [CC21] use transposed additive FFTs to compute the double syndrome. This design substantially modifies the leakage characteristics and greatly reduces the correlation with Hamming weight, making our attack ineffective in such environments.

Second, notice that removing the computation of the private syndrome completely eliminates side-channel attacks such as [DCV+25,VCC+25] and even our attack. Theoretical solutions exist, since the classical Niederreiter scheme does not require to re-compute the double syndrome vector. Indeed, the ciphertext $\boldsymbol{z}$ could be directly decrypted by the following well-known procedure: i) compute $\boldsymbol{z}^* = \boldsymbol{S}^{-1}\boldsymbol{z}$, where $\boldsymbol{S}$ is the non-singular matrix used to compute the systematic form of $\boldsymbol{H}_{\text{priv}}$, and ii) decode $\boldsymbol{z}^*$ using Patterson decoding algorithm. However, we want to stress that it is not yet clear how to implement Patterson's algorithm efficiently and securely in constant time [Ber24].

Beyond changing the decoding algorithm or syndrome computation method, standard side-channel countermeasures—including masking techniques [ISW03] and shuffling [VMKS12]—can be employed to further mitigate leakage during the decoding process.

We strongly recommend that any deployment of *Classic McEliece* or similar code-based cryptosystems combine algorithm-level protections with robust implementation-level countermeasures to ensure resilience against both classical and quantum-capable adversaries.

5 Conclusion

In this work, we introduced a novel algebraic key-recovery attack against the reference implementation of the *Classic McEliece* KEM in scenarios involving side-channel leakage. Our method advances the state of the art by offering improved computational efficiency and enhanced robustness to estimation errors arising from noisy side-channel observations.

By analyzing the structure of the leakage and employing linear-algebraic techniques over finite fields, we demonstrated that it is possible to recover the secret key even in the presence of realistic noise, with the success probability depending primarily on the accuracy of the underlying classifier. Importantly, our approach scales well to larger parameter sets, making it practical for attacking implementations targeting higher security levels.

We also provided theoretical justification for the feasibility of distinguishing field elements via side-channel measurements, thereby offering partial validation of conjectures previously proposed in the literature.

Furthermore, we discussed the broader relevance of our techniques to other code-based cryptosystems, including their application to generic Goppa and alternant code key-recovery scenarios. Our framework serves as a foundation for combining algebraic recovery strategies with other advanced cryptanalytic tools.

Future work will focus on extending our analysis to broader error models, improving recovery rates under lower measurement accuracy, and designing more effective countermeasures. In addition, we plan to evaluate our attack on actual power traces collected from hardware implementations, in order to assess its practical impact in real-world environments.

Acknowledgements

The authors would like to thank the reviewers for their constructive feedback and helpful suggestions, which were invaluable in refining this work. We would like to especially thank our shepherd, Prof. Ruben Niederhagen, for his detailed and constructive guidance during the revision process. V-F. Dragoi was financed by Aurel Vlaicu University of Arad through the research grant UAV-IRG-1-2025-12. P-L. Cayrel and V. Grosso received funding from the France 2030 program, managed by the French National Research Agency under grant agreement No. ANR-22-PETQ-0008 PQ-TLS.

References

AAB+22. Carlos Aguilar-Melchor, Nicolas Aragon, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Edoardo Persichetti, Gilles Zémor, Jurjen Bos, Arnaud Dion, Jerome Lacan, Jean-Marc Robert, and Pascal Veron. HQC. Technical report, National Institute of Standards and Technology, 2022. available at `https://csrc.nist.gov/Projects/post-quantum-cryptography/round-4-submissions`.

ABC+22. Martin R. Albrecht, Daniel J. Bernstein, Tung Chou, Carlos Cid, Jan Gilcher, Tanja Lange, Varun Maram, Ingo von Maurich, Rafael Misoczki, Ruben Niederhagen, Kenneth G. Paterson, Edoardo Persichetti, Christiane Peters, Peter Schwabe, Nicolas Sendrier, Jakub Szefer, Cen Jung Tjhai, Martin Tomlinson, and Wen Wang. Classic McEliece. Technical report, National Institute of Standards and Technology, 2022. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-4-submissions`.

AHPT11. Roberto Avanzi, Simon Hoerder, Dan Page, and Michael Tunstall. Side-channel attacks on the McEliece and Niederreiter public-key cryptosystems. *Journal of Cryptograhic Engineering*, 1(4):271–281, 2011.

BBK08. Hacène Belbachir, Sadek Bouroubi, and Abdelkader Khelladi. Connection between ordinary multinomials, fibonacci numbers, bell polynomials and discrete uniform distribution. In *Annales Mathematicae et Informaticae*, volume 35, pages 21–30. Eszterházy Károly College, Institute of Mathematics and Computer Science Eger, 2008.

BCM+25. Marcus Brinkmann, Chitchanok Chuengsatiansup, Alexander May, Julian Nowakowski, and Yuval Yarom. Leaky mceliece: Secret key recovery from highly erroneous side-channel information. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2025(2):94–125, Mar. 2025.

Ber24. Daniel J. Bernstein. Understanding binary-Goppa decoding. *IACR Communications in Cryptology*, 1(1), 2024.

CC21. Ming-Shing Chen and Tung Chou. Classic McEliece on the ARM Cortex-M4. *IACR TCHES*, 2021(3):125–148, 2021.

CCD+22. Po-Jen Chen, Tung Chou, Sanjay Deshpande, Norman Lahr, Ruben Niederhagen, Jakub Szefer, and Wen Wang. Complete and improved FPGA implementation of classic McEliece. *IACR TCHES*, 2022(3):71–113, 2022.

CEvMS16. Cong Chen, Thomas Eisenbarth, Ingo von Maurich, and Rainer Steinwandt. Horizontal and vertical side channel analysis of a McEliece cryptosystem. *IEEE Transactions on Information Forensics and Secuity.*, 11(6):1093–1105, 2016.

DCV+25. Vlad-Florin Drăgoi, Brice Colombier, Nicolas Vallet, Pierre-Louis Cayrel, and Vincent Grosso. Full key-recovery cubic-time template attack on classic mceliece decapsulation. *IACR TCHES*, 2025(1):367–391, 2025.

FLZG24. Daniel Fallnich, Christian Lanius, Shutao Zhang, and Tobias Gemmeke. Efficient ASIC architecture for low latency classic McEliece decoding. *IACR TCHES*, 2024(2):403–425, 2024.

GCCD23. Vincent Grosso, Pierre-Louis Cayrel, Brice Colombier, and Vlad-Florin Dragoi. Punctured syndrome decoding problem - efficient side-channel attacks against classic McEliece. In Elif Bilge Kavun and Michael Pehl, editors, *COSADE 2023*, volume 13979 of *LNCS*, pages 170–192. Springer, Cham, April 2023.

GJJ22. Qian Guo, Andreas Johansson, and Thomas Johansson. A key-recovery side-channel attack on classic McEliece implementations. *IACR TCHES*, 2022(4):800–827, 2022.

HMP10. Stefan Heyse, Amir Moradi, and Christof Paar. Practical power analysis attacks on software implementations of McEliece. In Nicolas Sendrier, editor, *Third International Workshop on Post-Quantum Cryptography*, volume 6061 of *Lecture Notes in Computer Science*, pages 108–125, Darmstadt, Germany, May 2010. Springer.

ISW03. Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 463–481. Springer, Berlin, Heidelberg, August 2003.

KM23. Elena Kirshanova and Alexander May. Breaking Goppa-based McEliece with hints. *Inf. Comput.*, 293:105045, 2023.

Mas69. J. Massey. Shift-register synthesis and bch decoding. *IEEE Transactions on Information Theory*, 15(1):122–127, 1969.

MSSS11. H. Gregor Molter, Marc Stöttinger, Abdulhadi Shoufan, and Falko Strenzke. A simple power analysis attack on a McEliece cryptoprocessor. *Journal of Cryptographic Engineering*, 1(1):29–36, 2011.

Nie86. H. Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. *Problems of Control and Information Theory*, 15(2):159–166, 1986.

OC14. Colin O'Flynn and Zhizhang (David) Chen. ChipWhisperer: An open-source platform for hardware embedded security research. In Emmanuel Prouff, editor, *COSADE 2014*, volume 8622 of *LNCS*, pages 243–260. Springer, Cham, April 2014.

SAB+20. Peter Schwabe, Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, and Damien Stehlé. CRYSTALS-KYBER. Technical report, National Institute of Standards and Technology, 2020. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions`.

SCD+23. Boly Seck, Pierre-Louis Cayrel, Vlad-Florin Dragoi, Idy Diop, Morgan Barbier, Jean Belo Klamti, Vincent Grosso, and Brice Colombier. A side-channel attack against classic mceliece when loading the goppa polynomial. In Nadia El Mrabet, Luca De Feo, and Sylvain Duquesne, editors, *Progress in Cryptology - AFRICACRYPT 2023*, pages 105–125, Cham, 2023. Springer Nature Switzerland.

Sen00. Nicolas Sendrier. Finding the permutation between equivalent linear codes: The support splitting algorithm. *IEEE Trans. Inf. Theory*, 46(4):1193–1203, 2000.

SKE+23. Minjoo Sim, Hyeokdong Kwon, Siwoo Eum, Gyeongju Song, Minwoo Lee, and Hwajeong Seo. Efficient implementation of the classic McEliece on ARMv8 processors. In Howon Kim and Jonghee Youn, editors, *WISA 23*, volume 14402 of *LNCS*, pages 324–337. Springer, Singapore, August 2023.

VCC+25. Nicolas Vallet, Pierre-Louis Cayrel, Brice Colombier, Vlad-Florin Dragoi, and Vincent Grosso. Optimizing key recovery in *Classic McEliece*: Advanced error correction for noisy side-channel measurements. Cryptology ePrint Archive, Paper 2025/802, 2025.

VMKS12. Nicolas Veyrat-Charvillon, Marcel Medwed, Stéphanie Kerckhof, and François-Xavier Standaert. Shuffling against side-channel attacks: A comprehensive study with cautionary note. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 740–757. Springer, Berlin, Heidelberg, December 2012.

A Berlekamp-Massey Algorithm

Algorithm 5 Berlekamp-Massey Algorithm

Input: A sequence $W = (w_0, w_1, \ldots, w_{2m-1})$ of Hamming weights modulo 2
Output: The minimal polynomial $\boldsymbol{\chi}(Y)$ that generates the sequence

$L \leftarrow 0$ ▷ Current length of the characteristic polynomial
$\boldsymbol{\chi}(Y) \leftarrow 1$ ▷ Initialize the polynomial as 1
$\boldsymbol{\chi}_{\text{temp}}(Y) \leftarrow 1$
$\Delta \leftarrow 1$ ▷ Discrepancy value (initialized)
$l_{\text{old}} \leftarrow 0$ ▷ Index of last successful update
for $i = 0$ **to** $2m - 1$ **do** ▷ Iterate over the sequence
    Compute the discrepancy: $\Delta \leftarrow w_i + \sum_{j=1}^{L} \chi_j w_{i-j}$ ▷ If $\Delta = 0$, the sequence remains consistent with the current polynomial
    if $\Delta \neq 0$ **then** ▷ Update is needed if the discrepancy is nonzero
        $\boldsymbol{\chi}_{\text{temp}}(Y) \leftarrow \boldsymbol{\chi}(Y)$ ▷ Backup the current polynomial
        Update $\boldsymbol{\chi}(Y) : \boldsymbol{\chi}(Y) \leftarrow \boldsymbol{\chi}(Y) - \Delta Y^{i-l_{\text{old}}} \boldsymbol{\chi}_{\text{temp}}(Y)$
        if $2L \leq i$ **then** ▷ Check if the length of χ should be increased
            $L \leftarrow i + 1 - L$ ▷ Update the length of the polynomial
            $l_{\text{old}} \leftarrow i$
return $\boldsymbol{\chi}(Y)$ ▷ Return the minimal polynomial

B Proofs of results from Section 3.3

Let us begin with the proof of Lemma 5. We shall first recall some of the conventions needed here. $[x^l]P(x)$ will denote the extraction of the coefficient P_l from $P(x) = \sum_{i=0}^{n} P_i x^i$. Also we will denote the binary alphabet by $\mathcal{A} = \{0, 1\}$. Let $\mathcal{A}^* = \varepsilon, 0, 1, 00, \ldots$ denote the free monoid of finite length binary words where string concatenation is the monoid operation ($\varepsilon = 1_{\mathcal{A}^*}$ being the empty word). In terms of non-commutative formal series we can write

$$\left(\sum_{w \in \mathcal{A}^*} w \right) \left(1_{\mathcal{A}^*} - \sum_{a \in \mathcal{A}} a \right) = 1_{\mathcal{A}^*}. \tag{6}$$

Enumeration of binary words can now be done using the ordinary generating series. For example, when searching for all binary words of a given length, say l we are extracting $[x^l] \quad \frac{1}{1-(x+x)} = \frac{1}{1-2x}$ since both letters from $\mathcal{A}$ count the same, which gives 2^l binary words. We can now proceed to the proof of our result.

Proof. To estimate the wanted probability we will determine its complement, i.e., the probability of having all blocks of consecutive zeros with length at most $2m-1$. Thus, we need to enumerate all binary words of length $2t$ with l occurrences

of the letter 1 having less than $2m$ consecutive 0s. Such words are of the form $0^{<2m}(10^{<2m})^l$. If we map the letter 0 into the formal variable z and 1 into x we get that extracting $[z^{2t-l}]$ from $(1+z+\cdots+z^{2m-1})^{l+1}$ gives the wanted result. Indeed, in terms of (now commutative) formal series in z, we have

$$[x^l z^{2t-l}] \sum_{i\geq 0} (1+z+\cdots+z^{2m-1}) \left(x(1+z+\cdots+z^{2m-1})\right)^i$$

$$[x^l z^{2t-l}] \sum_{i\geq 0} x^i \left(\frac{1-z^{2m}}{1-z}\right)^{i+1}$$

$$[z^{2t-l}] \left(\frac{1-z^{2m}}{1-z}\right)^{l+1}.$$

The coefficient of z^{2t-l} in $(1+z+\cdots+z^{2m-1})^{l+1}$ also denoted in the literature by $\binom{l+1}{2t-l}_{2m-1}$ is a generalization of the binomial coefficient which admits the following combinatorial formula (see [BBK08])

$$\binom{l+1}{2t-l}_{2m-1} = \sum_{j=0}^{\lfloor \frac{2t-l}{2m} \rfloor} (-1)^j \binom{l+1}{j} \binom{2t-2mj}{l}.$$

The total number of binary words of length $2t$ having l occurrences of 1 equals $\binom{2t}{l}$ which yields the wanted result. □

Now, we can continue with the proof of Proposition 1

Proof. Using Lemma 5 and Remark 6 we deduce

$$Pr(succes) = \sum_{l=0}^{2t} Pr(succes \mid \mathsf{wt}(\boldsymbol{e}) = l) Pr(\mathsf{wt}(\boldsymbol{e}) = l) \tag{7}$$

$$= \sum_{l=0}^{2t} \binom{2t}{l} (1-a)^l a^{2t-l} \sum_{j=1}^{\lfloor \frac{2t-l}{2m} \rfloor} \frac{(-1)^{j+1} \binom{l+1}{j} \binom{2t-2mj}{l}}{\binom{2t}{l}} \tag{8}$$

$$= \sum_{l=0}^{2t} (1-a)^l a^{2t-l} \sum_{j=1}^{\lfloor \frac{2t-l}{2m} \rfloor} (-1)^{j+1} \binom{l+1}{j} \binom{2t-2mj}{l} \tag{9}$$

□