AIMS Mathematics, 11(6): 17673–17721.
DOI: 10.3934/math.2026722
Received: 21 March 2026
Revised: 15 May 2026
Accepted: 27 May 2026
Published: 17 June 2026

https://www.aimspress.com/journal/Math

Research article

Guided syndrome decoding under posterior leakage

Andrés Florián-Quitián[1], Valérie Gauthier-Umaña[1], Estefanía Laverde-Becerra[1] and Ricardo Villanueva-Polanco[2,*]

[1] Department of Systems and Computing Engineering, Universidad de los Andes, Bogotá, Colombia

[2] Department of Systems and Computing Engineering, Universidad del Norte, Barranquilla, Colombia

* **Correspondence:** Email: rpolanco@uninorte.edu.co.

Abstract: The syndrome decoding problem (SDP) is the computational foundation of code-based cryptography, underlying schemes such as McEliece, Niederreiter, and the Hamming quasi-cyclic key encapsulation mechanism standardized in 2025. While these constructions are secure in the standard model, practical implementations may still leak partial information about secret error vectors through physical attacks such as cold boot attacks and related side-channel scenarios. Motivated by this setting, we study syndrome decoding in the presence of probabilistic leakage. We adopt a bitwise Bayesian leakage model (BBLM) that represents leakage as coordinate-wise posterior beliefs over the secret error vector, providing a generic abstraction for noisy bitwise leakage. Building on this model, we develop a posterior-guided decoding framework that integrates leakage-derived information directly into the information set decoding (ISD) process through conditioned decoding and recursive instance reduction. The framework is decoder-agnostic and can be combined with different ISD variants and subset-enumeration strategies. As a proof of concept, we instantiate the framework using a genetic-algorithm-based enumerator guided by posterior-informed fitness functions. Experimental results on random linear codes and Reed–Muller (RM) codes show that informative leakage can guide subset selection toward conditioned instances that are substantially easier to decode under realistic asymmetric noise conditions.

Keywords: cryptography; quantum computing; side-channel attacks; information security; parity check codes; error correction codes; genetic algorithms
Mathematics Subject Classification: 11T61, 81P94

1. Introduction

Error-correcting codes emerged in the mid-twentieth century to address the fundamental problem of reliable communication over noisy channels [1]. The field has since evolved into a diverse landscape of algebraic constructions, including Reed-Solomon codes [2], Reed-Muller (RM) codes [3], and Goppa codes [4], which underpin modern digital communication, data storage, and optical media.

A central concept underlying many of these constructions is the *syndrome*, which compactly represents the deviation of a received word from a valid codeword. The *syndrome decoding problem* (SDP) formalizes the challenge of finding an error vector of a specified Hamming weight that matches a given syndrome. In the general case, this problem is NP-hard [1].

Public-key cryptography is currently undergoing a paradigm shift, as advances in quantum computing threaten the security of widely deployed schemes. Post-quantum cryptography addresses this challenge by designing primitives intended to remain secure against quantum adversaries.

In 1978, McEliece [5] introduced a seminal code-based cryptosystem that has withstood decades of cryptanalytic scrutiny under appropriate parameter choices. Its practical deployment, however, remains limited by large public keys. Since then, several variants have been proposed, either by replacing binary Goppa codes with alternative code families or by modifying the masking techniques used to conceal the underlying structure. Examples include constructions based on quasi-cyclic codes [6], RM codes [7], and rank-metric Gabidulin codes [8].

In recent years, post-quantum cryptographic schemes have attracted significant attention, largely driven by the post-quantum cryptography standardization process led by the national institute of standards and technology (NIST) [9]. Among code-based candidates, the Hamming quasi-cyclic scheme [10] has emerged as a prominent construction, with security based on the hardness of the quasi-cyclic Syndrome decoding problem, a quasi-cyclic variant of the syndrome decoding problem. Hamming quasi-cyclic has been selected by NIST for standardization as an additional post-quantum key-establishment mechanism. In parallel, several candidates considered in the NIST digital signature process rely on coding-theoretic assumptions [11].

Code-based cryptography offers a well-studied approach to post-quantum security, with practical performance profiles that continue to motivate implementation-oriented research. As attention increasingly focuses on hard problems such as the SDP, it is also necessary to account for physical attack vectors that exploit side-channel leakage. Such attacks can undermine implementations even when the underlying schemes satisfy formal security notions [12]. In cold boot attacks (CBA) [13], the adversary does not observe the secret key directly, but instead obtains noisy and asymmetric bit-level information that provides probabilistic hints about the secret error vector.

A substantial body of cryptanalytic research addresses key recovery from partial, noisy, or biased leakage, including cold boot attacks and partial key exposure scenarios. Since the pivotal work of Halderman et al. on cold boot attacks [14], these models have been studied across a wide range of cryptographic settings. Several studies investigated Rivest-Shamir-Adleman and related partial key exposure problems [15–17]. Other studies considered additional public-key and symmetric-key settings under noisy or biased leakage assumptions [18–20]. Further work explored implementation-specific attacks and practical key-recovery techniques [21–23]. These techniques have also been extended to post-quantum schemes and structured secret-key recovery problems [13, 24]. More recent contributions study related leakage, cold-boot, and enumeration-based

attacks in modern cryptographic settings [25, 26].

Related partial key exposure attacks consider incomplete or biased information about secret material, often modeled through oracle access or correctness constraints rather than explicit noise processes [27–29]. Recent work has further studied these leakage settings in related cryptanalytic contexts [30, 31]. These perspectives motivate the bitwise Bayesian leakage model (BBLM) adopted in this work, which represents leakage as posterior beliefs about individual secret bits [32].

Motivated by these leakage scenarios, we study syndrome decoding in the presence of probabilistic hints derived from side-channel observations. Building on the classical information set decoding (ISD) framework introduced by Prange [33], we develop a posterior-guided decoding framework that incorporates leakage-derived posterior information directly into the decoding process. The proposed approach uses coordinate-wise posterior probabilities to guide conditioned decoding and recursive instance reduction, thereby reducing the effective decoding effort under informative leakage conditions.

The proposed framework is decoder-agnostic and can be combined with different ISD variants, enumeration strategies, and code families. Rather than introducing a fundamentally new asymptotic decoding algorithm, the contribution of this work lies in integrating probabilistic leakage information into the syndrome decoding process itself. As a proof of concept, we instantiate the framework using a genetic-algorithm-based enumerator that explores candidate information sets through posterior-informed fitness scores. The genetic algorithm should therefore be viewed as a heuristic enumerator within the broader posterior-guided decoding framework, rather than as the primary contribution of the work.

Overall, this work bridges classical coding theory, modern side-channel threat models, and post-quantum cryptography within a unified framework. By incorporating probabilistic leakage into syndrome decoding, we provide a principled methodology for studying leakage-assisted decoding attacks against code-based cryptosystems. More broadly, the interaction between probabilistic modeling, combinatorial decoding, and heuristic subset enumeration suggests new directions for leakage-aware cryptanalysis and leakage-resilient code-based cryptography.

The remainder of this paper is organized as follows. Section 2 introduces the notation, reviews the syndrome decoding problem, and formalizes the threat model considered in this work. Section 4 presents the proposed leakage-aware decoding approach. Section 5 provides its theoretical analysis, focusing on f-subset enumeration and generic success-probability bounds. Section 6 describes the genetic-algorithm-based enumerator, while Section 7 reports the experimental results. Finally, Section 8 concludes the paper and discusses future research directions.

2. Background

This section introduces the notation and formalizes the decoding problem considered in this work. It also reviews RM codes used as test instances, the threat model underlying our attack scenario, and Prange's classical ISD algorithm, which serves as the basis for our posterior-guided decoding approach.

2.1. Notation

Throughout this paper, we denote by $\mathbb{F}_q$ the finite field with q elements. Lowercase letters, such as x, denote vectors, while uppercase letters, such as H, denote matrices or sets depending on the context.

Unless otherwise stated, all computations are carried out over $\mathbb{F}_2$. We write $[n]$ for the set $\{1, 2, \ldots, n\}$. Finally, $GL_{n-k}(\mathbb{F}_2)$ denotes the general linear group of invertible $(n-k)\times(n-k)$ matrices over $\mathbb{F}_2$, with matrix multiplication as the group operation.

2.2. The SDP

Let $C \subset \mathbb{F}_2^n$ be a binary linear code of length n and dimension k, with parity-check matrix $H \in \mathbb{F}_2^{(n-k)\times n}$.

Definition 2.1. *Given a parity-check matrix H, a syndrome $s \in \mathbb{F}_2^{n-k}$, and a target Hamming weight w, find a vector $e \in \mathbb{F}_2^n$ such that*

$$H \cdot e^T = s \quad \text{and} \quad \mathrm{wt}(e) = w.$$

This problem is known to be NP-hard in the general case and underpins the security of many code-based cryptosystems, including McEliece and Niederreiter-type constructions and variants based on RM codes. Efficiently solving the SDP remains a central challenge in cryptanalysis. In this work, we study settings in which noisy partial side information about e is available, as in cold boot attacks.

2.3. RM codes

The RM codes were introduced in 1954 by Muller [3] and Reed [34] who proposed the first decoding algorithm. To describe RM codes, consider the polynomial ring $\mathbb{F}_2[x_1, x_2, \ldots, x_m]$ and let $f \in \mathbb{F}_2[x_1, x_2, \ldots, x_m]$ be a multivariate polynomial. For a point $z = (z_1, \ldots, z_m) \in \mathbb{F}_2^m$, define $\mathrm{Eval}_z(f) = f(z_1, \ldots, z_m)$ as the evaluation of f at z. Then, the complete evaluation of f over all points in $\mathbb{F}_2^m$ is given by

$$\mathrm{Eval}(f) = \left(\mathrm{Eval}_{z_1}(f), \mathrm{Eval}_{z_2}(f), \ldots, \mathrm{Eval}_{z_{2^m}}(f)\right),$$

where $z_i \in \mathbb{F}_2^m$ for $i = 1, \ldots, 2^m$.

The RM code $RM(r, m)$ consists of the evaluation vectors of all polynomials in m variables with total degree less than or equal to r [35].

Definition 2.2. *The RM code with parameters r and m, $\mathrm{RM}(r, m)$, is defined as the following set of binary vectors:*

$$\mathrm{RM}(r, m) = \{\mathrm{Eval}(f) \mid f \in \mathbb{F}_2[x_1, x_2, \ldots, x_m],\ \deg(f) \le r\}.$$

These codes are notable for their recursive definition.

Definition 2.3. *The RM code with parameters r and m is defined recursively as:*

$$\begin{aligned}\mathrm{RM}(r, m) &= \{\,(u \mid u + v) \mid u \in \mathrm{RM}(r, m-1), v \in \mathrm{RM}(r-1, m-1)\}\\ &= \{\,(u \mid u + v) \mid u \in U, v \in V\,\},\end{aligned}$$

where

$$n = 2^m, \quad k = \sum_{i=0}^{r} \binom{m}{i}, \quad d = 2^{m-r}.$$

It has a generator matrix given by

$$G(r,m) = \begin{pmatrix} G(r,m-1) & G(r,m-1) \\ 0 & G(r-1,m-1) \end{pmatrix}, \tag{2.1}$$

where 0 *is the zero matrix with the same dimensions as* $G(r-1,m-1)$*,* $G(r,m-1)$ *and* $G(r-1,m-1)$ *are the generator matrices of* $\mathrm{RM}(r,m-1)$ *and* $\mathrm{RM}(r-1,m-1)$*, respectively. This recursive construction is known as the Plotkin construction [36].*

If $r = 0$, the code $\mathrm{RM}(0,m)$ consists of repetition codewords of length n. On the other hand, if $r = m$, we have

$$\mathrm{RM}(m,m) = \mathbb{F}_2^n,$$

that is, the entire vector space. Some key structural properties of Reed–Muller codes are presented below.

Proposition 2.4. The minimum distance for a $\mathrm{RM}(r,m)$ code is $d = 2^{m-r}$ [35].

2.4. Threat model

We formalize the threat model used throughout this work, termed the BBLM [32]. The BBLM provides a general probabilistic abstraction for modeling information leakage at the level of individual secret bits. Instead of being tied to a specific physical or algorithmic attack vector, the model captures leakage through per-bit likelihood functions and the corresponding posterior distributions. This abstraction enables principled reasoning about key recovery across a broad range of cryptographic attack scenarios, including memory decay, noisy partial disclosure, and side-channel inference.

Formally, let $x^* = (x_1^*, \ldots, x_N^*) \in \{0,1\}^N$ denote the unknown secret bit string. The adversary does not observe x^* directly. Instead, it obtains leakage observations $L_1, \ldots, L_N$, where each L_i provides possibly noisy or partial information about the corresponding bit x_i^*. The leakage alphabet for L_i may vary depending on the attack setting, including binary symbols, erasures, or real-valued classifier outputs.

For each bit position i, the adversary computes the posterior distribution of x_i^* conditioned on L_i. By Bayes' rule, for each $b \in \{0,1\}$,

$$\Pr(x_i^* = b \mid L_i) = \frac{\Pr(L_i \mid x_i^* = b)\Pr(x_i^* = b)}{\Pr(L_i)},$$

where

$$\Pr(L_i) = \sum_{b\in\{0,1\}} \Pr(L_i \mid x_i^* = b)\Pr(x_i^* = b).$$

Here, $\Pr(x_i^* = b)$ denotes the prior probability of bit value b at position i, while $\Pr(L_i \mid x_i^* = b)$ denotes the likelihood of observing leakage L_i given that bit value.

Independence assumptions. To ensure tractable inference at scale, we adopt two standard simplifying assumptions:

(1) *Bitwise independent prior:*

$$\Pr(x^*) = \prod_{i=1}^{N} \Pr(x_i^*).$$

That is, prior knowledge does not introduce statistical dependence across bits.

(2) *Conditional independence of leakage:*

$$\Pr(L_1, \ldots, L_N \mid x^*) = \prod_{i=1}^{N} \Pr(L_i \mid x_i^*).$$

Thus, conditioned on the secret, each leakage observation depends only on the corresponding bit.

Under these assumptions, the posterior distribution factorizes as

$$\Pr(x^* \mid L_1, \ldots, L_N) = \prod_{i=1}^{N} \Pr(x_i^* \mid L_i).$$

This factorization is central to the BBLM: The unconstrained posterior over the N-bit secret can be represented through N independent bitwise posterior distributions. Such independence assumptions are standard in the analysis of cold boot attacks and related physical leakage models. Empirical studies indicate near-independent bit-decay behavior in modern DRAM modules [37–39], supporting this approximation for large-scale inference.

Since the secret is binary, each posterior can be represented by a single scalar:

$$\Pr(x_i^* = 0 \mid L_i) = 1 - \Pr(x_i^* = 1 \mid L_i).$$

2.4.1. Leakage instantiations

The BBLM encompasses several important leakage models commonly encountered in practice.

Erasure setting. In the erasure model, the adversary learns a subset of bits exactly, while the remaining bits are completely unobserved [30]. This corresponds to a binary erasure channel abstraction:

$$L_i = \begin{cases} x_i^* & \text{(revealed)}, \\ \perp & \text{(erased)}, \end{cases}$$

where $\perp$ denotes the absence of information.

Assuming a bitwise-independent uniform prior

$$\Pr(x_i^* = 0) = \Pr(x_i^* = 1) = \tfrac{1}{2},$$

revealed positions yield posterior probability 1, while erased positions satisfy

$$\Pr(x_i^* \mid L_i = \perp) = \tfrac{1}{2}.$$

More generally, for non-uniform priors, erased positions satisfy

$$\Pr(x_i^* \mid L_i = \perp) = \Pr(x_i^*),$$

i.e., in the absence of leakage, the posterior equals the prior.

Error setting. The error setting generalizes erasure by allowing noisy leakage rather than complete loss of information. It corresponds to a binary symmetric channel with crossover probability $\delta \in [0, \frac{1}{2}]$:

$$\Pr(L_i \mid x_i^*) = \begin{cases} 1 - \delta, & \text{if } L_i = x_i^*, \\ \delta, & \text{if } L_i \neq x_i^*. \end{cases}$$

Under uniform priors, Bayes' rule yields

$$\Pr(x_i^* = L_i \mid L_i) = 1 - \delta, \qquad \Pr(x_i^* \neq L_i \mid L_i) = \delta.$$

Thus, the posterior interpolates between certainty ($\delta = 0$) and complete uncertainty ($\delta = \frac{1}{2}$).

CBA. In cold boot attacks, the adversary obtains a decayed memory image $\widetilde{x}$ of the secret stored in DRAM. Leakage is modeled via asymmetric bit-flip probabilities:

$$\Pr(\widetilde{x_i} \mid x_i^*) = \begin{cases} 1-\beta, & x_i^* = 1,\ \widetilde{x_i} = 1, \\ \beta, & x_i^* = 1,\ \widetilde{x_i} = 0, \\ 1-\alpha, & x_i^* = 0,\ \widetilde{x_i} = 0, \\ \alpha, & x_i^* = 0,\ \widetilde{x_i} = 1, \end{cases}$$

where α and β denote $0 \to 1$ and $1 \to 0$ flip probabilities, respectively. Empirical evidence indicates $\alpha \ll \beta$ [15], reflecting the asymmetric decay behavior of DRAM cells. The resulting posterior probabilities inherit this asymmetry, typically assigning higher confidence to observed zeros than to observed ones.

Side-channel classifiers. In profiling side-channel attacks, leakage features $L_i = f_i(T)$ are extracted from physical measurements T. Statistical models or machine-learning classifiers implement functions

$$g_i : \mathcal{L}_i \to [0, 1],$$

where

$$g_i(L_i) \approx \Pr(x_i^* = 1 \mid L_i).$$

When correctly specified and well calibrated, g_i coincides with the Bayesian posterior; otherwise it provides an approximation that can still be used for probabilistic ranking within the BBLM.

Model lmitations. The BBLM relies on bitwise independence and accurate posterior estimation. In practice, residual inter-bit correlations may arise from architectural, algorithmic, or physical coupling effects. Similarly, classifier outputs may be imperfectly calibrated, particularly under distribution shift or limited profiling data. While moderate deviations from the assumptions do not invalidate the framework, explicitly modeling structured multivariate leakage lies beyond the present scope and is left for future work.

2.4.2. Posterior scoring

Given a candidate bitstring $x' \in \{0, 1\}^N$, the joint posterior under the BBLM factorizes as

$$\Pr(x' \mid L_1, \ldots, L_N) = \prod_{i=1}^{N} \Pr(x_i^* = x_i' \mid L_i).$$

Taking logarithms yields the additive scoring rule

$$\texttt{Score}(x') = \sum_{i=1}^{N} \log \Pr(x_i^* = x_i' \mid L_i),$$

which is numerically stable and computationally convenient. Maximizing $\texttt{Score}(x')$ is equivalent to maximum a posteriori (MAP) estimation under the BBLM and guides the search toward the most probable secret.

2.5. The Prange algorithm

The ISD algorithm introduced by Prange [33] is one of the foundational methods for solving the SDP. We recall its classical form as a baseline for the posterior-guided extensions introduced later.

In its standard formulation, given a parity-check matrix $H \in \mathbb{F}_2^{(n-k)\times n}$, a syndrome $s \in \mathbb{F}_2^{n-k}$, and a target weight w, Prange's algorithm proceeds as follows:

(1) Select a random *information set* $I \subset \{1, \ldots, n\}$ of size k, under the hypothesis that $x_I = 0$.

(2) Let $J = \{1, \ldots, n\} \setminus I$ and decompose $H = [H_I \mid H_J]$.

(3) If H_J is nonsingular, solve $H_J x_J^T = s$ for $x_J \in \mathbb{F}_2^{n-k}$.

(4) Form the candidate vector $x = (x_I = 0, x_J)$ and check whether $\mathrm{wt}(x) = w$.

For a fixed solution vector of weight w, the probability that the randomly selected information set I is error-free is

$$\Pr(x_I = 0) = \frac{\binom{n-k}{w}}{\binom{n}{w}}.$$

As n grows, the probability that a randomly selected information set is error-free decreases exponentially, requiring a large number of decoding attempts. In its classical form, Prange's algorithm samples information sets uniformly at random, implicitly treating all coordinate positions as equally likely to contain errors. Consequently, it cannot exploit auxiliary information such as reliability estimates, probabilistic leakage, or partial knowledge of the error vector to bias the search toward more promising subsets. This uniform sampling strategy becomes statistically inefficient when informative side information is available.

The approach proposed in this work addresses this limitation by incorporating leakage-derived posterior information directly into the information-set selection process, while remaining complementary to the underlying ISD decoder. Although our experimental evaluation focuses on Prange's original algorithm for clarity and ease of analysis, the framework is not tied to a specific ISD variant. Modern decoding algorithms such as Stern's algorithm [40] and BJMM [41] retain the same general information-set sampling structure and could similarly benefit from posterior-guided subset selection or leakage-aware coordinate weighting. Consequently, the proposed strategy naturally extends to a broad class of ISD-based decoders.

3. Related work

Research on decoding algorithms dates back to the first ISD algorithm introduced by Prange in 1962. Originally proposed as a generic decoding method for linear codes, ISD later became the central attack framework in code-based cryptography. Since then, numerous variants have improved the efficiency of ISD-based decoding. In particular, the algorithm of May and Ozerov [42] introduced refined nearest-neighbor and list-based techniques that significantly improve decoding performance for random linear codes. These advances sharpen the best-known complexity bounds for code-based cryptanalysis and provide a baseline for evaluating further reductions in decoding cost.

A second line of research studies *partial key exposure and leakage models* in post-quantum cryptography. In lattice-based settings, schemes based on the ring learning with errors problem have also been studied in leakage settings [43] formalizes the impact of revealing partial information about

the secret in the number theoretic transform domain and shows how algebraic structure can amplify leakage. More generally, partial key exposure attacks against lattice-based schemes [44] demonstrate that redundancy in secret representations may enable full recovery once erasures or errors exceed certain thresholds. Related work on seed recovery under probabilistic leakage [32] introduces posterior-guided enumeration and empirical success estimation, showing that probabilistic side information can substantially reduce effective security. Additional implementation-level attacks, such as average-case noise estimation against Cheon-Kim-Kim-Song-based homomorphic encryption schemes [45], further illustrate how practical leakage can compromise post-quantum constructions.

Another important body of work focuses on *side-channel attacks against code-based cryptosystems*, including classic McEliece and related schemes [29, 46]. These attacks demonstrate that physical leakage, such as timing or power consumption, can reveal core secret components. In particular, recent attacks recover the secret support and Goppa polynomial of classic McEliece through power-trace analysis of the additive fast Fourier transform step during decryption [47]. Such results bridge abstract leakage models and practical key-recovery attacks.

Closely related are approaches that model leakage as *hints* modifying the underlying decoding problem. In the lattice setting, several works incorporate perfect, modular, approximate, or short-vector hints into distorted bounded distance decoding instances to estimate the residual hardness of learning with errors [48]. In code-based cryptography, partial information such as known error positions, error-free coordinates, block weights, or subsets of Goppa points has similarly been integrated into ISD algorithms, yielding substantial reductions in decoding complexity [29, 46].

More recently, D'Achille et al. [49] studied syndrome decoding in the presence of probabilistic hints by transforming the problem into a soft-decision decoding instance and incorporating hint-derived reliabilities directly into ISD algorithms. Their approach uses leakage information to bias information-set selection and derives complexity improvements, interpolation results, and polynomial-time decoding thresholds under sufficient hint exposure. Conceptually, both approaches exploit probabilistic leakage information to guide syndrome decoding beyond uniform ISD search. However, whereas their work focuses on reliability-guided adaptations of ISD derived from noisy linear hints, our framework models leakage through a general BBLM and studies decoder-agnostic posterior-guided conditioning, recursive instance reduction, and generic probabilistic subset enumeration independently of the underlying decoding algorithm.

In contrast, the framework proposed in this work models leakage directly at the level of posterior probabilities over the unknown error vector. Rather than deriving coordinate reliabilities from explicit linear hints, we assume a general probabilistic leakage channel represented through the BBLM, which provides posterior correctness estimates for individual coordinates. These posteriors are then exploited through recursive conditioning, guided subset selection, and probabilistic instance reduction strategies that remain largely independent of the underlying decoding algorithm. This perspective allows leakage-aware decoding techniques beyond reliability-guided ISD alone and enables a more general study of posterior-assisted syndrome decoding under noisy memory leakage.

To the best of our knowledge, this work is the first to study syndrome decoding under a general Bayesian posterior leakage framework derived from noisy memory exposure models such as cold boot leakage. While recent work has demonstrated the effectiveness of reliability-guided ISD under probabilistic hints, the proposed framework instead focuses on decoder-agnostic posterior-guided conditioning and recursive probabilistic reduction directly at the SDP level. This perspective

complements existing work on ISD with hints, partial key exposure, and side-channel cryptanalysis by providing a generic methodology for incorporating posterior leakage information into syndrome decoding independently of the specific decoding algorithm or cryptographic construction.

4. Posterior-guided ISD

We now introduce a posterior-guided decoding framework that incorporates side-channel leakage through bitwise posterior probabilities derived from the BBLM. These probabilities are used to bias information-set selection toward subsets that are more likely to satisfy the assumptions required for successful decoding, thereby improving the efficiency of the search process.

The following subsections formalize the resulting problem reduction and describe the associated posterior-guided subset enumeration strategy.

4.1. Guided syndrome decoding

We begin by considering syndrome decoding in the presence of partial information about the error vector. In particular, we assume that a subset of coordinates is known. Conditioning on these positions induces a reduction of the original SDP to a smaller decoding instance with fewer variables and reduced target weight.

The following theorem formalizes this reduction and generalizes Theorems 1 and 3 from [46].

Theoretical foundations: reduction with known error positions.

Theorem 4.1. *Let $C \subseteq \mathbb{F}_2^n$ be a binary linear code of dimension k with parity-check matrix $H \in \mathbb{F}_2^{(n-k)\times n}$. Suppose $x \in \mathbb{F}_2^n$ is an error vector of Hamming weight w, and $s = Hx^\top$ is the observed syndrome.*

Assume x is known at a subset $I_0 \subset [n]$ with $|I_0| = f \leq k$, and let $J = [n] \setminus I_0$. Denote by H_{I_0} and H_J the submatrices of H restricted to the columns in I_0 and J, respectively. Define:

$$s' := s - H_{I_0} x_{I_0}^\top.$$

Let $r := \mathrm{rk}(H_J)$, and apply a full-rank transformation $U \in \mathrm{GL}_{n-k}(\mathbb{F}_2)$ such that:

$$UH_J = \begin{bmatrix} H_J^{(r)} \\ 0 \end{bmatrix}, \quad Us' = \begin{bmatrix} s_1'' \\ s_2'' \end{bmatrix},$$

*where $H_J^{(r)} \in \mathbb{F}_2^{r\times(n-f)}$ * has full row rank and $s_2'' = 0$. Then, the original decoding problem reduces to the following: find $x_J \in \mathbb{F}_2^{n-f}$ such that*

$$H_J^{(r)} x_J^\top = s_1'', \quad \mathrm{wt}(x_J) = w - \mathrm{wt}(x_{I_0}).$$

The reduced instance has the effective parameters

$$(n', k', w') = (n - f,\ (n - f) - r,\ w - \mathrm{wt}(x_{I_0})).$$

*The decomposition with a zero block is only meaningful when $r < n-k$; that is, when the rows of H_J are not all linearly independent. If H_J has full row rank (i.e., $r = n - k$), then no zero rows are introduced by the row-reduction.

Proof. Decompose the error vector as $x = (x_{I_0}, x_J) \in \mathbb{F}_2^n$. Then the syndrome can be written as:

$$s = Hx^\top = H_{I_0} x_{I_0}^\top + H_J x_J^\top.$$

Since x_{I_0} is known, define the residual syndrome:

$$s' := s - H_{I_0} x_{I_0}^\top = H_J x_J^\top.$$

Let $r := \mathrm{rk}(H_J)$, and $U \in \mathrm{GL}_{n-k}(\mathbb{F}_2)$ be the invertible matrix obtained as the product of the elementary row-operation matrices used in Gaussian elimination on H_J. Applying these row operations to H_J, we obtain

$$UH_J = \begin{bmatrix} H_J^{(r)} \\ 0 \end{bmatrix}, \quad \text{with } H_J^{(r)} \in \mathbb{F}_2^{r\times(n-f)}.$$

Then, the transformed syndrome satisfies

$$Us' = \begin{bmatrix} H_J^{(r)} x_J^\top \\ 0 \end{bmatrix} = \begin{bmatrix} s_1'' \\ s_2'' \end{bmatrix}.$$

Hence, $s_2'' = 0$, and the system reduces to

$$H_J^{(r)} x_J^\top = s_1'',$$

which defines a system of r independent equations in $n - f$ variables. The parameters of the reduced instance are:

$$n' = n - f, \quad k' = n' - r = (n - f) - \mathrm{rk}(H_J), \quad w' = w - \mathrm{wt}(x_{I_0}).$$

This completes the proof. □

This reduction theorem provides the formal basis for posterior-guided decoding: When some bits of the error vector can be estimated with sufficient confidence, the decoding problem effectively shrinks in both length and weight. Building on this result, we now describe a practical decoding procedure that uses these posterior probabilities to drive subset selection and error recovery.

4.2. Posterior-guided decoding framework

We now introduce a posterior-guided decoding framework that uses probabilistic leakage information to reduce the original SDP to smaller decoding instances. The approach consists of two main stages: (i) selecting reliable coordinate positions according to posterior confidence, and (ii) decoding the resulting reduced instance using a generic syndrome decoding algorithm.

Strategy overview. Let $C \subseteq \mathbb{F}_2^n$ be a binary linear code of length n and dimension k, with parity-check matrix $H \in \mathbb{F}_2^{(n-k)\times n}$. Given a syndrome $s = Hx^\top$ corresponding to an unknown error vector $x \in \mathbb{F}_2^n$ of Hamming weight w, the objective is to recover x.

We assume that the adversary obtains coordinate-wise leakage observations L_i, from which it derives:

- Posterior probabilities

$$p_i := \Pr(x_i = 1 \mid L_i).$$

- A bitwise MAP estimate

$$\hat{x}_i := \mathbf{1}_{\{p_i \geq 0.5\}}.$$

- Posterior correctness probabilities

$$q_i := \Pr(x_i = \hat{x}_i) = \max\{p_i, 1 - p_i\}.$$

4.2.1. Decoding framework

The proposed decoding procedure operates as follows:

(1) **Subset selection.** An enumerator $\mathcal{E}$ generates candidate subsets $I \subseteq [n]$ of fixed size f, according to a ranking policy based on posterior reliability (e.g., descending aggregate confidence).

(2) **Instance reduction.** For each candidate subset I, the decoder assumes that the corresponding MAP estimates are correct, i.e.,

$$x_I = \hat{x}_I.$$

Let

$$J := [n] \setminus I.$$

The residual syndrome and target weight are then updated as

$$s' := s - H_I \hat{x}_I^\top,$$
$$w' := w - \mathrm{wt}(\hat{x}_I).$$

Let H_J denote the submatrix of H restricted to the columns indexed by J. Since H_J may contain linearly dependent rows, the decoder applies row reduction to obtain

$$UH_J = \begin{bmatrix} H_J^{(r)} \\ 0 \end{bmatrix}, \qquad Us' = \begin{bmatrix} s_1'' \\ s_2'' \end{bmatrix},$$

where $H_J^{(r)}$ has full row rank r, and necessarily $s_2'' = 0$. The resulting reduced decoding instance is therefore

$$H_J^{(r)} x_J^\top = s_1'', \qquad \mathrm{wt}(x_J) = w'.$$

(3) **Decoding and reconstruction.** The reduced instance is solved using a generic syndrome decoding algorithm. If decoding succeeds, the complete error vector is reconstructed as

$$x_i := \begin{cases} \hat{x}_i & \text{if } i \in I, \\ (x_J)_i & \text{if } i \in J. \end{cases}$$

4.2.2. Choosing the subset size f

The parameter f determines the number of coordinates assumed correct in the MAP estimate. Choosing f involves a trade-off: Larger values lead to smaller reduced decoding instances, but also increase the probability of conditioning on incorrect estimates, potentially rendering the reduced instance inconsistent.

To balance these effects, we estimate the expected number of correctly recovered coordinates as

$$R := \sum_{i=1}^{n} q_i = \mathbb{E}\left[\sum_{i=1}^{n} \mathbf{1}_{\{x_i=\hat{x}_i\}}\right],$$

where

$$q_i = \Pr(x_i = \hat{x}_i)$$

denotes the posterior correctness probability at coordinate i. Given a confidence parameter $\tau \in (0, 1)$, we define

$$f := \min\left(\lfloor \tau R \rfloor,\ k\right).$$

This choice selects a conservative fraction of the expected reliable coordinates while ensuring that the conditioned subset does not exceed the code dimension k, thereby preserving a valid reduction to a syndrome decoding instance.

4.2.3. Enumerator instantiation

The subset enumerator $\mathcal{E}$ is a central component of the proposed framework. Its purpose is to generate candidate subsets $I \subseteq [n]$ of fixed size f, prioritizing those whose associated MAP estimates are most likely to be correct according to the posterior model. Importantly, the enumerator is not fixed: It acts as an abstract search module that can be instantiated using different strategies depending on the computational budget, coverage requirements, or desired success guarantees.

Representative instantiations include:

- **Greedy:** Select the top-f coordinates with the largest posterior correctness probabilities q_i.
- **Exhaustive likelihood:** Enumerate all subsets of size f in decreasing order of joint confidence $\prod_{i\in I} q_i$, whenever exhaustive search is computationally feasible.
- **Randomized:** Generate subsets using posterior-biased randomized procedures, such as weighted sampling, genetic algorithms, or posterior-guided Markov chains.
- **Uniform:** Sample subsets uniformly at random, recovering a Prange-like information-set selection strategy.

Algorithm 1 summarizes the proposed recursive decoding framework. At each recursion level, the decoder selects a candidate subset I, conditions on the corresponding MAP estimates, and constructs a reduced decoding instance. The procedure is then recursively applied until a prescribed depth D is reached, at which point a generic syndrome decoder is invoked on the final reduced instance.

At each recursive stage, the enumerator $\mathcal{E}$ is reinitialized using the updated posterior scores, MAP estimates, and reduced decoding instance. This allows the subset-selection process to adapt dynamically to the evolving decoding problem.

The computational cost of the framework depends heavily on the choice of enumerator $\mathcal{E}$. Exhaustive likelihood-based enumeration requires evaluating all $\binom{n}{f}$ candidate subsets and quickly becomes infeasible for large parameters. Randomized strategies, such as posterior-guided sampling or genetic algorithms, reduce this cost by exploring only a fraction of the search space.

A uniform enumerator ignores the posterior information derived from leakage and recovers a classical random information-set selection strategy. In contrast, posterior-guided enumerators exploit

coordinate-wise confidence scores to prioritize subsets that are more likely to satisfy the assumptions required for successful decoding. By conditioning on high-confidence coordinates, the framework reduces the effective decoding instance before invoking the underlying decoder, potentially lowering the expected decoding complexity under informative leakage conditions.

Algorithm 1: Recursive posterior-guided decoding.

Require: $H \in \mathbb{F}_2^{(n-k)\times n}$; syndrome $s \in \mathbb{F}_2^{n-k}$; target weight w; code dimension k; MAP estimate $\hat{x}$; posterior correctness probabilities $q_i := \Pr(x_i = \hat{x}_i)$; recursion depth d; maximum depth D; confidence parameter τ

Ensure: $x \in \mathbb{F}_2^n$ such that $Hx^\top = s$ and $\mathrm{wt}(x) = w$, or failure

$n \leftarrow$ length of $\hat{x}$
$R \leftarrow \sum_i q_i$
$f \leftarrow \min(\lfloor \tau R \rfloor, k)$
Instantiate enumerator $\mathcal{E}$ over subsets $I \subseteq [n]$ of size f
for each $I \in \mathcal{E}$ **do**
  $J \leftarrow [n] \setminus I$
  Construct submatrices H_I and H_J
  $s' \leftarrow s - H_I \hat{x}_I^\top$
  $w' \leftarrow w - \mathrm{wt}(\hat{x}_I)$
  Apply row reduction:

$$UH_J = \begin{bmatrix} H_J^{(r)} \\ 0 \end{bmatrix}, \qquad Us' = \begin{bmatrix} s_1'' \\ 0 \end{bmatrix}$$

  Restrict posterior scores and MAP estimates to J
  if $d = D$ **then**
    Attempt to decode x_J from $(H_J^{(r)}, s_1'', w')$
    if successful **then**
      return Reconstruct x from $\hat{x}_I$ and x_J
    end if
  else
    Recursively decode:

$$x_J \leftarrow \textsc{RecursiveDecode}(H_J^{(r)}, s_1'', w', \hat{x}_J, q_J, d+1, D, \tau)$$

    if $x_J \neq$ failure **then**
      return Reconstruct x from $\hat{x}_I$ and x_J
    end if
  end if
end for
return failure

4.3. Leakage in CBA

To illustrate a practical instantiation of posterior-guided decoding, we consider CBAs, where the leakage L_i arises from decayed memory contents. In this setting, each observation $\tilde{x}_i \in \{0, 1\}$

corresponds to a noisy version of the true error bit x_i, affected by asymmetric decay dynamics. The framework models this process through the asymmetric transition probabilities

$$\Pr(\tilde{x}_i \mid x_i) = \begin{cases} 1-\beta, & \text{if } x_i = 1,\ \tilde{x}_i = 1, \\ \beta, & \text{if } x_i = 1,\ \tilde{x}_i = 0, \\ 1-\alpha, & \text{if } x_i = 0,\ \tilde{x}_i = 0, \\ \alpha, & \text{if } x_i = 0,\ \tilde{x}_i = 1. \end{cases}$$

Let $\rho := \frac{w}{n}$ denote the prior probability that $x_i = 1$. Applying Bayes' rule yields the posterior probabilities:

- If $\tilde{x}_i = 0$:

$$\Pr(x_i = 0 \mid \tilde{x}_i = 0) = \frac{(1-\alpha)(1-\rho)}{(1-\alpha)(1-\rho)+\beta\rho},$$
$$\Pr(x_i = 1 \mid \tilde{x}_i = 0) = \frac{\beta\rho}{(1-\alpha)(1-\rho)+\beta\rho}.$$

- If $\tilde{x}_i = 1$:

$$\Pr(x_i = 0 \mid \tilde{x}_i = 1) = \frac{\alpha(1-\rho)}{\alpha(1-\rho)+(1-\beta)\rho},$$
$$\Pr(x_i = 1 \mid \tilde{x}_i = 1) = \frac{(1-\beta)\rho}{\alpha(1-\rho)+(1-\beta)\rho}.$$

These posterior values determine the confidence scores q_i used to guide subset selection within the decoding framework. CBAs therefore provide a concrete and practically motivated instantiation of the BBLM.

4.4. Practical implications and limitations

The proposed framework illustrates how partial and noisy leakage can reduce the effective complexity of syndrome decoding through posterior-guided conditioning and recursive instance reduction. Rather than replacing existing ISD techniques, the approach acts as a complementary layer that can be combined with different decoders, subset-enumeration strategies, and code families.

The effectiveness of the framework depends primarily on the quality of the posterior estimates and the ability of the enumerator to identify reliable coordinate subsets. Highly noisy or poorly calibrated leakage may lead to incorrect conditioning assumptions and inconsistent reduced instances, while inefficient enumeration strategies may fail to exploit the available posterior information effectively. The framework also assumes conditional independence of leakage observations, an approximation commonly adopted in cold boot attack models but not always valid in correlated side-channel settings.

Although the framework does not alter the asymptotic complexity class of ISD, it can reduce the effective decoding effort by shrinking the syndrome decoding instance before invoking the underlying decoder. The resulting gains therefore depend jointly on the informativeness of the leakage, the quality of the subset-enumeration strategy, and the structure of the underlying code.

Countermeasures and practical feasibility. The practical impact of posterior-guided decoding depends on the availability of sufficiently informative leakage. Countermeasures such as memory

encryption [50], masking [51], memory sanitization [52], and data re-randomization [53] reduce the reliability of posterior estimates and consequently weaken the effectiveness of the attack.

Therefore, posterior-guided decoding is most relevant in settings where residual leakage remains sufficiently informative to support reliable posterior estimation and effective subset prioritization.

5. Probabilistic analysis of f-subset enumeration

We analyze the probabilistic behavior of f-subset enumeration under fixed coordinate-wise correctness probabilities derived from posterior leakage estimates. In this model, each coordinate is associated with a correctness probability that remains constant throughout the enumeration process, and the outcome of one trial does not affect subsequent trials.

This abstraction provides a tractable framework for studying posterior-guided subset selection strategies within the proposed decoding framework. In particular, it enables the derivation of exact success expressions, iteration estimates, and probabilistic interpretations of different enumeration strategies.

5.1. Probability space and success events

Let $(\Omega, \mathcal{F}, \Pr)$ be a finite probability space where

$$\Omega := \{0,1\}^n, \quad \omega = (\omega_1, \ldots, \omega_n)$$

represents coordinate-wise correctness patterns. Here, $\omega_i = 1$ indicates that the ith coordinate estimate is correct, while $\omega_i = 0$ indicates an incorrect estimate.

For each coordinate $i \in [n]$, define an independent Bernoulli random variable

$$Y_i : \Omega \to \{0,1\}, \quad \Pr(Y_i = 1) = p_i,$$

where $p_i \in [0,1]$ denotes the posterior correctness probability associated with coordinate i. Independence induces the product measure

$$\Pr(\omega) = \prod_{i=1}^{n} p_i^{\omega_i}(1-p_i)^{1-\omega_i}.$$

For an f-subset $I \subseteq [n]$, define the event

$$A(I) := \bigcap_{i \in I} \{Y_i = 1\},$$

corresponding to the event that all coordinates indexed by I are correctly estimated. By independence,

$$\Pr(A(I)) = \prod_{i \in I} p_i =: p(I).$$

Suppose an enumerator outputs subsets $I_1, \ldots, I_m$, and define

$$A_t := A(I_t).$$

The overall success event is

$$\text{Success} := \bigcup_{t=1}^{m} A_t,$$

that is, at least one enumerated subset consists entirely of correct coordinates.

By inclusion-exclusion,

$$\Pr(\text{Success}) = \sum_{\emptyset \neq T \subseteq [m]} (-1)^{|T|+1} \Pr\Big(\bigcap_{t \in T} A_t \Big).$$

Since

$$\bigcap_{t \in T} A_t = \bigcap_{i \in \bigcup_{t \in T} I_t} \{Y_i = 1\},$$

independence yields

$$\Pr(\text{Success}) = \sum_{\emptyset \neq T \subseteq [m]} (-1)^{|T|+1} \prod_{i \in \bigcup_{t \in T} I_t} p_i.$$

This expression shows that the success probability depends not only on the quality of each subset, measured by $p(I_t)$, but also on the overlap structure among the enumerated subsets.

5.2. Disjoint subsets and independent success events

Exact inclusion-exclusion becomes impractical for large m. A particularly important case arises when the subsets $I_1, \ldots, I_m$ are pairwise disjoint. In this setting, the events $A_1, \ldots, A_m$ are mutually independent, yielding

$$\Pr(\text{Success}) = 1 - \prod_{t=1}^{m} (1 - \Pr(A_t)) = 1 - \prod_{t=1}^{m} \left(1 - \prod_{i \in I_t} p_i \right).$$

In the homogeneous case $p_i = p$, this simplifies to

$$\Pr(\text{Success}) = 1 - (1 - p^f)^m.$$

This characterization highlights an important trade-off in posterior-guided enumeration: Selecting high-confidence subsets improves the individual probabilities $p(I_t)$, while maintaining low overlap increases search diversity and improves overall coverage.

5.3. Bounds and approximations

Exact evaluation of Pr(success) requires summing $2^m - 1$ inclusion-exclusion terms and quickly becomes infeasible. We therefore consider several useful approximations and bounds.

5.3.1. Union bound

By Boole's inequality,

$$\Pr(\text{Success}) \leq \sum_{t=1}^{m} p(I_t).$$

This bound is exact when the events A_t are disjoint and may be loose otherwise.

5.3.2. Independence approximation

Treating the events A_t as approximately independent yields

$$\Pr(\text{Success}) \approx 1 - \prod_{t=1}^{m}(1 - p(I_t)).$$

This approximation becomes accurate when overlap among subsets is limited.

5.3.3. Pairwise-overlap lower bound

A second-order inclusion-exclusion bound gives

$$\Pr(\text{Success}) \geq \sum_{t=1}^{m} p(I_t) - \sum_{1\leq s<t\leq m} \prod_{i\in I_s\cup I_t} p_i.$$

This expression explicitly captures the negative effect of overlap among candidate subsets.

5.4. Enumerator interpretations

The proposed framework does not prescribe a specific subset enumerator. Instead, different enumeration strategies induce different probabilistic behaviors depending on the trade-off between subset quality and search diversity.

5.4.1. Uniform sampling

Uniform random sampling over

$$\binom{n}{f}$$

promotes broad coverage but ignores posterior information.

5.4.2. Greedy strategies

Greedy selection prioritizes subsets with large values of $p(I)$, but may repeatedly select highly overlapping subsets, reducing effective exploration.

5.4.3. Randomized posterior-guided search

Randomized procedures, such as posterior-weighted sampling or genetic algorithms, attempt to balance subset quality and diversity. Their effectiveness depends critically on avoiding excessive overlap while still prioritizing reliable coordinates.

5.4.4. Exhaustive enumeration

Exhaustive search tests all f-subsets and succeeds whenever at least f coordinates are correct:

$$\Pr(\text{Success}) = \Pr\left(\sum_{i=1}^{n} Y_i \geq f\right).$$

For homogeneous probabilities $p_i = p$, this becomes the binomial tail

$$\sum_{k=f}^{n} \binom{n}{k} p^k (1-p)^{n-k}.$$

For heterogeneous probabilities p_i, the distribution becomes Poisson–binomial.

5.4.5. Optimal static enumeration

Under the static product model, if $mf \leq n$, one can select m pairwise disjoint subsets, yielding independent success events. In this regime, optimal strategies prioritize coordinates with the largest posterior correctness probabilities.

If $mf > n$, disjointness becomes impossible, and the resulting optimization problem reduces to a weighted set-packing problem, which is NP-hard.

5.5. Iteration estimates

Let

$$q_t := p(I_t).$$

Under the independence approximation,

$$\Pr(\text{Success}) = 1 - \prod_{t=1}^{m} (1 - q_t).$$

In the homogeneous case $q_t = q$, achieving

$$\Pr(\text{Success}) \geq \frac{1}{2}$$

requires

$$m \geq \left\lceil \frac{\ln(1/2)}{\ln(1-q)} \right\rceil.$$

For small probabilities $q_t \ll 1$, using

$$\ln(1 - q_t) \approx -q_t,$$

gives the approximation

$$\Pr(\text{Success}) \approx 1 - e^{-\sum_{t=1}^{m} q_t}.$$

Consequently, a useful heuristic condition for constant success probability is

$$\sum_{t=1}^{m} q_t \gtrsim \ln 2.$$

Without assuming independence, the second-order inclusion–exclusion inequality yields

$$\Pr(\text{Success}) \geq \sum_{t=1}^{m} q_t - \sum_{1 \leq s < t \leq m} \prod_{i \in I_s \cup I_t} p_i.$$

These estimates illustrate that the effectiveness of posterior-guided enumeration depends jointly on subset reliability and search diversity. High-confidence subsets alone are insufficient if the enumeration process repeatedly explores highly overlapping regions of the search space.

6. A genetic-algorithm enumerator

Genetic algorithms have previously been used as heuristic optimization tools in combinatorial decoding contexts [54]. In the present work, we use a genetic algorithm as one possible enumerator within the broader posterior-guided decoding framework. Its role is to explore candidate conditioned subsets using leakage-derived posterior information, rather than to define a new decoding algorithm.

In our framework, the leakage model provides coordinate-wise reliability scores, but the decoder still requires an effective method for selecting high-quality subsets. Exhaustive enumeration is infeasible for large parameters because it requires searching through

$$\binom{n}{f}$$

subsets. We therefore instantiate the enumerator as a genetic algorithm, using posterior confidence as a fitness signal to explore candidate subsets without enumerating the full search space.

Unlike the classical Prange framework, which seeks positions that are likely error-free, the present approach aims to identify bits that are most likely to be *correct*, regardless of whether they are 0 or 1. Within the BBLM, this corresponds to selecting reliable positions from the MAP estimate $\hat{x} = (\hat{x}_1, \ldots, \hat{x}_n)$, where

$$q_i = \Pr(x_i = \hat{x}_i)$$

denotes the posterior correctness probability of bit i. The objective is therefore to identify a subset $I \subset \{1, \ldots, n\}$ of size f maximizing the aggregate posterior reliability:

$$\texttt{Score}(I) = \sum_{i \in I} \log q_i,$$

which maximizes the aggregate posterior confidence of the selected subset under the independence assumptions of the BBLM.

To address this combinatorial optimization problem, we instantiate the enumerator as a genetic algorithm, summarized in Algorithm 2. The method evolves a population of candidate subsets across multiple generations using three operators: selection, crossover, and mutation, described in Algorithms 3–5. The objective is to maximize the aggregate posterior reliability score. The main stages are as follows:

(1) **Fitness evaluation.** Each subset I is evaluated using the posterior reliability function $F(I) := \texttt{Score}(I)$. Higher fitness values correspond to larger posterior confidence that the selected MAP estimates are correct (lines 4–7 in Algorithm 2).

(2) **Population initialization.** The initial population $\mathcal{P}_0$ is generated by sampling P random subsets of size f, optionally biased toward indices with larger posterior probabilities q_i (line 1 in Algorithm 2).

(3) **Selection (Algorithm 3).** Parent selection uses tournament sampling: Several candidate subsets are drawn from the population, and the subset with the highest fitness is selected as a parent.

(4) **Crossover (Algorithm 4).** The crossover operator combines two parent subsets I_1 and I_2 to produce an offspring I_{child}, allowing reliable elements to be inherited from both parents.

(5) **Mutation (Algorithm 5).** Each offspring undergoes mutation with probability μ, replacing one subset element with a randomly selected unused index. This introduces variation and prevents premature convergence.

(6) **Elitism and termination.** The best-performing subsets are carried over to the next generation. After T generations, the algorithm outputs the subset $I^\star$ with the highest fitness score (lines 8–12 in Algorithm 2).

Algorithm 2: Genetic algorithm for posterior-guided information set search.

Require: Posterior scores $q_i \leftarrow \Pr(x_i = \hat{x}_i)$ for all $i \in \{1, \ldots, n\}$; subset size f; population size P; elite size E; number of generations T; crossover probability $p_c \in (0, 1)$; mutation rate μ.

Ensure: Best subset $I^\star$ and its fitness value

1: Initialize population $\mathcal{P}_0 = \{I^{(1)}, \ldots, I^{(P)}\}$, where each $I^{(j)}$ is a random f-subset of $\{1, \ldots, n\}$
2: Set $I^\star \leftarrow \emptyset$, best $\leftarrow -\infty$
3: **for** $t = 1$ **to** T **do**
4: **for all** $I \in \mathcal{P}_{t-1}$ **do**
5: Compute fitness $F(I)$
6: **end for**
7: Select the top E individuals by fitness and store them in $\mathcal{E}$
8: **if** $\max_{I \in \mathcal{P}_{t-1}} F(I) >$ best **then**
9: Update best and $I^\star$
10: **end if**
11: Select a parent pool via tournament selection (Algorithm 3)
12: Initialize next generation $\mathcal{P}_t \leftarrow \mathcal{E}$
13: **while** $|\mathcal{P}_t| < P$ **do**
14: Draw $r \sim \mathcal{U}[0, 1]$
15: **if** $r < p_c$ **then**
16: Select parents I_1, I_2 from the parent pool
17: $I_{\text{child}} \leftarrow \textsc{Crossover}(I_1, I_2, f, n)$
18: **else**
19: $I_{\text{child}} \leftarrow$ copy of a randomly selected parent
20: **end if**
21: $I_{\text{child}} \leftarrow \textsc{Mutation}(I_{\text{child}}, n, \mu)$
22: Add I_{child} to $\mathcal{P}_t$
23: **end while**
24: **end for**
25: **return** $I^\star$, best

Algorithm 3: Tournament selection.

Require: Population $\mathcal{P}$, tournament size s
Ensure: Selected parent pool $\mathcal{M}$
Initialize $\mathcal{M} \leftarrow \emptyset$
for $j = 1$ **to** $|\mathcal{P}|$ **do**
  Sample a tournament set $\mathcal{T} \leftarrow \textsc{Sample}(\mathcal{P}, s)$ **without replacement**
  $I^\star \leftarrow \arg\max_{I \in \mathcal{T}} F(I)$
  $\mathcal{M} \leftarrow \mathcal{M} \cup \{I^\star\}$
end for
return $\mathcal{M}$

Algorithm 4: Crossover.

Require: Parent subsets I_1, I_2 of size f; total index range length n
Ensure: Child subset I_{child} of size f
$U \leftarrow I_1 \cup I_2$
if $|U| > f$ **then**
  Randomly select f indices from U (uniformly, without replacement)
  $I_{\text{child}} \leftarrow$ selected indices
else
  $I_{\text{child}} \leftarrow U$
end if
return I_{child}

Algorithm 5: Random mutation.

Require: Subset I of size f, total size n, mutation rate μ
Ensure: Mutated subset I'
Draw random number $r \sim \mathcal{U}[0, 1]$
if $r > \mu$ **then**
  return I
end if
Define $A \leftarrow \{0, \ldots, n-1\} \setminus I$
Select $j \in A$ uniformly at random
Select $i \in \{1, \ldots, f\}$ uniformly at random
Replace $I[i]$ with j
return I

These operators explore the subset space to identify high-posterior-reliability subsets, which are then used by posterior-guided decoders such as guided Prange. The modular structure of the algorithm makes it compatible with different leakage models, decoding objectives, and subset-selection policies.

The genetic algorithm should therefore be viewed as a heuristic enumerator rather than a standalone decoding algorithm. It is evaluated against other heuristic strategies, such as greedy search, rather than against a global optimality criterion. Its objective is not to consistently recover perfectly correct subsets, but to efficiently identify high-quality approximations containing many reliable positions. Such approximations are sufficient to improve the effectiveness of subsequent decoding stages.

7. Results

This section presents the experimental evaluation of the proposed posterior-guided decoding framework and its genetic-algorithm-based enumerator introduced in Section 6. We consider two

code families: random linear codes and RM codes.

7.1. Ablation studies for the genetic algorithm

We analyze how the main genetic parameters, population size, number of generations, mutation rate, and subset size f, affect decoding performance under simulated cold boot leakage conditions. These experiments evaluate the convergence and robustness of the proposed heuristic enumerator and compare it against a greedy search baseline for subset selection.

We also study how structural properties of the decoding problem influence performance, including the code length N, the Hamming weight w, the leakage bit-flip probabilities α and β, and the subset size f.

7.1.1. Experimental setup for genetic algorithm parameters

Each experimental configuration is specified by a pair (N, w), as listed in Table 1. For each configuration, the code dimension is fixed to $k = N/2$, and leakage is simulated using the asymmetric bit-flip probabilities of the cold boot attack model:

- $\alpha = 0.01$ for $0 \to 1$ flips.
- $\beta = 0.20$ for $1 \to 0$ flips.

Table 1. Parameter configurations explored in the posterior-guided decoding experiments.

Parameter	Description	Values/range tested
N	Code length	60, 124, 260, 340, 460, 510, 770, 1000
w	Hamming weight	10, 17, 33, 42, 56, 62, 93, 119
k	Code dimension	$N/2$
f	Fraction of k used in selection	$0.1k$–$1.0k$ (step $0.1k$)
α	Probability of $0 \to 1$ flip	0.01
β	Probability of $1 \to 0$ flip	0.20
Generations	Evolutionary horizon	100–5000 (step 500)
Population size	Individuals per generation	50
Trials per configuration	Independent decoding attempts	10
Instances per (N, w)	Independent leakage samples	10

For each (N, w) pair, ten independent leakage instances and ten decoding trials were generated for every parameter configuration. A trial was considered successful when the selected subset I satisfied $e_I = \tilde{e}_I$, meaning that all bits in the subset remained unchanged after leakage. The average success rate across instances was used as the primary performance metric.

7.1.2. Heuristic comparison: genetic vs. greedy search

For each leakage realization, we compare the information set selected by the genetic algorithm with a greedy baseline. The greedy strategy selects the f indices with the largest individual posterior

probabilities, whereas the genetic algorithm optimizes the aggregate posterior score over subsets of size f, enabling global exploration of candidate information sets.

We conducted a grid search over the genetic hyperparameters: population sizes from 10 to 80, generations from 50 to 200, and mutation rates from 0.1 to 0.5. The leakage parameters were varied as $\alpha \in [0.001, 0.005]$ and $\beta \in \{0.05, 0.10, 0.15, 0.20\}$. For each configuration (N, w, α, β) and hyperparameter setting, we evaluated 30 independent leakage instances.

Among the tested configurations, we observe parameter settings for which the genetic algorithm outperforms the greedy baseline across the considered (N, w) values. These settings are used to generate the comparative results shown in Figure 1.

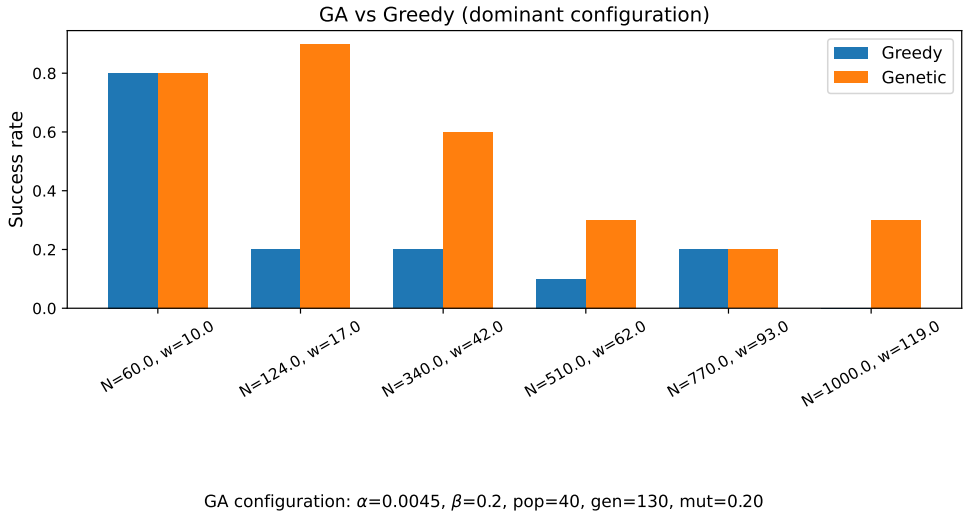


Figure 1. Exact recovery rate comparison between greedy selection and the proposed genetic algorithm for a fixed subset size $f = 0.2k$, under a representative hyperparameter configuration for which the genetic approach outperforms greedy across all tested (N, w).

In all experiments, the subset size is fixed to $f = 0.2k$, where $k = N/2$ denotes the code dimension. Figure 1 reports the average exact recovery rate, defined as the fraction of experiments in which the selected information set I satisfies $e_I = \tilde{e}_I$.

The results show that greedy selection is a strong and computationally efficient baseline. Nevertheless, suitable genetic configurations can exploit mutation and crossover to identify higher-quality subsets, particularly when coordinate-wise ranking alone is insufficient.

7.1.3. Number of generations

The number of generations was varied from 100 to 5000 in increments of 500, with the population size fixed at 50. This experiment evaluates how evolutionary depth affects the quality of the selected subsets and the convergence behavior of the heuristic enumerator.

As shown in Figure 2, increasing the number of generations does not produce a clear monotonic improvement in the exact recovery rate. The smallest configuration, $(N{=}60, w{=}10)$, reaches a relatively stable recovery rate near 0.28, whereas larger configurations remain close to zero or fluctuate irregularly across the tested range.

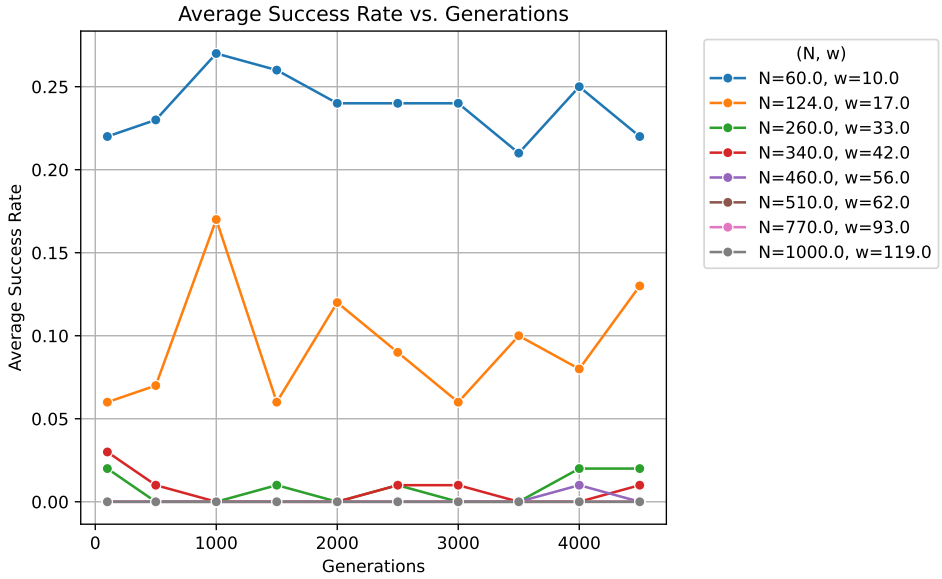


Figure 2. Average exact recovery rate as a function of the number of generations for multiple (N, w) configurations, with population size fixed at 50. Each point represents the mean across 10 leakage instances and 10 enumeration trials.

These results suggest that the genetic enumerator rapidly converges toward a stable set of high-posterior candidate subsets, after which additional generations provide limited improvement. In practice, the search tends to stabilize after a few hundred generations, indicating that the evolutionary dynamics quickly reach a local equilibrium under the posterior-based fitness landscape.

This behavior can be partially explained by the structure of the posterior scoring function. Since the fitness score aggregates coordinate-wise posterior confidences, many distinct subsets may obtain very similar scores despite differing in exact correctness. In particular, positions assigned large posterior confidence by the leakage model tend to dominate the evolutionary process, even when some of these positions are incorrect due to rare leakage events.

Consequently, the genetic algorithm often converges to subsets with high aggregate posterior reliability rather than to perfectly correct subsets. This effect becomes more pronounced as the subset size increases, since the probability that all selected coordinates are simultaneously correct decreases with the dimension of the conditioned subset.

This saturation should therefore not be interpreted as a failure of the posterior-guided enumerator. Rather, it reflects the strict success criterion adopted in the experiments, which requires all coordinates in the selected subset to match the true error vector. In practice, subsets containing a small number of incorrectly conditioned positions may still substantially reduce the effective decoding effort of the subsequent syndrome decoding stage.

7.1.4. Population size

The population size was varied from 50 to 500 in increments of 50, with the number of generations fixed at 100. This experiment evaluates whether larger populations improve success rates through increased diversity or mainly add computational overhead.

As shown in Figure 3, the success rate remains relatively low, typically below 0.35, and does not display a consistent increasing trend as the population size grows. Thus, within the explored range of 50–500 individuals, population scaling alone does not significantly improve the genetic algorithm's ability to identify the correct subset.

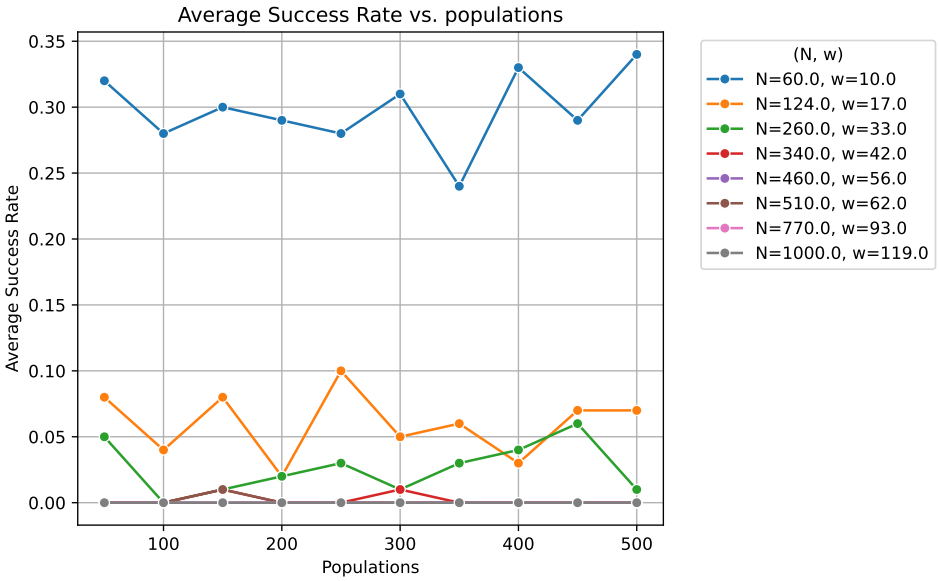


Figure 3. Average success rate as a function of population size for multiple (N, w) configurations. The number of generations was fixed at 100.

The flat trend observed across configurations suggests that the genetic diversity introduced by larger populations is not effectively translated into higher-quality solutions, possibly due to early convergence or limited evolutionary depth. This stagnation indicates that the search space may be too rugged or too constrained for population size alone to be the primary factor influencing performance.

Instead, the algorithm's progress may depend more strongly on other design choices, such as evolutionary depth, mutation strength, or adaptive genetic operators, which directly affect exploration capacity and convergence behavior. Further experiments with extended evolutionary horizons or adaptive operators could help determine whether the observed plateau arises from intrinsic structural limits of the posterior-guided search or from premature convergence dynamics.

7.1.5. Mutation rate

The mutation rate was varied from 0.1 to 0.9 in increments of 0.1, with population size fixed at 50 and the number of generations fixed at 80. This experiment evaluates how stochastic perturbation affects the quality of the selected subsets in the posterior-guided enumeration process.

As shown in Figure 4, the exact recovery rate remains relatively stable across the tested mutation range. Larger values of N show slight improvement with higher mutation rates, whereas smaller instances exhibit mild degradation.

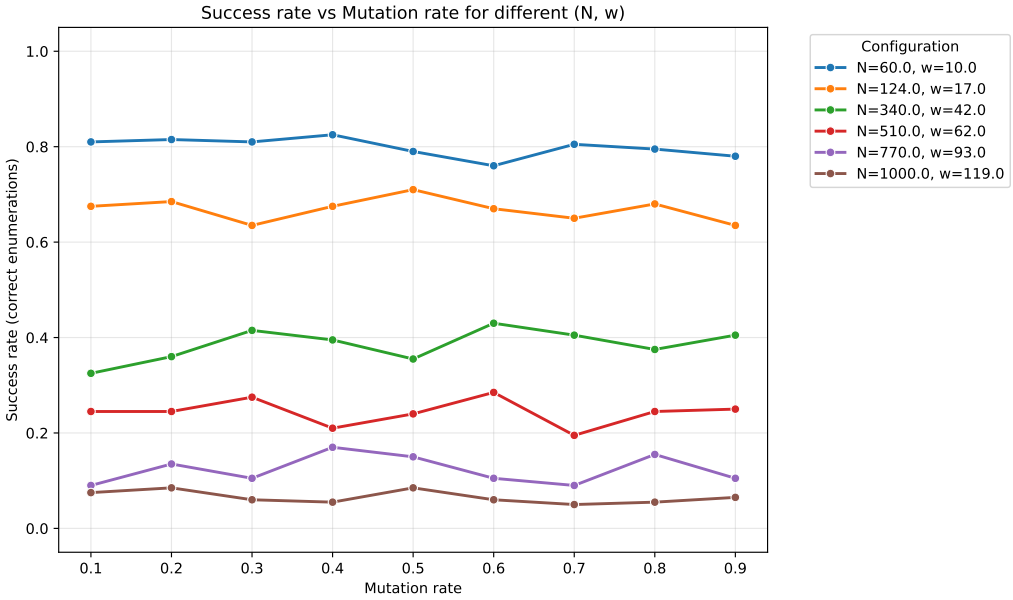


Figure 4. Average exact recovery rate as a function of the mutation rate for multiple (N, w) configurations. The population size was fixed at 50 and the number of generations at 80.

For larger instances, increased stochastic exploration helps the genetic enumerator identify subsets with larger aggregate posterior reliability. For smaller search spaces, however, excessive randomness may disrupt high-quality candidate subsets already present in the population.

Overall, mutation plays a secondary role in the evolutionary process. Its effect becomes more visible as the search space grows, but the overall performance remains primarily determined by the quality of the posterior information provided by the leakage model.

7.1.6. Subset size f and noise sensitivity

The final set of experiments focuses on the subset size f. We examine how f affects the quality of the selected conditioned subsets and how the posterior-guided enumerator responds to variations in the leakage noise level β. Increasing f produces smaller reduced instances, but also increases the risk of conditioning on incorrect positions. Varying β allows us to evaluate how enumeration quality deteriorates as leakage becomes noisier.

Effect of selected subset fraction f. We first study how the selected subset size f, expressed as a percentage of k, affects the exact recovery rate of the genetic enumerator. Figure 5 reports the average exact recovery rate across multiple (N, w) configurations as f varies from 10% to 100% of k. The curves exhibit a concave behavior: Performance improves as f increases up to an intermediate range, after which it saturates or declines.

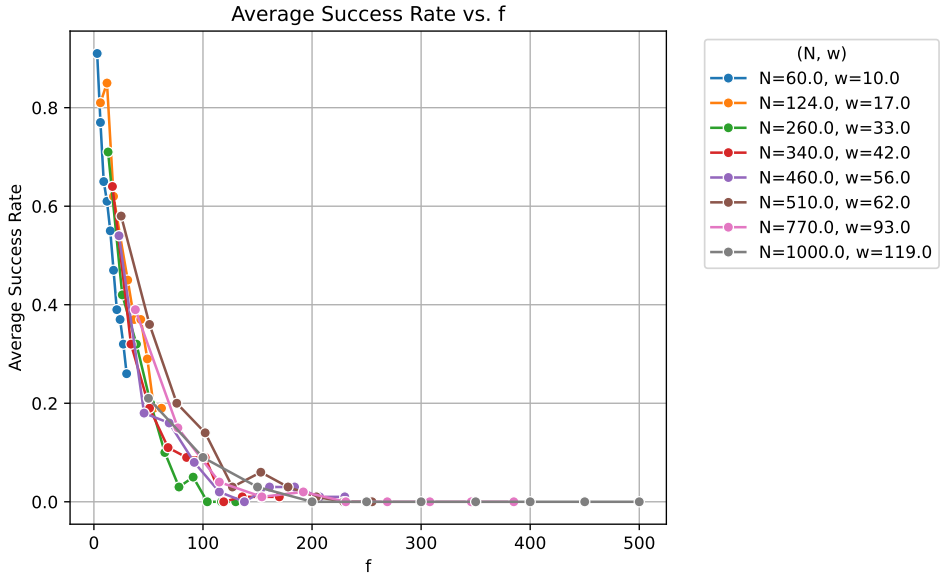


Figure 5. Average exact recovery rate as a function of the selected subset size f across multiple (N, w) configurations. Each point represents the mean over 10 leakage instances and 10 enumeration trials.

This behavior reflects a trade-off between instance reduction and conditioning reliability. Small values of f provide limited reduction of the decoding instance, whereas larger values increase the probability that at least one conditioned coordinate is incorrect. Consequently, large subset sizes may degrade exact recovery performance despite producing smaller reduced instances.

These results suggest that posterior-guided enumeration benefits from moderate subset sizes, which balance effective instance reduction against the accumulation of conditioning errors. More generally, the experiments support the central hypothesis of the framework: Informative posterior leakage can guide subset selection toward conditioned instances that are substantially easier to decode than uniformly selected subsets.

Effect of leakage parameters (α, β)**.** To evaluate robustness under realistic cold boot leakage conditions, we jointly vary the asymmetric bit-flip probabilities (α, β). We consider $\alpha \in \{0.001, 0.005, 0.01, 0.02\}$ for $0 \to 1$ flips and $\beta \in \{0.05, 0.10, 0.15, 0.20\}$ for $1 \to 0$ flips. All other parameters were fixed at their default values: population size 50, 80 generations, and $f = 0.5k$. Performance metrics were averaged over all evaluated (N, w) configurations.

Table 2 reports the average exact recovery rate and runtime for each (α, β) pair. The results show that enumeration quality degrades consistently as β increases, with comparatively weaker dependence on α. Across all tested values of α, exact recovery rates exceed 40% for $\beta = 0.05$, but decrease substantially once $\beta \geq 0.15$.

In contrast, variations in α have a milder impact on performance. For low to moderate values of β, increasing α from 0.001 to 0.01 does not significantly affect the recovery rate and may occasionally improve performance by reducing excessively concentrated posterior scores. This behavior reflects the dependence of the posterior-guided search process on the reliability of the leakage-derived posterior information. As β increases, the leakage observations become progressively less informative, reducing

the ability of the enumerator to identify highly reliable conditioned subsets. In this regime, misleading posterior estimates accumulate and the effectiveness of posterior-guided subset selection decreases.

Table 2. Average decoding performance aggregated over all (N, w) configurations for different leakage parameters (α, β).

α	β	Exact recovery rate	Runtime (s)
0.001	0.05	0.4333	0.3168
0.001	0.10	0.3733	0.3582
0.001	0.15	0.3133	0.3328
0.001	0.20	0.2400	0.2905
0.005	0.05	0.4967	0.2350
0.005	0.10	0.3167	0.2497
0.005	0.15	0.2167	0.2598
0.005	0.20	0.1833	0.2600
0.010	0.05	0.5167	0.2580
0.010	0.10	0.3267	0.2634
0.010	0.15	0.2300	0.2596
0.010	0.20	0.1933	0.2538
0.020	0.05	0.4700	0.2718
0.020	0.10	0.3200	0.2875
0.020	0.15	0.2167	0.2849
0.020	0.20	0.1600	0.3023

Across the tested configurations, the enumerator maintains non-negligible exact recovery rates for moderate leakage levels, particularly when $\beta \leq 0.05$. A weaker but still meaningful recovery regime persists up to approximately $\beta = 0.15$, with limited dependence on α.

Overall, the joint (α, β) analysis indicates that attack feasibility is primarily governed by β, while α plays a secondary role within realistic leakage ranges. Prior empirical studies on cold boot memory decay, including the seminal work of Halderman [37], report typical $0 \to 1$ flip probabilities on the order of $\alpha \approx 0.01$. This regime aligns with the region where the posterior-guided enumerator achieves its strongest and most stable performance, supporting both the relevance of the leakage model and the practical applicability of the proposed framework.

7.2. Bound estimation

In this section, we empirically estimate a bound on the number of iterations required by the decoding algorithm for the parameter sets considered in our experiments. Rather than deriving an asymptotic or worst-case theoretical bound, we characterize the typical behavior observed under the posterior-guided setting introduced in this work.

The overall complexity of the proposed algorithm is decomposed into the number of iterations required until success and the cost of a single iteration:

$$\text{Iterations} \times C_{\text{ISD}},$$

where C_{ISD} denotes the complexity of the chosen ISD algorithm.

Since all complexity estimates are reported in logarithmic scale, we use the base-2 logarithm of the overall decoding cost. The multiplicative decomposition then becomes

$$\log_2(\text{Iterations}) \; + \; \log_2(C_{\mathrm{ISD}}).$$

This representation facilitates empirical estimation of the iteration bound and direct comparison of decoding costs across parameter configurations.

The experiments in this section empirically estimate the average number of iterations required by the algorithm across the parameter configurations studied. By analyzing iteration counts over multiple instances and posterior configurations, we obtain a bound that reflects the typical convergence behavior of the algorithm rather than an adversarial worst-case scenario.

This iteration bound is then combined with the estimated cost of a single decoding iteration to estimate the overall decoding cost under the posterior-guided framework. The following subsections first present the bound estimation for Prange's algorithm and then compare the resulting costs with additional ISD variants and with the original, unreduced code parameters.

7.2.1. Quantile regression

In order to characterize the variability of the number of iterations required by the decoding algorithm, we go beyond average-case estimates and explicitly model the dispersion of the observed convergence behavior. Due to the stochastic nature of the posterior-guided decoding process, identical parameter configurations may exhibit significantly different outcomes across instances. Consequently, point estimates alone are insufficient to capture the effective decoding cost.

To this end, we introduce a stabilization parameter $\gamma \in [0, 1]$, defined as the ratio between the number of correctly identified positions in the set I and the total size of said set at any given iteration. That is,

$$\gamma \; = \; \frac{|\{i \in I \mid i \text{ is correctly recovered}\}|}{|I|}.$$

This quantity provides a direct measure of the quality of the recovered information set.

Empirically, we observe that after a certain number of iterations, the posterior-guided decoding process enters a *stabilization regime*. In this regime, the obtained set I consistently contains a fixed proportion of correct entries, and further iterations do not lead to a significant improvement in γ. That is, beyond a stabilization point, the algorithm continues to explore new information sets, but the number of correctly recovered positions remains approximately constant.

This stabilization phenomenon motivates the use of γ as an indicator for convergence: Although the algorithm may not immediately succeed, reaching a sufficiently high and stable value of γ indicates that the decoding process has approached its effective limit under the given parameter configuration. Consequently, γ serves as an informative indicator of how close the algorithm is to successful decoding and of the expected remaining number of iterations.

Rather than predicting a single expected value of γ, we employ *quantile regression* to estimate conditional bounds on the stabilization parameter given the system parameters. Let X denote the vector of parameters describing each decoding instance, including the code parameters, posterior parameters, and derived quantities related to the observed error pattern. For a given quantile level $\tau \in (0, 1)$,

quantile regression estimates the conditional quantile function $Q_\gamma(\tau \mid X)$ by minimizing the pinball loss.

In our experiments, we estimate the lower and upper conditional quantiles corresponding to $\tau = 0.1$ and $\tau = 0.9$, which define an empirical 80% prediction interval for the stabilization parameter. This formulation captures uncertainty in the stabilization behavior, reflecting the fact that the convergence plateau and its variability depend on the specific problem parameters.

Figure 6 illustrates the empirical behavior of the proposed quantile regression model. Each point represents a test instance, where the horizontal axis corresponds to the true value of the stabilization parameter γ, and the vertical axis shows the midpoint of the predicted quantile interval. The vertical error bars denote the prediction interval defined by the estimated lower and upper quantiles.

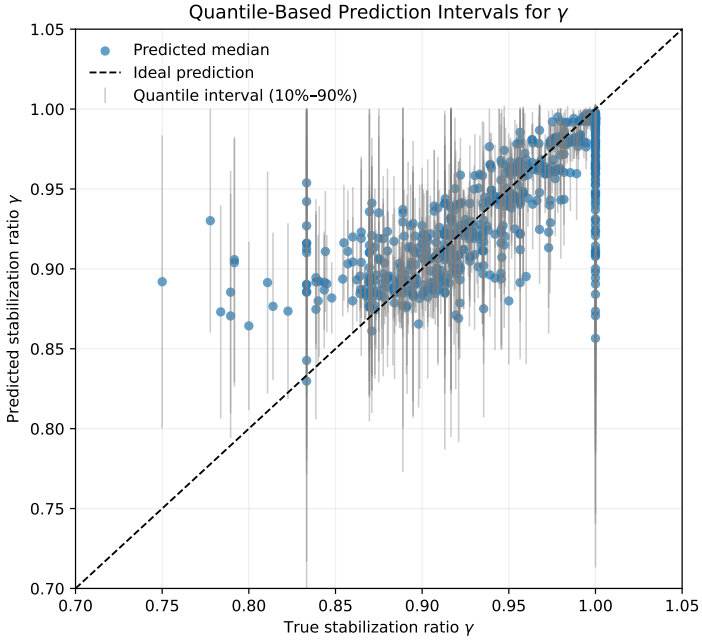


Figure 6. Quantile-based prediction intervals for the stabilization ratio γ.

Each point represents the central prediction obtained from the quantile regression model, while the vertical bars denote the empirical 10%–90% prediction interval. The dashed diagonal corresponds to the ideal prediction, where the estimated stabilization ratio matches the observed value.

The diagonal line represents the ideal prediction, where the predicted central value coincides with the true stabilization level. As observed in the figure, most instances lie close to this diagonal and are covered by the predicted intervals, indicating that the model provides well-calibrated bounds on γ. Moreover, the width of the intervals reflects the instance-dependent uncertainty: Configurations exhibiting more variability in the stabilization behavior are associated with wider intervals, while more stable instances yield tighter bounds.

These results confirm that quantile regression effectively captures the dispersion of the stabilization phenomenon observed in practice, providing informative empirical bounds rather than single-point estimates.

Finally, the estimated quantile bounds on γ are translated into bounds on the number of iterations and incorporated into the logarithmic complexity expression introduced earlier. This yields a refined

empirical characterization of the decoding cost, which accounts not only for average convergence behavior but also for the dispersion observed across decoding instances.

7.2.2. Time comparison: Prange vs. posterior-guided Prange

The Prange algorithm is used as the default ISD instance for the experimental analysis, since the posterior-guided framework developed in this work is presented and evaluated primarily in this setting. Nevertheless, the proposed approach is not intrinsically tied to Prange: The same analysis can be carried out with any ISD algorithm whose complexity can be expressed through a per-iteration cost and an expected number of iterations.

The experiments in this section consider posterior configurations parameterized by the value f, the number of error-vector positions assumed to be correctly identified by the posterior information. For each (n, k, w) instance, f was varied from $0.1k$ up to the full dimension k, yielding 10 posterior configurations per instance. For each value of f, we computed 80 independent time-complexity estimates. These estimates correspond to the cost of a decoding attempt and were obtained using the `CryptographicEstimators` framework [55]. The reported posterior-guided cost estimates are obtained by adding the correction term $\log_2(1/0.8)$ to the estimator output, where 0.8 corresponds to the mean success interval identified through quantile regression.

Figure 7 compares the classical Prange estimated time complexity with the posterior-guided Prange estimate averaged over all posterior configurations f. Each point corresponds to a fixed (n, k, w) instance. The points lie below the diagonal $y = x$, showing that posterior information reduces the estimated decoding cost. The gap also increases with the instance size, suggesting that posterior guidance has a stronger effect for larger parameters.

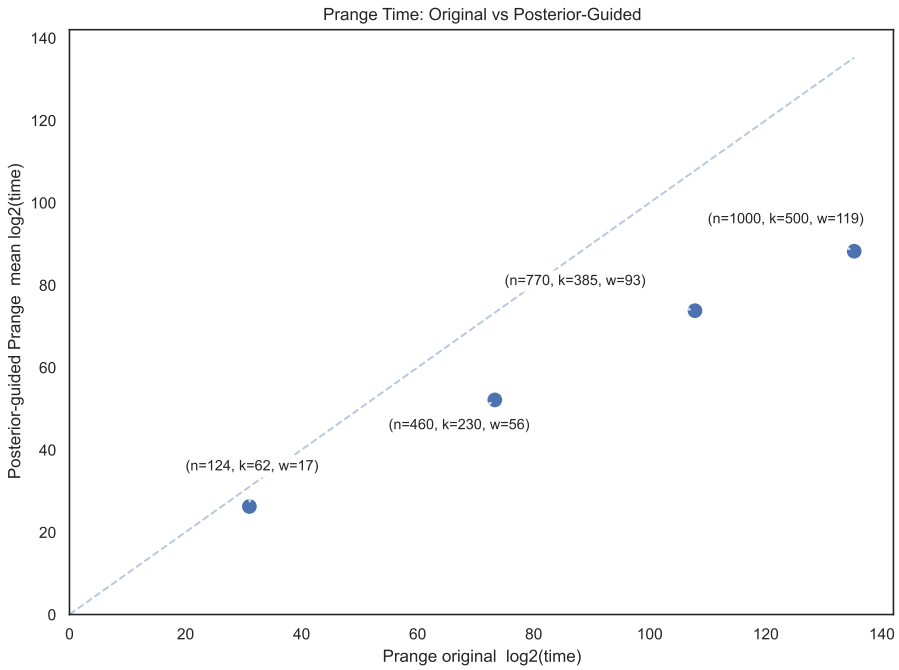


Figure 7. Comparison between classical Prange and posterior-guided Prange estimated time complexities, averaged over all posterior configurations f.

Table 3 reports time (T) and memory (M) complexity estimates in base-2 logarithmic scale for classical Prange and posterior-guided Prange. For each (n, k, w) configuration, the posterior-guided variant is evaluated over 10 values of the posterior parameter f. The table reports the average time complexity over these values, together with the corresponding minimum and maximum. These statistics summarize both the typical behavior of the posterior-guided algorithm and its variability across posterior configurations. In every tested case, the average posterior-guided complexity remains below the classical Prange estimate.

Table 3. Comparison between classical Prange and posterior-guided Prange complexity estimates.

n	k	w	$\#f$	$\log_2 T_{\text{orig}}$	$\log_2 T_{\text{mean}}$	$\log_2 T_{\min}$	$\log_2 T_{\max}$	$\log_2 M_{\text{orig}}$	$\log_2 M_{\text{mean}}$
124	62	17	10	31.01	26.23	17.15	31.21	13.24	13.10
460	230	56	10	73.30	52.11	22.13	74.84	17.05	16.87
770	385	93	10	107.80	73.77	24.15	110.46	18.59	18.42
1000	500	119	10	135.24	88.21	25.18	137.10	19.26	19.09

Posterior-guided results correspond to the mean, minimum, and maximum time complexity over all evaluated posterior configurations f. Time (T) and memory (M) complexities are reported in base-2 logarithmic scale.

Figure 8 analyzes the influence of the posterior parameter f on the posterior-guided Prange estimated time complexity. Since f represents the number of error-vector positions assumed to be correctly identified, increasing f incorporates more posterior information into the decoding process and reduces the estimated decoding cost. This monotonic decrease is observed across all tested instances, with larger parameter sets showing a steeper reduction in complexity.

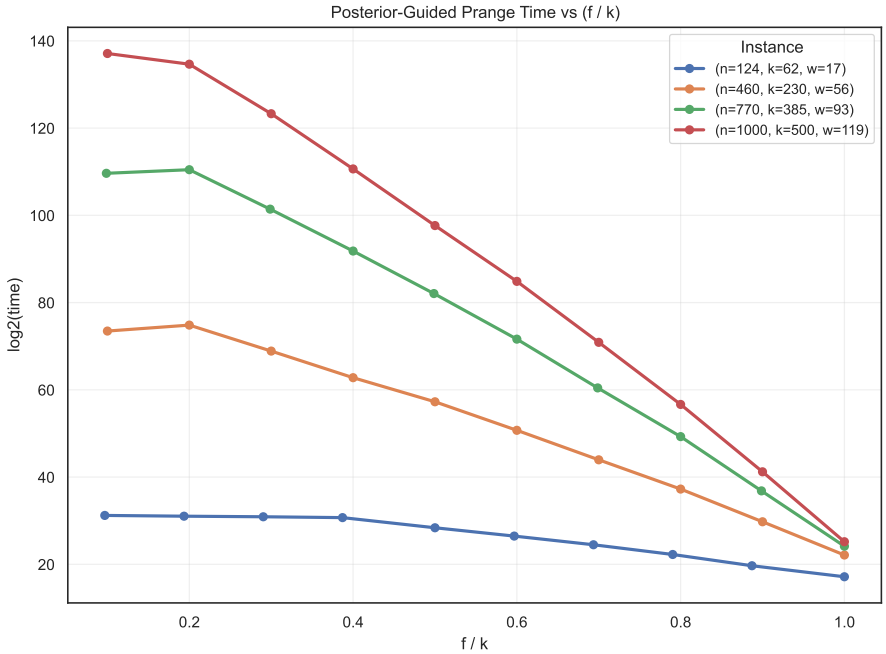


Figure 8. Posterior-guided Prange time complexity as a function of the posterior parameter f for all tested (n, k, w) instances.

Overall, the analysis shows that posterior-guided parameter reductions yield a robust and scalable decrease in the estimated Prange decoding cost. These improvements persist when averaged across posterior configurations, rather than being driven by isolated favorable cases, providing empirical support for the proposed bound estimation framework.

7.3. Comparative time estimation of guided decoding across ISD algorithms

This section compares the estimated time complexity of several ISD algorithms implemented in the `CryptographicEstimators` framework [55]. We consider BallCollision, BJMMdw, BJMMpdw, BJMM, BJMMplus, BothMay, Dumer, May-Ozerov, Prange, and Stern.

As in the previous section, the guidance parameter f is varied from $0.1k$ to k. For each value of f and each code length n, we compute 80 independent estimates to obtain stable average time-complexity values. The experiments cover $n \in \{124, 460, 770, 1000, 2000, 5000, 10000\}$, with $k = n/2$ and error weight w chosen according to the Gilbert-Varshamov bound.

For each code length, we identify the three best-performing and three worst-performing algorithms based on their mean posterior-guided time-complexity estimate across all values of f. This enables a clearer comparison of performance trends and highlights the variability in algorithmic behavior under guided decoding.

Table 4 summarizes the best-performing and worst-performing ISD algorithms under posterior-guided decoding for different code lengths n. For each value of n, the table reports the three algorithms with the lowest and highest estimated decoding times in $\log_2$ scale. The former are ranked in ascending order, and the latter in descending order, allowing direct comparison between the most and least efficient strategies under identical parameter settings.

Overall, May-Ozerov consistently appears among the three best-performing algorithms across all considered values of n, indicating strong performance under posterior-guided decoding. However, its relative advantage decreases for larger code lengths, particularly for $n = 5000$ and $n = 10000$, where BJMMplus becomes the best-performing variant. This suggests that BJMMplus scales more favorably than May-Ozerov in the largest tested regimes.

For smaller configurations, Stern tends to outperform several modern variants. This behavior, however, does not persist across all regimes: For larger values of n, Stern becomes one of the worst-performing algorithms. Classical ISD algorithms such as Prange and Dumer also perform poorly in the largest configurations, with Prange consistently showing the highest estimated decoding time across all considered code lengths.

Overall, the results suggest that May-Ozerov, BJMM, and BJMMplus are the most suitable algorithms within the posterior-guided decoding framework. By contrast, Stern, Dumer, Prange, and BallCollision appear less suitable for large-scale configurations because of their unfavorable estimated time complexity.

Table 4. Best-performing and worst-performing ISD algorithms under posterior-guided decoding, grouped by code length n. Reported times are in $\log_2$ scale.

n	Rank best algorithm	Best algorithm	$\log_2 T_{mean}$	Rank worst algorithm	Worst algorithm	$\log_2 T_{mean}$
124	1	Stern	21.18	3	Prange	25.91
	2	May-Ozerov	21.23	2	BJMMdw	22.18
	3	BothMay	21.25	1	BallCollision	22.04
460	1	May-Ozerov	41.80	3	Prange	51.79
	2	BothMay	41.92	2	BallCollision	43.05
	3	Stern	42.00	1	BJMMdw	42.86
770	1	May-Ozerov	60.31	3	Prange	73.45
	2	BJMMplus	60.71	2	BallCollision	62.16
	3	BJMM	60.78	1	BJMMdw	61.59
1000	1	May-Ozerov	72.71	3	Prange	87.89
	2	BJMMplus	73.18	2	BallCollision	75.27
	3	BJMM	73.26	1	Dumer	74.67
2000	1	May-Ozerov	133.96	3	Prange	158.08
	2	BJMMplus	134.52	2	BallCollision	140.96
	3	BJMM	134.67	1	Stern	140.72
5000	1	BJMMplus	306.10	3	Prange	347.09
	2	BJMM	306.29	2	Stern	321.76
	3	May-Ozerov	307.49	1	BallCollision	321.71
10000	1	BJMMplus	584.17	3	Prange	640.38
	2	BJMM	584.42	2	Stern	605.83
	3	May-Ozerov	586.57	1	Dumer	605.20

Figure 9 compares the estimated decoding time of the original instances with that of the posterior-guided instances. As described in the previous section, the reduced parameters (n', k', w') are obtained by averaging the reduced instances across all experimental runs, allowing comparison with the original configuration (n, k, w). The x-axis reports the estimated decoding time for the original parameters, while the y-axis reports the estimate for the posterior-guided parameters. The top row shows the best-performing algorithms (May-Ozerov, BJMM, and BJMMplus), while the bottom row shows the worst-performing algorithms (Stern, Dumer, and BallCollision). Across all considered algorithms, the data points lie below the identity line, indicating that posterior-guided decoding consistently reduces the estimated decoding cost. This behavior appears for both the best-performing and worst-performing algorithms, showing that the reduced guided instances are easier to decode than the original ones. The gap also tends to increase with code length, highlighting the effectiveness of the guided-decoding approach for larger problem sizes.

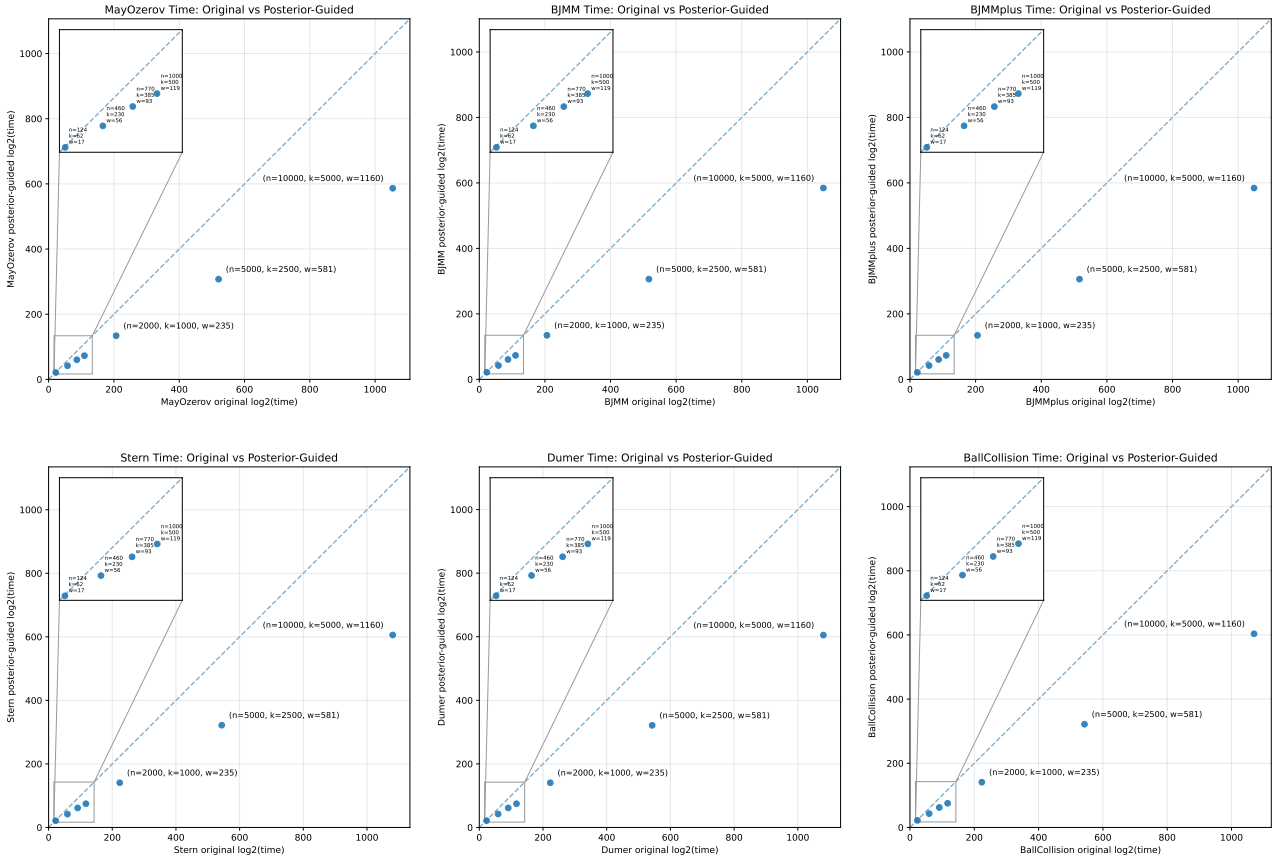


Figure 9. Comparison between original and posterior-guided time complexity estimates for different ISD algorithms.

Figure 10 shows the posterior-guided estimated decoding time as a function of the guidance ratio f/k for different ISD algorithms. Across all algorithms and parameter configurations, the estimated decoding time decreases monotonically as f/k increases. This reflects the effect of incorporating additional posterior information, which reduces the effective search space and therefore the computational effort required for decoding. The top row corresponds to the best-performing algorithms (May-Ozerov, BJMM, and BJMMplus), while the bottom row corresponds to the worst-performing algorithms (Stern, Dumer, and BallCollision). The trend appears for both the best-performing algorithms (May-Ozerov, BJMM, and BJMMplus) and the worst-performing ones (Stern, Dumer, and BallCollision), although their absolute estimated costs differ substantially. Larger code lengths show a steeper decrease as f/k grows, suggesting that posterior-guided decoding becomes increasingly beneficial for larger instances. Overall, these results show that increasing posterior guidance substantially reduces the estimated decoding cost across the considered ISD algorithms.

Overall, the experimental results show that posterior-guided decoding consistently reduces the estimated decoding cost of ISD algorithms across the considered parameter configurations. Comparisons between original and guided instances indicate that guided decoding systematically yields lower estimated decoding times, with gains becoming more pronounced as the code length increases. These estimates remain stable up to n=10000, approaching the parameter ranges

considered in practical cryptographic schemes.

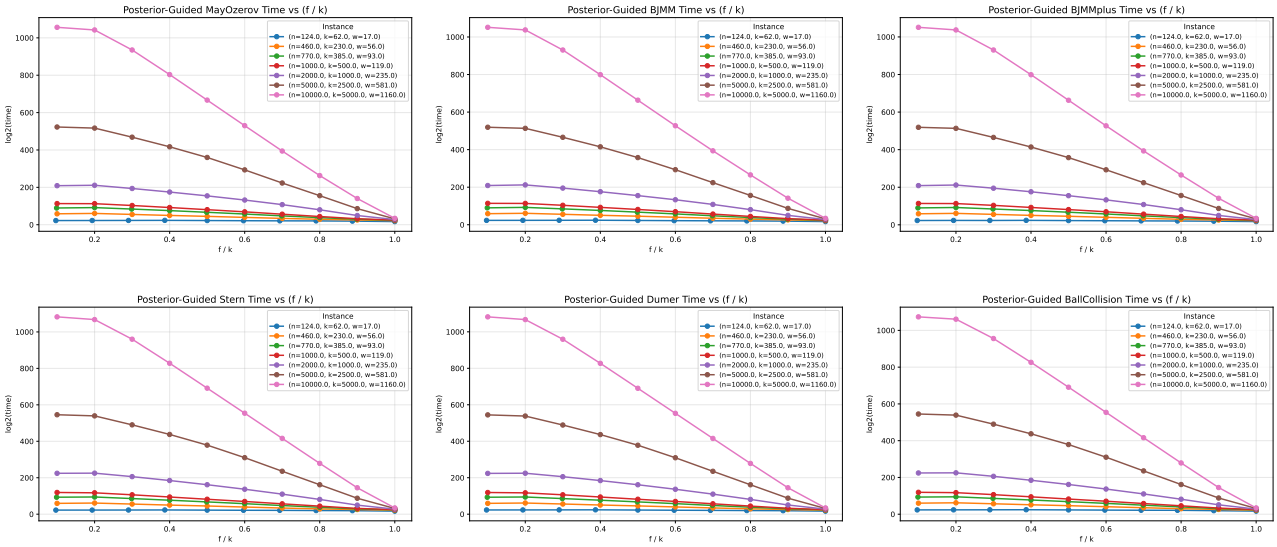


Figure 10. Posterior-guided time complexity as a function of f/k for different ISD algorithms.

The analysis as a function of the guidance ratio f/k further shows that incorporating additional posterior information leads to a monotonic reduction in estimated decoding cost across the considered algorithms. These results support the effectiveness and robustness of the proposed guided-decoding framework and show that modern ISD variants such as May-Ozerov, BJMM, and BJMMplus benefit the most from posterior guidance, particularly for large-scale instances.

7.4. Recursive decoding simulation

We conducted experiments using Algorithm 1, with the proposed genetic algorithm as the enumerator. Two code families were considered: random codes and RM codes.

RM codes were selected because of their well-established algebraic structure and relevance in code-based cryptography. They belong to the class of $(u \mid u+v)$ codes, which have been widely studied from both coding-theoretic and cryptographic perspectives [56,57]. This structure has motivated code-based digital signature schemes such as WAVE and SURF [58, 59]. More recently, modified RM codes were proposed in the *enhanced pqsigRM* signature scheme, submitted to the first round of the NIST post-quantum cryptography standardization process, further supporting their relevance in post-quantum cryptographic design [60].

The code lengths are determined by the RM parameters, since an RM(r, m) code has length $n = 2^m$. The experiments are divided into two categories: In the first, the error weight w is defined by the error-correcting capability of the corresponding RM code; in the second, w is set according to the Gilbert-Varshamov distance. The parameters used for each configuration are summarized in Table 5.

The length n and dimension k are the same for the random codes in order to preserve the same parameter set as the RM codes.

Table 5. Parameters of the RM codes used in the experiments.

m	r	n	k
5	3	32	26
6	3	64	42
7	4	128	99

7.4.1. Results for the first scenario

In the first experimental scenario, the error weights were set to $w = 1$ for $n = 32$ and to $w \in \{1, 2, 3\}$ for $n = 64$ and $n = 128$. For each code length n, thirty error vectors e were generated to simulate cold boot attacks under a binary asymmetric channel with bit-flip probabilities $\alpha = 0.01$ and $\beta = 0.2$. A genetic algorithm was then used to identify indices likely to remain unchanged after the bit-flipping stage. For each input error vector e, the genetic algorithm returns a candidate set of reliable indices, which is subsequently used within the recursive decoding algorithm.

Table 6 reports descriptive statistics of decoding times for random and RM codes across these parameter configurations. The confidence parameter τ was also varied, since it determines the size of the set of indices assumed to be correct in Algorithm 1.

Table 6. Descriptive statistics of decoding times by code length n, confidence parameter τ, and decoder type.

n	τ	Decoder	Count	Mean	Std	Min	25%	50%	75%	Max
32	0.1	Random	30	0.120	0.261	0.005	0.015	0.027	0.117	1.384
	0.1	RM	30	1.602	8.296	0.003	0.022	0.048	0.091	45.516
	0.3	Random	30	38.784	184.011	0.002	0.005	0.014	0.098	1005.121
	0.3	RM	30	50.191	228.493	0.002	0.003	0.017	0.062	1237.187
	0.5	Random	30	2.578	12.848	0.002	0.004	0.007	0.012	70.291
	0.5	RM	30	1.699	7.992	0.002	0.003	0.005	0.010	43.820
64	0.1	Random	30	37.806	35.132	0.009	11.908	29.303	58.725	134.187
	0.1	RM	30	24.587	30.107	0.009	1.826	15.964	29.479	112.373
	0.3	Random	30	15.052	23.912	0.011	0.393	2.217	20.421	73.892
	0.3	RM	30	9.994	17.101	0.009	0.034	0.839	8.823	55.044
	0.5	Random	30	0.017	0.013	0.010	0.012	0.014	0.017	0.081
	0.5	Reed–Muller	30	0.033	0.063	0.008	0.010	0.012	0.018	0.254
128	0.1	Random	30	324.079	241.646	0.037	157.087	304.210	487.0	1053.591
	0.1	RM	30	227.220	193.580	0.037	62.422	191.866	372.480	804.185
	0.3	Random	30	122.998	134.216	0.034	2.180	124.295	195.895	632.626
	0.3	RM	30	92.202	135.197	0.033	5.812	43.286	141.545	636.369
	0.5	Random	30	3.977	16.785	0.030	0.033	0.046	0.393	92.193
	0.5	RM	30	8.710	18.763	0.030	0.033	0.360	5.346	75.263

The reported statistics include the number of samples, mean, standard deviation, minimum,

maximum, and the 25^{th}, 50^{th}, and 75^{th} percentiles. The analysis reveals consistent trends across code lengths and confidence parameters. Smaller threshold values, such as $\tau = 0.1$, tend to produce longer and more variable decoding times, especially as the code length increases. By contrast, larger values of τ generally yield lower median runtimes and greater stability, because they condition on more posterior-reliable positions and therefore produce a stronger reduction of the decoding instance.

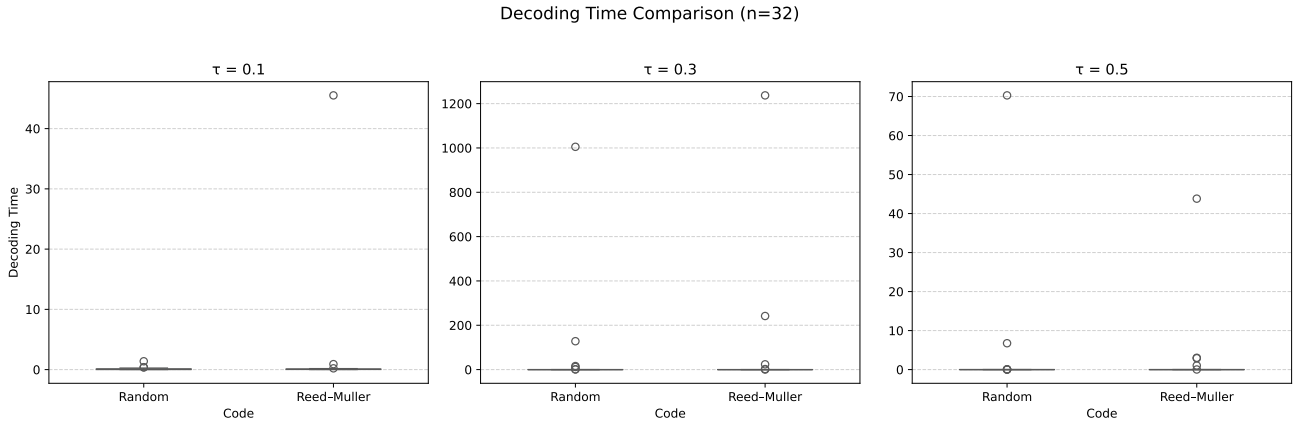
This behavior explains some initially counter-intuitive results in Table 6. Small values of τ leave a larger residual decoding problem, which increases the number of candidate configurations explored by the enumerator and leads to higher variance. Larger values of τ reduce the remaining instance more aggressively and increase the likelihood of early termination. Thus, selecting a sufficiently large τ can substantially reduce runtime, although at the cost of a higher risk of incorporating incorrect assumptions.

When comparing recursive decoding performance, RM codes achieve faster and more stable results than random codes for $n = 32$ and $n = 64$, as reflected by their lower mean decoding times and smaller standard deviations. This suggests that the structural properties of RM codes help guide the posterior-based search more effectively, allowing the algorithm to converge more efficiently and predictably.

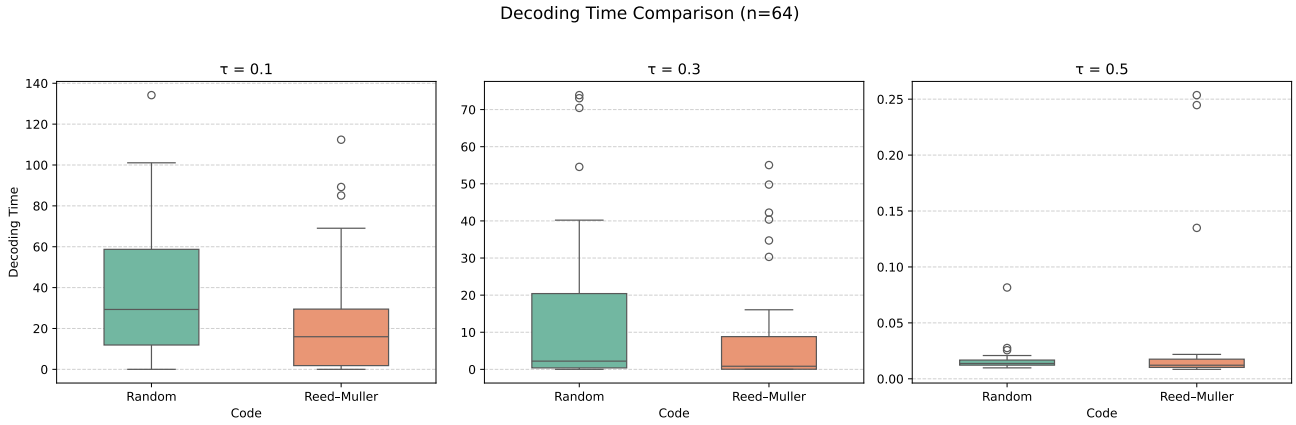
Regarding scalability, at $n = 128$, both experience a notable increase in average decoding time, while the performance gap between them diminishes. In specific cases (e.g., $\tau = 0.5$), the random-code instances even slightly surpass the RM instances, indicating that the computational complexity inherent to the RM structure becomes more pronounced at higher dimensions. The variability analysis shows that the RM instances maintain narrower interquartile ranges (25–75%) for smaller and moderate code lengths, confirming its stable runtime behavior. In contrast, the random decoder exhibits wider dispersion and several extreme outliers, particularly within the low γ region, reflecting a higher degree of instability in its decoding performance.

Overall, the RM instances provide faster and more stable decoding in most configurations up to $n = 64$. However, its scalability degrades for larger code lengths, where both decoders exhibit higher variance and longer average execution times. As shown in Figure 11, the RM decoder consistently outperforms the random decoder in execution time, demonstrating greater stability as n increases, despite the increase in execution time observed for $\tau = 0.5$ when $n = 128$. This loss of consistency can be attributed in part to the behavior of the genetic-algorithm-based enumerator. As n increases, the probability of selecting incorrect positions as correct also increases. Consequently, the algorithm may identify erroneous positions as reliable during the decoding process, which leads to longer execution times.

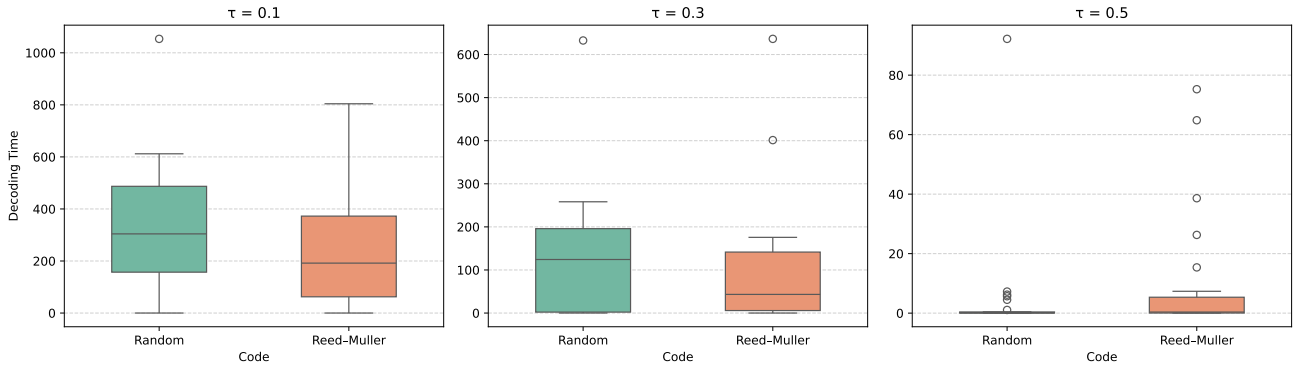
In this figure, the error weight w was defined by the maximum error-correcting capacity of the corresponding RM code. Each plot shows the distribution of execution times across the Recursive Decoding, varying the confidence parameter τ.



(a) $n = 32$



(b) $n = 64$



(c) $n = 128$

Figure 11. Boxplots of decoding execution times for the **first experimental scenario**.

7.4.2. Results for the second scenario

Table 7 summarizes the results for the second experimental scenario, where the error weights are defined according to the Gilbert-Varshamov bound. In this configuration, we use a single confidence

threshold τ, namely the value identified in the previous experiment as yielding the best decoding performance, although it increases the risk of conditioning on incorrect positions. For each code length n, sixty error vectors e were generated, with $w = 4$ for $n = 32$, $w = 7$ for $n = 64$, and $w = 8$ for $n = 128$. To keep the experiments computationally feasible, each decoding attempt was limited to 25 minutes.

Table 7. Descriptive statistics of decoding times by code length n and decoder type.

n	Decoder	Count	Mean	Std	Min	25%	50%	75%	Max
32	Random	57	26.939	103.315	0.002	0.006	1.037	5.422	702.085
	Reed–Muller	57	7.622	30.025	0.002	0.005	0.013	2.999	202.902
64	Random	21	44.128	181.689	0.014	0.015	0.020	0.050	834.645
	Reed–Muller	21	26.871	79.828	0.015	0.056	4.717	11.845	367.918
128	Random	5	18.892	39.447	0.265	0.408	1.851	2.499	89.437
	Reed–Muller	5	400.420	503.697	66.247	98.667	139.413	435.515	1262.255

Figure 12 shows the decoding accuracy obtained within the 25-minute limit for each tested code length n. The decline in accuracy as n increases reflects the increasing difficulty of completing decoding under a fixed time constraint. The count values in Table 7 report the number of syndromes successfully decoded within the imposed limit.

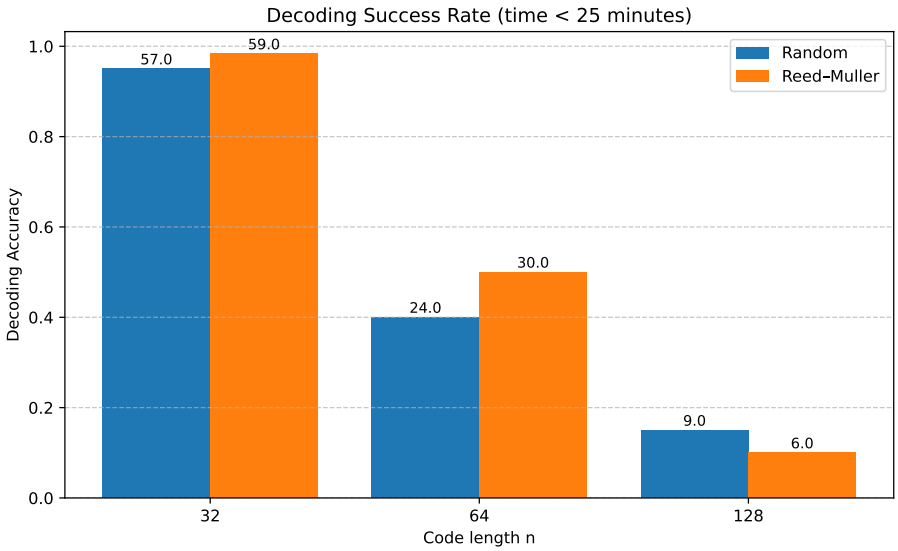


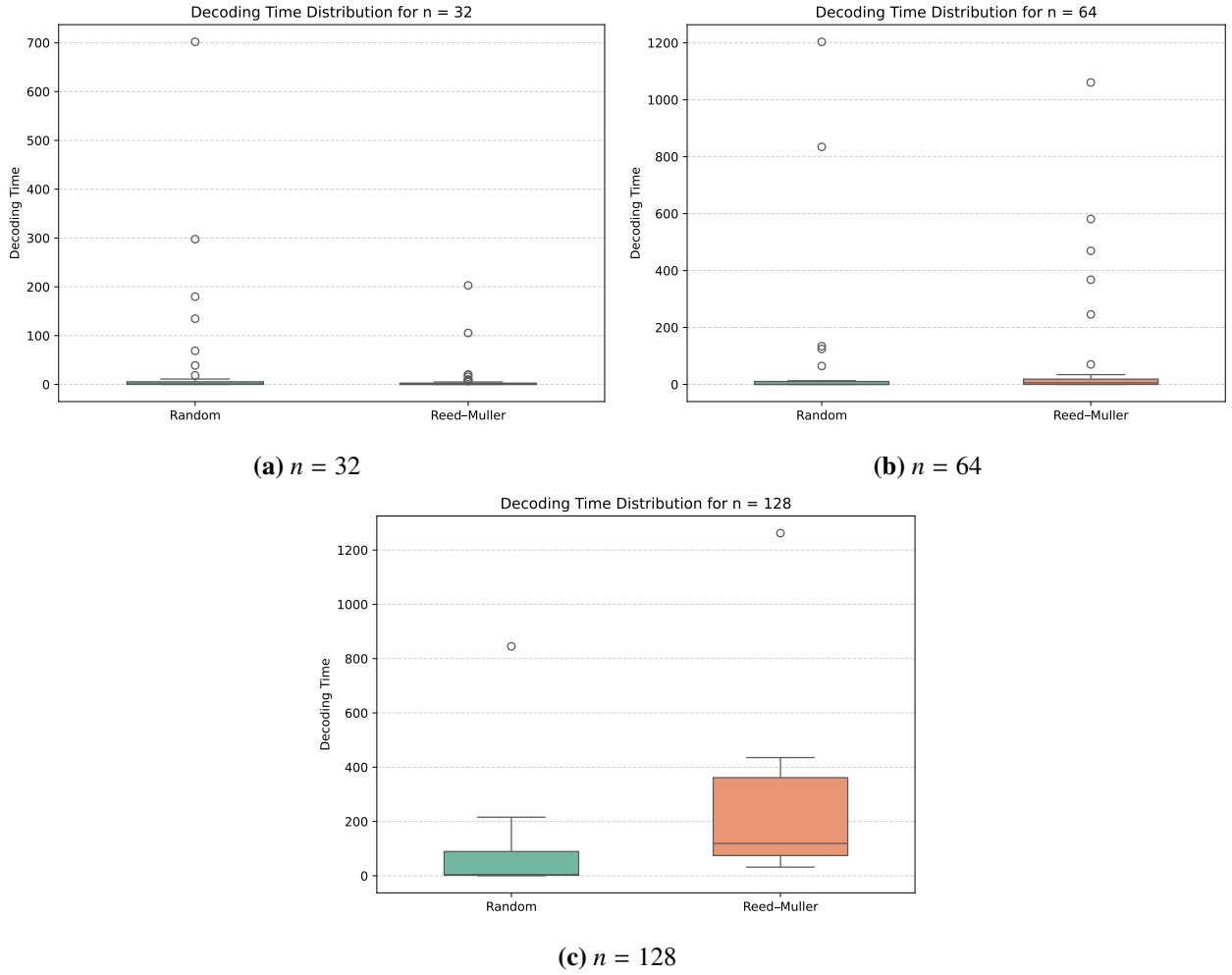
Figure 12. Decoding accuracy for each n within the established time limit.

Based on the results in Table 7, several conclusions can be drawn. For shorter code lengths (n= 32 and n = 64), RM instances show lower mean decoding times and smaller standard deviations than random-code instances. This reflects both faster execution and greater consistency across runs, suggesting that the structured RM code guides the posterior-based search more effectively.

The interquartile ranges (25%–75%) for the RM cases are also tighter, indicating reduced variability and more predictable behavior. By contrast, random-code instances show greater dispersion and occasional outliers, with maximum decoding times in some cases exceeding 700 units. These results suggest that RM codes provide more stable decoding performance for small and

medium code lengths under the given experimental conditions.

Figure 13 shows the distribution of decoding times for the second experimental scenario. As in the first scenario, RM instances tend to decode faster and with greater stability. Increasing the Hamming weight of the error vector raises decoding complexity by enlarging the search space explored by the enumerator. As a result, the genetic algorithm must search a more difficult candidate space, which leads to longer convergence times and a higher probability of selecting incorrect positions.



(a) $n = 32$ **(b)** $n = 64$

(c) $n = 128$

Figure 13. Boxplots of decoding execution times for the **second experimental scenario**, where the error weight w was defined by the Gilbert-Varshamov bound. Each plot shows the distribution of execution times across the recursive decoding, varying the confidence parameter τ.

The apparent discrepancy between RM and random-code performance across code lengths can be attributed to both structural and statistical factors. Table 6 shows that RM codes are generally faster and more stable than random codes for $n \leq 64$, suggesting that their recursive $(u \mid u + v)$ structure helps guide the posterior-based search. However, in Table 7, the number of successful decoding instances for $n = 128$ is much smaller because of the imposed time limit. Thus, the slower observed performance for RM codes at larger dimensions should not be interpreted as a general trend, but rather

as an artifact of limited sample size and increased algorithmic variance. A more conclusive scalability assessment would require additional tests with larger datasets, more decoding instances, and extended computational budgets. Such an extended evaluation is beyond the scope of the present work.

8. Conclusions and future work

This work introduced a *recursive posterior-guided decoding algorithm* that integrates conditional bitwise posterior information directly into the ISD process. By combining a MAP estimate of the error vector with a genetic-algorithm-based enumerator, the method prioritizes bit positions according to their estimated reliability, shifting ISD from a purely random sampling paradigm toward a leakage-aware guided search strategy.

The proposed framework is not tied to a particular cryptosystem or code family. Rather, it applies to any code-based construction whose security relies on the hardness of the SDP. In this sense, the contribution is formulated at the level of the underlying hard problem: Whenever leakage can be transformed into coordinate-wise posterior information, this information can be incorporated into the decoding process to guide the search. Moreover, the ISD algorithm used in the final decoding stage can be chosen according to the target parameters and available computational resources. This modularity makes the framework adaptable to different decoding strategies and different code-based cryptographic settings.

The experimental results highlight several key findings. First, incorporating posterior information improves decoding efficiency, particularly in leakage models with asymmetric bias such as the cold boot attack setting. Second, for short and moderate block lengths ($n \leq 64$), RM codes outperform random linear codes with comparable parameters. Their recursive structure appears to help the guided decoder converge faster and with greater stability. Third, as the dimension increases ($n = 128$), this performance gap narrows. In this regime, the genetic enumerator increasingly misclassifies altered positions as correct, which reduces the effectiveness of the recursive reduction and increases computational cost. Finally, the experiments show that the genetic algorithm is a feasible and computationally affordable enumerator for small and moderate instances, but its robustness decreases as the code size or the bit-flip probability β grows. This underscores the importance of improving the quality of the leakage signal and investigating alternative enumeration strategies.

Overall, the results show that posterior-guided decoding provides a viable framework for evaluating leakage-assisted attacks against code-based cryptographic constructions. The stability advantages observed for structured codes such as RM suggest interesting potential, but they also expose scaling challenges that must be addressed before targeting high-security parameter regimes. In particular, the current experiments with RM codes should be viewed as an initial step. Future work should evaluate larger code lengths and additional parameter sets in order to better understand how the recursive structure of RM codes interacts with posterior-guided reductions and whether this structure consistently improves convergence under leakage.

Several promising directions emerge for future work. From an algorithmic perspective, it is important to emphasize that the genetic algorithm used in this work is not presented as the unique or optimal enumerator. Instead, it represents a feasible heuristic choice: It is relatively simple to implement, does not require exhaustive enumeration of the subset space, and provides a computationally affordable way to explore candidate information sets. Its use also illustrates a natural

interaction between artificial intelligence techniques and cryptanalysis, since an evolutionary search method is used to guide a hard combinatorial decoding problem arising in post-quantum cryptography. This connection between cryptography and artificial intelligence opens a broader research direction in which learning-based, evolutionary, or adaptive optimization methods may be used to guide decoding under leakage.

A more systematic study of enumerators is therefore needed. Future work should compare genetic algorithms with other heuristic and probabilistic strategies, including posterior-biased sampling, Markov-chain-based methods, beam search, adaptive greedy procedures, and learning-based ranking models. Such a study would help determine which enumerators are most effective under different leakage regimes, code families, and parameter sizes. In particular, adaptive enumerators that update their search policy according to intermediate decoding outcomes may improve robustness in larger dimensions, where fixed genetic parameters may lead to early saturation or local equilibrium.

The final decoding stage also offers several directions for improvement. Although this work primarily uses Prange's algorithm for clarity and interpretability, the posterior-guided reduction is largely orthogonal to the specific ISD variant employed. Advanced ISD algorithms could therefore be incorporated after the posterior-guided reduction. Starting from Stern's algorithm [61] and extending to modern meet-in-the-middle approaches and their refinements, including the MMT framework [62] and the BJMM algorithm [41], may further reduce the overall decoding cost and extend the applicability of the framework to higher security levels. More generally, the decoder can be selected depending on the reduced instance parameters, allowing the framework to combine leakage-aware reduction with the most appropriate ISD method for each regime.

From a cryptographic standpoint, posterior-guided decoding provides a practical methodology for evaluating leakage resilience in code-based schemes. Applying this framework to structured constructions such as enhanced pqsigRM may enable a more precise quantification of key-recovery likelihood under repeated or partial leakage, contributing to more realistic side-channel and cold boot attack models. At the same time, the consistently shorter decoding times observed for RM codes raise important questions about possible structural effects under leakage. Future work should therefore study larger RM instances and analyze whether their recursive $(u \mid u + v)$ structure systematically facilitates posterior-guided reductions or whether the observed behavior is specific to the tested parameter ranges.

The framework can also be extended beyond RM codes. Since the approach only requires an SDP instance and coordinate-wise posterior information, it can be applied to any cryptographic construction whose security is underpinned by syndrome decoding. This includes classical families such as binary Goppa codes and structured families such as LDPC and QC-MDPC codes. In these settings, the posterior-guided strategy does not replace the underlying decoder; rather, it biases the search using leakage-derived posterior information. Depending on the algebraic structure of the code family, future work may also combine the reduced instances produced by posterior guidance with algebraic decoding methods. This direction is especially relevant for structured codes, where the reduced problem may preserve exploitable algebraic properties that are not used by generic ISD algorithms.

Such comparative studies would help determine whether the observed gains are specific to particular code structures or reflect more general trade-offs between redundancy, decoding efficiency, and leakage resilience. Ultimately, the proposed framework provides a bridge between physical

leakage models, combinatorial decoding, and heuristic artificial intelligence methods, offering a flexible basis for studying the practical security of code-based cryptography in leakage-prone environments.

Author contributions

Andrés Florián-Quitián: primarily responsible for the development and writing of the Results section, proposed the comparison of the proposed approach with RM codes and random codes; Valérie Gauthier-Umaña: proposed the comparison of the proposed approach with Reed–Muller codes and random codes, Estefanía Laverde-Becerra: developed the Genetic Algorithm used in the study, primarily responsible for the development and writing of the Results section; Ricardo Villanueva-Polanco: conceived the general idea of the study, contributed to its overall conceptual structure, led the enumerator analysis, and was primarily responsible for structuring the manuscript and guiding the revision process. All authors have read and agreed to the published version of the manuscript.

Use of Generative-AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Conflict of interest

The authors declare no conflicts of interest.

References

1. J. Justesen, T. Høholdt, *A course in error-correcting codes*, European Mathematical Society Zürich, 2004.
2. I. S. Reed, G. Solomon, Polynomial codes over certain finite fields, *J. Soc. Ind. Appl. Math.*, **8** (1960), 300–304, https://doi.org/10.1137/0108018
3. D. E. Muller, Application of boolean algebra to switching circuit design and to error detection, *Trans. IRE Prof. Group Electron. Comput.*, **EC-3** (1954), 6–12. https://doi.org/10.1109/IREPGELC.1954.6499441
4. V. D. Goppa, A new class of linear correcting codes, *Probl. Peredachi Inf.*, **6** (1970), 24–30.
5. R. J. McEliece, A public-key cryptosystem based on algebraic, *Coding Theory*, **4244** (1978), 114–116.
6. M. Baldi, F. Chiaraluce, Cryptanalysis of a new instance of mceliece cryptosystem based on QC-LDPC codes, *2007 IEEE International Symposium on Information Theory*, 2007. https://doi.org/10.1109/ISIT.2007.4557609
7. V. M. Sidelnikov, A public-key cryptosystem based on binary reed-muller codes, *Discrete Math. Appl.*, **4** (1994), 191–208, https://doi.org/10.1515/dma.1994.4.3.191

8. E. M. Gabidulin, A. V. Paramonov, O. V. Tretjakov, Ideals over a non-commutative ring and their application in cryptology, In: D. W. Davies, *Advances in cryptology — EUROCRYPT '91*, Springer Berlin Heidelberg, 1991, 482–489. https://doi.org/10.1007/3-540-46416-6_41

9. National Institute of Standards and Technology, *Pqc standardization process*, 2017. Avaible from: `https://www.nist.gov/pqcrypto`.

10. C. Aguilar-Melchor, J. C. Deneuville, A. Dion, Hamming quasi-cyclic (hqc), *Technical report*, 2025.

11. National Institute of Standards and Technology, *Pqc standardization process: announcing four candidates to be standardized, plus fourth round candidates*, 2022. Avaible from: `https://csrc.nist.gov/news/2022/pqc-candidates-to-be-standardized-and-round-4`.

12. F. X. Standaert, Introduction to side-channel attacks, In: I. M. R. Verbauwhede, *Secure integrated circuits and systems*, Springer, 2009, 27–42. https://doi.org/10.1007/978-0-387-71829-3_2

13. R. V. Polanco, *Cold boot attacks on post-quantum schemes*, Ph.D. Thesis, Royal Holloway, University of London, 2019.

14. J. A. Halderman, S. D. Schoen, N. Heninger, W. Clarkson, W. Paul, J. A. Calandrino, et al., Lest we remember: cold-boot attacks on encryption keys, *Commun. ACM*, **52** (2009), 91–98. https://doi.org/10.1145/1506409.1506429

15. N. Heninger, H. Shacham, Reconstructing RSA private keys from random key bits, In: S. Halevi, *Advances in cryptology - CRYPTO 2009*, Springer Berlin Heidelberg, 2009. https://doi.org/10.1007/978-3-642-03356-8_1

16. W. Henecka, A. May, A. Meurer, Correcting errors in RSA private keys, In: T. Rabin, *Advances in cryptology – CRYPTO 2010*, Springer Berlin Heidelberg, 2010, 351–369. https://doi.org/10.1007/978-3-642-14623-7_19

17. K. G. Paterson, A. Polychroniadou, D. L. Sibborn, A coding-theoretic approach to recovering noisy RSA keys, In: *Advances in cryptology – ASIACRYPT 2012*, Springer Berlin Heidelberg, 2012, 386–403.

18. H. T. Lee, H. Kim, Y. J. Baek, J. H. Cheon, Correcting errors in private keys obtained from cold boot attacks, In: H. Kim, *Information security and cryptology - ICISC 2011*, Springer Berlin Heidelberg, 2012, 74–87. https://doi.org/10.1007/978-3-642-31912-9_6

19. B. Poettering, D. L. Sibborn, Cold boot attacks in the discrete logarithm setting, In: K. Nyberg, *Topics in cryptology — CT-RSA 2015*, Springer International Publishing, 2015, 449–465. https://doi.org/10.1007/978-3-319-16715-2_24

20. M. Albrecht, C. Cid, Cold boot key recovery by solving polynomial systems with noise, In: J. Lopez, G. Tsudik, *Applied cryptography and network security*, Springer Berlin Heidelberg, 2011, 57–72. https://doi.org/10.1007/978-3-642-21554-4_4

21. A. A. Kamal, A. M. Youssef, Applications of SAT solvers to AES key recovery from decayed key schedule images, *2010 Fourth International Conference on Emerging Security Information, Systems and Technologies*, 2010, 216–220. https://doi.org/10.1109/SECURWARE.2010.42

22. K. G. Paterson, R. Villanueva-Polanco, Cold boot attacks on NTRU, In: A. Patra, N. P. Smart, *Progress in cryptology – INDOCRYPT 2017*, Springer International Publishing, 2017, 107–125. https://doi.org/10.1007/978-3-319-71667-1_6

23. R. Villanueva-Polanco, Cold boot attacks on bliss, In: P. Schwabe, N. Thériault, *Progress in cryptology – LATINCRYPT 2019*, Springer International Publishing, 2019, 40–61. https://doi.org/10.1007/978-3-030-30530-7_3

24. R. Villanueva-Polanco, E. Angulo-Madrid, Cold boot attacks on the supersingular isogeny key encapsulation (SIKE) mechanism, *Appl. Sci.*, **11** (2021), 193. https://doi.org/10.3390/app11010193

25. M. R. Albrecht, A. Deo, K. G. Paterson, Cold boot attacks on ring and module LWE keys under the NTT, *IACR Trans. Cryptographic Hardware Embedded Syst.*, **2018** (2018), 173–213, https://doi.org/10.13154/tches.v2018.i3.173-213

26. G. Banegas, R. Villanueva-Polanco, On recovering block cipher secret keys in the cold boot attack setting, *Cryptography Commun.*, **17** (2025), 311–335. https://doi.org/10.1007/s12095-022-00625-z

27. G. Teseleanu, Partial exposure attacks against a family of rsa-like cryptosystems, *Cryptography*, **9** (2025), 2. https://doi.org/10.3390/cryptography9010002

28. A. Esser, A. May, J. Verbel, W. Wen, Partial key exposure attacks on bike, rainbow and NTRU, In: Y. Dodis, T. Shrimpton, *Advances in cryptology – CRYPTO 2022*, Springer Nature Switzerland, 2022, 346–375. https://doi.org/10.1007/978-3-031-15982-4_12

29. E. Kirshanova, A. May, Decoding mceliece with a hint – secret Goppa key parts reveal everything, In: C. Galdi, S. Jarecki, *Security and cryptography for networks*, Springer International Publishing, 2022, 3–20. https://doi.org/10.1007/978-3-031-14791-3_1

30. G. D'Alconzo, A. Esser, A. Gangemi, C. Sanna, Sneaking up the ranks: partial key exposure attacks on rank-based schemes, *Designs Codes Cryptography*, **94** (2024), 15. https://doi.org/10.1007/s10623-025-01738-1

31. Y. Seto, H. Furue, A. Takayasu, Partial key exposure attacks on UOV and its variants, *Cryptology ePrint Arch.*, 2025.

32. V. Gauthier-Umaña, A. M. Ochoa-Toro, R. Villanueva-Polanco, Seed recovery from probabilistic leakage via resource-aware, posterior-guided enumeration, *IEEE Access*, **14** (2026), 17264–17277. https://doi.org/10.1109/ACCESS.2026.3659061

33. E. Prange, The use of information sets in decoding cyclic codes, *IRE Trans. Inf. Theory*, **8** (1962), 5–9. https://doi.org/10.1109/TIT.1962.1057777

34. I. Reed, A class of multiple-error-correcting codes and the decoding scheme, *Trans. IRE Prof. Group Inf. Theory*, **4** (1954), 38–49. https://doi.org/10.1109/TIT.1954.1057465

35. E. Abbe, A. Shpilka, M. Ye, Reed–Muller codes: theory and algorithms, *IEEE Trans. Inf. Theory*, **67** (2020), 3251–3277. https://doi.org/10.1109/TIT.2020.3004749

36. M. Plotkin, Binary codes with specified minimum distance, *IRE Trans. Inf. Theory*, **6** (1960), 445–450. https://doi.org/10.1109/TIT.1960.1057584

37. J. A. Halderman, S. D. Schoen, N. Heninger, W. Clarkson, W. Paul, J. A. Calandrino, et al., Lest we remember: cold-boot attacks on encryption keys, *Commun. ACM*, **52** (2009), 91–98. https://doi.org/10.1145/1506409.150642
38. S. Lindenlauf, H. Höfken, M. Schuba, Cold boot attacks on DDR2 and DDR3 SDRAM, *2015 10th International Conference on Availability, Reliability and Security*, 2015, 287–292. https://doi.org/10.1109/ARES.2015.28
39. Y. S. Won, J. Y. Park, D. G. Han, S. Bhasin, Practical cold boot attack on IoT device - case study on Raspberry Pi -, *2020 IEEE International Symposium on the Physical and Failure Analysis of Integrated Circuits (IPFA)*, 2020. https://doi.org/10.1109/IPFA49335.2020.9260613
40. J. Stern, A method for finding codewords of small weight, *Proceedings of the 3rd International Colloquium on Coding Theory and Applications*, 1988, 106–113. https://doi.org/10.5555/646721.702702
41. A. Becker, A. Joux, A. May, A. Meurer, Decoding random binary linear codes in $2^{n/20}$: How 1+ 1= 0 improves information set decoding, In: D. Pointcheval, T. Johansson, *Advances in cryptology – EUROCRYPT 2012*, Springer, 2012, 520–536. https://doi.org/10.1007/978-3-642-29011-4_31
42. A. May, I. Ozerov, On computing nearest neighbors with applications to decoding of binary linear codes, In: E. Oswald, M. Fischlin, *Advances in cryptology – EUROCRYPT 2015*, Springer, 2015, 203–228. https://doi.org/10.1007/978-3-662-46800-5_9
43. D. Dachman-Soled, H. Gong, M. Kulkarni, A. Shahverdi, Partial key exposure in ring-lwe-based cryptosystems: attacks and resilience, *Cryptology ePrint Arch.*, 2018.
44. A. Esser, A. May, J. Verbel, W. Wen, Partial key exposure attacks on bike, rainbow and NTRU, *Ann. Int. Cryptology Conference*, Springer, 2022, 346–375.
45. Q. Guo, D. Nabokov, E. Suvanto, T. Johansson, Key recovery attacks on approximate homomorphic encryption with non-worst-case noise flooding countermeasures, *33rd USENIX Security Symposium (USENIX Security 24)*, 2024, 7447–7461.
46. A. L. Horlemann, S. Puchinger, J. Renner, T. Schamberger, A. Wachter-Zeh, Information-set decoding with hints, In: A. Wachter-Zeh, H. Bartz, G. Liva, *Code-based cryptography*, Springer International Publishing, 2022, 60–83. https://doi.org/10.1007/978-3-030-98365-9_4
47. Q. Guo, A. Johansson, T. Johansson, A key-recovery side-channel attack on classic McEliece implementations, *IACR Trans. Cryptographic Hardware Embedded Syst.*, **2022** (2022), 800–827. https://doi.org/10.46586/tches.v2022.i4.800-827
48. D. Dachman-Soled, L. Ducas, H. Gong, M. Rossi, Lwe with side information: Attacks and concrete security estimation, In: D. Micciancio, T. Ristenpart, *Advances in cryptology – CRYPTO 2020*, Springer, 2020, 329–358. https://doi.org/10.1007/978-3-030-56880-1_12
49. L. D'Achille, A. Esser, N. Kraus, Syndrome decoding with hints, *Cryptology ePrint Arch.*, 2026.
50. P. Simmons, Security through amnesia: a software-based solution to the cold boot attack on disk encryption, *Proceedings of the 27th Annual Computer Security Applications Conference*, 2011, 73–82. https://doi.org/10.1145/2076732.207674
51. M. Neagu, S. Manich, Defending cache memory against cold-boot attacks boosted by power or em radiation analysis, *Microelectron. J.*, **62** (2017), 85–98. https://doi.org/10.1016/j.mejo.2017.02.010

52. T. Farheen, S. Roy, A. Cannon, J. Di, S. Tajik, D. Forte, Amnesiac memory: a self-destructive polymorphic mechanism against cold boot data remanence attack, *Proceedings of the Great Lakes Symposium on VLSI 2024*, 2024, 564–568. https://doi.org/10.1145/3649476.3658778
53. P. Jedlicka, L. Malina, P. Socha, T. Gerlich, Z. Martinasek, J. Hajny, On secure and side-channel resistant hardware implementations of post-quantum cryptography, *Proceedings of the 17th International Conference on Availability, Reliability and Security*, 2022, 144. https://doi.org/10.1145/3538969.3544423
54. A. Azouaoui, I. Chana, M. Belkasmi, Efficient information set decoding based on genetic algorithms, *Int. J. Commun. Network Syst. Sci.*, **5** (2012), 423–429. https://doi.org/10.4236/ijcns.2012.57052
55. A. Esser, J. Verbel, F. Zweydinger, E. Bellini, Sok: cryptographicestimators – a software library for cryptographic hardness estimation, *Proceedings of the 19th ACM Asia Conference on Computer and Communications Security*, 2024, 560–574. https://doi.org/10.1145/3634737.3645007
56. I. Márquez-Corbella, J. P. Tillich, Using reed-solomon codes in the $(u|u+v)$ construction and an application to cryptography, *2016 IEEE International Symposium on Information Theory (ISIT)*, 2016, 930–934. https://doi.org/10.1109/ISIT.2016.7541435
57. I. Márquez-Corbella, J. P. Tillich, Attaining capacity with iterated $(u|u+v)$ codes based on AG codes and Koetter–Vardy soft decoding, *2017 IEEE International Symposium on Information Theory (ISIT)*, 2017, 6–10. https://doi.org/10.1109/ISIT.2017.8006479
58. T. Debris-Alazard, N. Sendrier, J. P. Tillich, Wave: a new code-based signature scheme, *IACR Cryptology ePrint Arch.*, 2018.
59. T. Debris-Alazard, N. Sendrier, J. P. Tillich, Surf: a new code-based signature scheme, *IACR Cryptology ePrint Arch.*, 2022.
60. J. Cho, J. S. No, Y. Lee, Z. Koo, Y. S. Kim, Enhanced pqsigrm: code-based digital signature scheme with short signature and fast verification for post-quantum cryptography, *Cryptology ePrint Arch.*, 2022.
61. J. Stern, A method for finding codewords of small weight, *International Colloquium on Coding Theory and Applications*, Springer, 1988, 106–113.
62. A. May, A. Meurer, E. Thomae, Decoding random linear codes in $\tilde{O}(2^{0.054n})$, In: D. H. Lee, X. Wang, *Advances in cryptology – ASIACRYPT 2011*, Springer, 2011, 107–124. https://doi.org/10.1007/978-3-642-25385-0_6

AIMS Press