Efficient QC-MDPC Cryptosystems with Bounded Decoding Failure Rate (Extended Version)

Alessandro Annechini[0009−0009−4452−802X], Alessandro Barenghi[0000−0003−0840−6358], Gerardo Pelosi[0000−0002−3812−5429], and Simone Perriello[0000−0001−9656−7252]

Department of Electronics, Information and Bioengineering - DEIB, Politecnico di Milano, Milano, Italy
{alessandro.annechini, alessandro.barenghi, gerardo.pelosi, simone.perriello}@polimi.it

Abstract. Niederreiter-style post quantum cryptosystems based on QC-MDPC codes, such as BIKE, have shown promising efficiency figures and enjoy a straightforward reduction to conjectured-hard problems in coding theory. The longstanding issue in their design is having a closed form Decoding Failure Rate (DFR) analysis of the iterative decoder employed by their decryption primitive, as decoding failures leak information on the private key. State of the art models either provide loose bounds, or do not consider the decoding algorithm employed in practice, using the behavior of a simpler one as a proxy. In this work, we provide a closed-form estimate of the DFR for the practically employed three-iterations parallel decoder, applied to QC-MDPC codes. This result constitutes the first closed form DFR model targeting both the same code family and the same decoder employed in the cryptosystem. Leveraging our estimation technique, we design the parameters for a QC-MDPC based Niederreiter encryption scheme, obtaining a 2× improvement in public key and ciphertext size w.r.t. the previous best cryptosystem design with DFR closed-form bounds, LEDAcrypt-KEM. Furthermore, we show that our new parameters yield up to 30% smaller public key size and 2.2× to 4.4× smaller ciphertexts w.r.t. HQC, the code based key encapsulation method selected by the US NIST for standardization, and achieve up to 3× speedup with respect to BIKE in ephemeral and long-term key usage.

Keywords: Post-quantum cryptosystems · QC-MDPC · Decoding failure rate

1 Introduction

The design of post-quantum cryptosystems has seen a significant amount of interest in recent years, also in the wake of the public international contest organized by the US National Institute of Standards and Technology (NIST). Among the most promising computationally hard problems, the one of decoding the syndrome of a fixed weight error vector in a random code [16] was employed to obtain

asymmetric encryption schemes, according to a template proposed by Niederreiter [47]. The Niederreiter scheme is based directly on both the hardness of the random code Syndrome Decoding Problem (SDP), and the indistinguishability of an obfuscated generator or parity-check matrix of an efficiently decodable code from the one of a random code. Currently, two efficiently decodable code families survived cryptanalytic attacks, giving rise to NIST contest participants: the binary Goppa codes selected by Classic McEliece [2], and the sparse Quasi Cyclic (QC) parity-check matrix codes selected by LEDAcrypt-KEM [13] and BIKE [7], which employ the Low/Medium Density Parity Check codes (QC-LDPC/QC-MDPC), providing compact public key sizes.

The crucial shortcoming in employing QC-MDPCs is the fact that the decoding strategies employed with them are a family of fixed-point algorithms, known as *iterative bit-flipping decoders*, that exhibit a non-null DFR. Having a sound estimate of the DFR is a necessary condition for security, as it is known that observing even a single decoding failure allows an attacker to obtain the private key of the Niederreiter PKE [59]. As a consequence, the cryptosystem parameters, decoding algorithm and its parameters (thresholds) choices, should ensure that the DFR is $\leq 2^{-\lambda}$, where λ is the security parameter (e.g., $\lambda = 128$), making the occurrence of a single failure as hard as a combinatorial attack against the cryptosystem. This requirement, crucial in scenarios where IND-CCA2 guarantees are desired [36], does not allow Monte Carlo simulations to estimate the DFR.

Current Decoding Failure Rate Models. To tackle the DFR estimation issue, LEDAcrypt-KEM [13] employs a closed-form loose upper bound on the DFR of a two-iteration parallel bit flipping decoder [11], and exhibits public key and ciphertext sizes roughly $2\times$ larger than the alternative design by BIKE [7]. A recent work [4] improved the two-iterations DFR bound, albeit being still limited to the use of a two-iteration decoder. Moreover, such analysis does not take into account the quasi-cyclic structure of the parity check matrix, which can negatively affect the decoding performance [58].

A different approach was employed by the BIKE design team [7], who chose to simulate the behavior of their chosen decoder in a high ($\approx 2^{-30}$) DFR regime, and extrapolate the behavior around the alleged parameter set to achieve $2^{-\lambda}$. Unfortunately, the trend of the DFR when increasing the code length, and keeping all other parameters still, is known to have an exponential (in the code length) decrease region known as *waterfall*, followed by a region where the decrease becomes polynomial known as *(error) floor*. Extrapolating results from the *waterfall* regime may provide optimistic estimates if the extrapolated point lies in the floor region. BIKE parameters were shown to provide a non satisfactory DFR that was at least $2^{11.39}$ larger than required [59].

A significant research effort was made to estimate the beginning and shape of the floor region for BIKE. Vasseur [58] observed that a specific set of error vectors (*near codewords*) account for most of the failures in the floor regime of specific decoders. The state of the art result (CRYPTO 2025 [8]) estimates the DFR of an infinite-iterations sequential bit flipping decoder as a proxy for the seven-iterations parallel bit flipping one employed in BIKE. The proxy decoder

is conjectured to have higher (worse) DFR than the one in BIKE, based on simulations in the 2^{-20}–2^{-30} DFR regime [58]. The proxy decoder performance estimates are thus employed as an upper bound to the DFR performance of the actual BIKE decoder to design cryptosystem parameters. This was not deemed sufficiently convincing to select BIKE for standardization [1], and in [53] it was stated that the result needs more vetting by the community, remarking that "no closed-form DFR analysis for more than 2 rounds of decoding [exists]".

1.1 Contributions

In this work, we provide a DFR model for the practically employed three-iteration parallel decoder applied to QC-MDPC codes, followed by a secure and efficient re-design of Niederreiter-style cryptosystems.

Error Structures in QC-MDPC Codes. We show that the error vectors dominating the floor region of a bit flipping decoder depend on the decoder nature and parameters, making the use of DFR estimates on proxy decoders potentially dangerous. To substantiate this claim, we generalize the concept of near codewords from [58], defining a new set of error vectors, which we call *half codewords*. We show that these error vectors, which are correctable by a sequential decoder (such as the proxy one in [8], expected to perform worse than a parallel decoder) can affect the DFR of parallel bit flipping decoders more significantly than near codewords under common decoding parameters.

Decoding Failure Rate Model. We derive the first closed form DFR model targeting both the exact *codes* (QC-MDPC) and *decoder* (parallel three-iterations decoder) practically employed to build cryptosystems. To do so, we first analyse the decoding failures induced by generic *hard-to-decode* errors extending the analysis from [4]. Then, we quantify the effect of *near codewords* and *half codewords* on the three-iteration decoder, successfully modeling both the waterfall and error floor regimes of QC-MDPC codes. The utility of the resulting model is twofold: it provides strong security guarantees for the usage of long-term keys, while employing the efficient three-iterations decoder, and it constitutes a versatile tool for the selection of optimal ephemeral keys parameters that minimize the public key and ciphertext size, while maintaining a DFR sufficiently low from an engineering perspective (e.g., 2^{-64}). The third iteration of the bit flipping decoder allows for a reduction in public key and ciphertext size w.r.t. LEDAcrypt-KEM, while requiring less than half the iterations employed by the BIKE-flip decoder.

Efficient QC-MDPC Cryptosystems Design. We leverage our model to re-design the parameters of Niederreiter-style post-quantum cryptosystems, with strong DFR bounding guarantees. To do so, we perform an exhaustive search for code parameters that provide security against passive and active attackers. Our design obtains ciphertexts smaller than BIKE and LEDAcrypt-KEM, and achieve a 3× speedup with respect to BIKE for parameters with the same public key and ciphertext size. Our parameter design also allows to obtain up to 30% smaller public key size and 2.2× to 4.4× smaller ciphertexts w.r.t. HQC [44], the code based KEM selected by the NIST for standardization.

2 Background

In this section, we summarize the relevant notions and definitions from coding theory, constructions of code-based post-quantum cryptosystems, and the structure of a parallel iterative bit flipping decoder.

Notation. In the following, $\mathbf{v} \in \mathbb{F}_2^n$, $n \geq 1$, is a column vector, while its transposed $\mathbf{v}^\intercal$ is row vector. $\mathrm{Supp}(\mathbf{v})$ denotes the set of positions in $\{0, \ldots, n-1\}$ of the non null coordinates of $\mathbf{v}$. $\mathtt{wt}(\mathbf{v})$ denotes the Hamming weight of $\mathbf{v}$, i.e., the count of its non null components, while $\mathbf{0}$ is the null vector. A matrix $\mathbf{M} \in \mathbb{F}_2^{n_1 \times n_2}$ has n_1 rows and n_2 columns, while $\mathbf{M}_{:,j}$ and $\mathbf{M}_{\ell,:}$ denote the j-th column and the ℓ-th row of $\mathbf{M}$, with $0 \leq j \leq n_2-1$, $0 \leq \ell \leq n_1-1$, respectively. Given a binary polynomial $\mathtt{a}(x) \in \mathbb{F}_2[x]/\langle x^p - 1\rangle$, we denote by $\mathbf{A} = \mathrm{circ}(\mathtt{a}(x))$ the associated $p \times p$ circulant matrix. $\mathbf{A}$ is thus a square matrix with the first row defined by the coefficients of $\mathtt{a}(x)$, ordered from least to most significant. Each subsequent row is a one-element circular right shift of the previous one. We denote as $\textsc{Bin}(\mathtt{tr}, \mathtt{pr}, \mathtt{ns})$ the probability mass function of obtaining $\mathtt{ns}$ successes out of $\mathtt{tr}$ independent trials each one having success probability $\mathtt{pr}$, and as $\mathcal{X} \sim \textsc{Bin}(\mathtt{tr}, \mathtt{pr})$ a r.v. $\mathcal{X}$ following such distribution. We denote Fisher's noncentral hypergeometric distribution as $\textsc{Fnchypg}(q_x; q_{tot}, N_x, N_{tot}, p_x, p_{tot})$, modeling the probability distribution of successful draws q_x within a subset of size N_x, given a fixed total number of successes q_{tot} and a total population size N_{tot}, where the success probability for each sample is p_x for element in the subset and p_{tot} for elements outside the subset.

2.1 Preliminaries on Coding Theory

A binary linear code is a subspace of $\mathbb{F}_2^n$ obtained as the image of a bijective linear map $\mathcal{C}[n,k] : \mathbb{F}_2^k \to \mathbb{F}_2^n$, $n, k \in \mathbb{N}^+$, with $k \leq n$, between any binary k-tuple, known as *information word*, and a binary n-tuple, known as *codeword*. The values n and k are known as the *length* and the *dimension* of the code, respectively, and their ratio $\frac{k}{n}$ defines the code *rate*. Encoding in $\mathcal{C}[n,k]$ means mapping an information word $\mathbf{u} \in \mathbb{F}_2^k$ into a codeword $\mathbf{c} \in \mathcal{C}[n,k] \subset \mathbb{F}_2^n$. Given a codeword $\widetilde{\mathbf{c}}$ corrupted by an unknown error vector $\mathbf{e} \in \mathbb{F}_2^n$, i.e.: $\widetilde{\mathbf{c}} = \mathbf{c} \oplus \mathbf{e}$, with $\mathtt{wt}(\mathbf{e}) \leq t$, recovering the original information word $\mathbf{u}$ and the error vector $\mathbf{e}$ requires the application of a *error correcting decoding procedure*.

In this work, we consider only the Hamming metric, which defines the distance between any $\mathbf{u}, \mathbf{v} \in \mathbb{F}_2^n$ as the number of their components with distinct when paired by position, i.e., $\mathtt{dist}(\mathbf{u}, \mathbf{v}) = \mathtt{wt}(\mathbf{u} \oplus \mathbf{v})$, where $\oplus$ is the bitwise `xor` between the vectors. The (minimum) distance of a code d is defined as the minimum Hamming distance between two codewords, i.e., $d = \min_{\mathbf{c}_1, \mathbf{c}_2 \in \mathcal{C}, \mathbf{c}_1 \neq \mathbf{c}_2} \{\mathtt{dist}(\mathbf{c}_1, \mathbf{c}_2)\} = \min_{\mathbf{c} \in \mathcal{C}, \mathbf{c} \neq \mathbf{0}} \{\mathtt{dist}(\mathbf{c}, \mathbf{0})\} = \min_{\mathbf{c} \in \mathcal{C}, \mathbf{c} \neq \mathbf{0}} \{\mathtt{wt}(\mathbf{c})\}$. The codewords of any $\mathcal{C}[n,k]$ can be obtained via a $k \times n$ *generator matrix* $\mathbf{G}$ built from k linearly independent vectors in $\mathbb{F}_2^n$ spanning $\mathcal{C}[n,k]$, i.e., $\mathbf{G} = (\mathbf{g_0}, \mathbf{g_1}, \ldots, \mathbf{g_{k-1}})^\intercal \in \mathbb{F}_2^{k \times n}$, and $\forall\, \mathbf{u} \in \mathbb{F}_2^k$, $\mathbf{c}^\intercal = \mathbf{u}^\intercal \mathbf{G}$, $\mathbf{c} \in \mathcal{C} \subset \mathbb{F}_2^n$. A code $\mathcal{C}[n,k]$ admits multiple generator matrices up to a pre-multiplication of $\mathbf{G}$ by a non-singular $k \times k$ matrix over $\mathbb{F}_2$ (i.e., a change of basis).

The set of all vectors in $\mathbb{F}_2^n$ that are orthogonal to the codewords in $\mathcal{C}[n,k]$ is known as *dual code*, $\mathcal{C}[n,k]^\perp$ and can be built via a generator matrix $\mathbf{H}$ composed by $n-k$ linearly independent vectors in $\mathbb{F}_2^n$, which is also known as a *parity-check matrix* of $\mathcal{C}[n,k]$. The quantity $r{=}n{-}k$ is known as *redundancy* of the code. For any $\mathbf{x} \in \mathbb{F}_2^n$, $\mathbf{s} = \mathbf{H}\mathbf{x} \in \mathbb{F}_2^r$, is known as the *syndrome* of $\mathbf{x}$ through $\mathbf{H}$. Being $\mathbf{H}$ a basis of $\mathcal{C}\left[n,k\right]^\perp$, every $\mathbf{c} \in \mathcal{C}\left[n,k\right]$ is s.t. $\mathbf{H}\mathbf{c} = \mathbf{0}_{r\times 1}$, i.e., any codeword in $\mathcal{C}\left[n,k\right]$ has a null syndrome through $\mathbf{H}$.

Definition 1 (Binary Regular Codes). *A binary linear code $\mathcal{C}[n,k]$ is (v, w)-regular if it admits a parity-check matrix, where v and w are the Hamming weights of each column and each row, respectively and $\frac{n-k}{n} = \frac{v}{w}$.*

It is customary to distinguish between regular Low Density Parity-check (LDPC) codes when the code admits at least one sparse parity-check matrix with $v \in \mathcal{O}\left(\log(n)\right)$ [3,32], and regular Medium Density Parity-check (MDPC) codes when the code admits at least one sparse parity-check matrix with $v \in \mathcal{O}\left(\sqrt{n}\right)$ [55].

Definition 2 (Quasi-cyclic Codes). *A linear code $\mathcal{C}[n,k]$, with $n = p\,n_0$, $k = p\,k_0$ (and redundancy $r = n - k = p\,r_0$, $r_0 = n_0 - k_0$) is said to be quasi-cyclic (QC) if its generator and parity-check matrices $\mathbf{G}$ and $\mathbf{H}$ are obtained tiling $p{\times}p$ circulant matrices on a $k_0{\times}n_0$ and a $r_0{\times}n_0$ grid, respectively.*

A quasi-cyclic code with rate $\frac{n_0-1}{n_0}$ (i.e., with $r_0{=}n_0{-}k_0{=}1$) where the $p \times pn_0$ parity-check matrix has a fixed column weight v is (v,vn_0)-regular. QC Low Density Parity-Check (QC-LDPC) codes admit a parity-check matrix with column weight $v \in \mathcal{O}\left(\log(n)\right)$, while QC Medium Density Parity-Check (QC-MDPC) codes admit parity-check matrices with column weight $v \in \mathcal{O}\left(\sqrt{n}\right)$. QC-LDPC/-MDPC codes $\mathcal{C}[n_0p, k_0p]$, with codewords made of n_0 circulant $p \times p$ blocks and rate $\frac{n_0-1}{n_0}$ have an upper bound on the distance of $\leq 2v$ [12].

2.2 Bit Flipping Decoders

An efficient and effective decoder for LDPC/MDPC codes, given the defining sparse parity-check matrix, is the parallel bit flipping decoding procedure introduced in [32], which provides a good engineering trade-off between speed and error correcting capabilities (i.e., low DFR). Given the sparse parity-check matrix $\mathbf{H}$ of the LDPC/MDPC code $\mathcal{C}[n,k]$ and the syndrome $\mathbf{s}$ corresponding to a corrupted codeword $\widetilde{\mathbf{c}} = \mathbf{c} \oplus \mathbf{e}$ with $\mathtt{wt}(\mathbf{e}) = t$, the bit-flipping decoder (reported in Alg. 1) iteratively estimates the most likely value $\bar{\mathbf{e}}$ of the unknown error vector $\mathbf{e}$ starting from an initial null estimate $\bar{\mathbf{e}} = \mathbf{0}_{1\times n}$ and iteratively refining it. In the following, we will refer to values at the end of any iteration of the loop at lines 3–11 adding a superscript integer with the value of the variable $\mathtt{iter}{\in}\{0,1,\ldots,\mathtt{iterMax}\}$ between round brackets, e.g., $\mathbf{s}^{(1)}$ will denote the value of the syndrome vector at the end of the first iteration, while $\mathbf{s}^{(0)}$ denotes the initial syndrome value. We denote as $\bar{\mathbf{d}}{=}\mathbf{e} \oplus \bar{\mathbf{e}}$ the vector of *discrepancies* between the error vector estimate $\bar{\mathbf{e}}$ and the unknown error vector $\mathbf{e}$, referring to the bit positions in $\bar{\mathbf{d}}$ as *d-positions*. The decoder maintains the invariant that, after the i-th iteration, $\mathbf{H}\,\bar{\mathbf{d}}^{(i)} = \mathbf{s}^{(i)}$ with $\bar{\mathbf{d}}^{(i)} = \bar{\mathbf{e}}^{(i)} \oplus \mathbf{e}$. We call each one

Algorithm 1: BIT FLIPPING DECODER

Input: $\mathbf{s}$, $\mathbb{F}_2^{r\times 1}$ syndrome; $\mathbf{H}$, $\mathbb{F}_2^{r\times n}$ parity-check matrix;
`iterMax`, max number of permitted iterations.
Output: $\bar{\mathbf{e}} = [\bar{\mathbf{e}}_0, \ldots, \bar{\mathbf{e}}_{n-1}]$: estimated error vector;
`decodeOk`: Boolean indicating success

1 $\bar{\mathbf{e}} \leftarrow \mathbf{0}$; `iter` $\leftarrow 1$
2 **while** *(*$\mathbf{s} \neq \mathbf{0}$ **and** `iter` $\leq$ `iterMax`*)* **do**
3 **for** j **from** 0 **to** $n-1$ **do**
4 $\mathtt{upc}_j \leftarrow \langle \mathbf{s}, \mathbf{H}_{:,j} \rangle$ `// inner product:` $\sum_i \mathbf{s}_i \mathbf{H}_{i,j}$
5 `th` $\leftarrow$ THRESHOLDCHOICE(`iter`, $\mathbf{s}$)
6 **for** j **from** 0 **to** $n-1$ **do**
7 **if** ($\mathtt{upc}_j \geq$ `th`) **then**
8 $\bar{\mathbf{e}}_j \leftarrow \bar{\mathbf{e}}_j \oplus 1$; $\mathbf{s} \leftarrow \mathbf{s} \oplus \mathbf{H}_{:,j}$
9 `iter` $\leftarrow$ `iter` $+ 1$
10 **if** ($\mathbf{s} = \mathbf{0}$) **then** `decodeOk` $\leftarrow$ `true`
11 **else** `decodeOk` $\leftarrow$ `false`
12 **return** $\bar{\mathbf{e}}$, `decodeOk`

of the equations $\mathbf{H}_{j,:}\bar{\mathbf{d}}^{(i)} = \mathbf{s}_j^{(i)}$, with $0 \leq j \leq r-1$ *parity check equation*, or *parity check* in short, and say that the equation is *satisfied* if the known term $\mathbf{s}_j^{(i)}$ is zero, or *unsatisfied* if it is one. The decoder updates its estimate $\bar{\mathbf{e}}$, trying to detect the d-positions where the discrepancy vector has a non-null entry, with the aim of changing the estimate $\bar{\mathbf{e}}$ so that the discrepancy vector becomes null and therefore $\bar{\mathbf{e}}$ equals the actual error vector $\mathbf{e}$. This is done by counting the number of unsatisfied parity-checks ($\mathtt{upc}_j$) in which a given d-position, the j-th, $0 \leq j \leq n-1$ is involved: this is efficiently done checking how many non-null elements in a column $\mathbf{H}_{:,j}$ (which imply that the j-th d-position is involved in the corresponding parity checks) have a known term of their parity check in $\mathbf{s}$ set to one (line 5). Once all $\mathtt{upc}_j$, for all $0 \leq j \leq n-1$ are computed, the decoder changes the error estimate values in the positions where the corresponding `upc`s are above a threshold $\mathtt{th}^{(i)}$ (lines 8–9), determined as a function of the iteration index `iter` and the current value of the syndrome $\mathbf{s}^{(\mathtt{iter}-1)}$ (line 6). This change in turn influences the discrepancy vector $\bar{\mathbf{d}}$, ideally reducing the number of set terms. The decoder also updates the syndrome value to match the required invariant (line 10) stopping if either a null syndrome is obtained (marking a decoding success, as $\mathbf{H}\bar{\mathbf{d}}^{(i)} = \mathbf{0} \leftrightarrow \bar{\mathbf{d}}^{(i)} = \mathbf{0}$) or if a fixed maximum number of iterations `iterMax` is reached. Bit flipping decoders are said to be *parallel* if they perform all the flipping decisions simultaneously (as in Alg. 1), and *sequential* if they perform one flipping decision at a time, updating all the `upc` values after each flip. In post-quantum cryptosystems, parallel decoders are preferred for efficiency reasons. If the error estimate $\bar{\mathbf{e}}$ does not match the error vector $\mathbf{e}$ after `iterMax` iterations (meaning that $\mathbf{d}^{(\mathtt{iterMax})} \neq \mathbf{0}$) then the decoding algorithm terminates with an incorrect solution. This implies that flipping decoders have a non-zero Decoding Failure Rate (DFR).

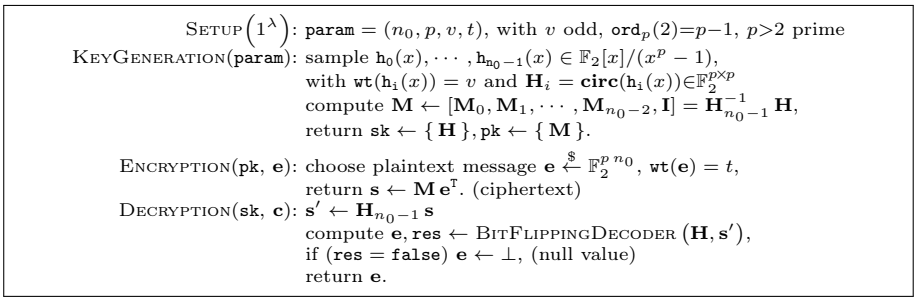


Fig. 1: Niederreiter based QC-MDPC PKE scheme matching BIKE-PKE when $n_0 = 2$ [7] , and the PKE in LEDAcrypt-KEM [13] when $n_0 \in \{2, 3, 4\}$.

2.3 QC-MDPC based Post-quantum Cryptosystems

A popular construction for code-based post-quantum Public Key Encryption (PKE) schemes was proposed by Harald Niederreiter [47]. It relies on the hardness of decoding a random-like linear block code, and on the impossibility of efficiently decoding a structured code when provided only with the systematic form of its scrambled parity-check matrix. Employing either a QC-MDPC or QC-LDPC code $\mathcal{C}[n_0p, k_0p]$, having $n_0-k_0=1$, to reduce the keypair size of a Niederreiter-style PKE was proposed in [14,45], and lead to the design of the Key Encapsulation Methods (KEMs) known as "Bit Flipping Key Encapsulation" (BIKE) [7], and "Low-dEnsity parity-check coDe-bAsed cryptographic systems"-KEM (LEDAcrypt-KEM) [13]. LEDAcrypt-KEM takes as a design parameter also the rate $\frac{n_0-1}{n_0} \in \{\frac{1}{2}, \frac{2}{3}, \frac{3}{4}\}$ of the underlying QC code, while BIKE assumes a fixed number of codeword blocks $n_0 = 2$, and consequently a fixed $\frac{1}{2}$ code rate.

The first proposal of the LEDAcrypt-KEM relied on QC-LDPC codes and a dedicated sampling procedure [9], later found to be affected by weak key selections [6]. In the following, we consider the LEDAcrypt-KEM instantiation of Niederreiter's cryptoscheme, while retaining QC-MDPCs as codes [13].

The three algorithms (key generation, encryption and decryption) of the Niederreiter PKE construction for both LEDAcrypt-KEM and BIKE are reported in Fig. 1. The KeyGeneration procedure samples the private key in the form of the $p \times n_0p$ parity-check matrix $\mathbf{H} = [\mathbf{H}_0, \cdots, \mathbf{H}_{n_0-1}]$ picking uniformly at random n_0 $p\times p$ circulant blocks, and odd column weight v. The public key $\mathbf{M} = [\mathbf{M}_0, \ldots, \mathbf{M}_{n_0-2}, \mathbf{I}]$ is obtained by computing the left systematic form of $\mathbf{H}$, which involves multiplying each of its blocks by the inverse of $\mathbf{H}_{n_0-1}$. Since v is odd, when p is prime, the condition $\mathtt{ord}_p(2) = p-1$ guarantees that any $p \times p$ circulant block with row weight v is invertible [13]. The plaintext message taken as input by the Encryption primitive of a KEM (as in BIKE and LEDAcrypt-KEM) is defined as an error vector $\mathbf{e} \in \mathbb{F}_2^n$ that is randomly and uniformly sampled among the vectors having Hamming weight equal to t. The ciphertext is computed as the syndrome of $\mathbf{e}$ through $\mathbf{M}$. The Decryption procedure multiplies the received syndrome by $\mathbf{H}_{n_0-1}$, turning it into the syndrome of the same error $\mathbf{e}$ through the private parity-check matrix $\mathbf{H}$, and

employs a parallel bit flipping decoder to retrieve $\mathbf{e}$ by decoding the syndrome of the QC-MDPC code defined by $\mathbf{H}$.

We recall the computationally hard problems that a passive attacker must solve and on which the choice of cryptosystem parameters should be based.

Definition 3 (Key Recovery Attack). *Given a public key, i.e., a binary QC-MDPC parity-check matrix* $\mathbf{M} \in \mathbb{F}_2^{p \times pn_0}$ *in systematic form, compute the corresponding private key* $\mathbf{H} \in \mathbb{F}_2^{p \times pn_0}$*, having column and row weight of its* n_0 *circulant blocks equal to* v*.*

Definition 4 (Message Recovery Attack). *Given a binary QC-MDPC parity-check matrix* $\mathbf{M} \in \mathbb{F}_2^{p \times pn_0}$ *in systematic form, and the syndrome* $\mathbf{s} = \mathbf{He} \in \mathbb{F}_2^p$ *of a weight* t *unknown error vector* $\mathbf{e} \in \mathbb{F}_2^{pn_0}$ *through* $\mathbf{H} \in \mathbb{F}_2^{p \times pn_0}$*, compute* $\mathbf{e}$*.*

KRA and MRA have been proven NP-hard problems (as their decision version is NP-complete) for random codes [16,57], and are conjectured-hard for QC codes. Information Set Decoding (ISD) constitutes the dominant paradigm for KRA and MRA. ISD is a probabilistic decoding strategy that aims to reconstruct a low-weight error vector (or codeword) by repeatedly selecting a random subset of coordinates that is conjectured to capture a favorable structural configuration, and solving the induced linear system. As a Las Vegas algorithm, ISD always outputs a correct solution upon success, while its running time is governed by the inverse of the per-iteration success probability.

Early refinements of Prange's original proposal [52] enlarge the class of admissible error configurations [40] and redistribute the error weight by introducing structured zero windows [41], increasing the per-iteration success probability at the price of additional computational overhead. Later developments exploits meet-in-the-middle strategies, recasting decoding as a structured, multi-level list-collision problem. Alg.s in this line of work [54,27,31,42,15] leverage exponential memory, layered list merging, and approximate matching techniques to improve asymptotic complexity. Subsequent advances reinterpret collision steps as nearest-neighbor searches [43,22,28], or adapt lattice-inspired sieving methods to the Hamming metric by generating increasingly promising candidates [34,26]. Quantum adaptations of these advanced variants provide only limited asymptotic speedups, with their practical relevance constrained by strong quantum memory assumptions [37,38]. A detailed description of the various ISD attacks is provided in Appendix A.

Besides the passive attacker scenario, a crucial observation for the security of the cryptosystem is that an active attacker is able to derive information on the private QC-MDPC matrix if the iterative bit flipping procedure fails to decode a given syndrome [35,59]. The security against active attackers of the PKE scheme in Fig. 1, and the INDistinguishability under adaptive Chosen Ciphertext Attack (IND-CCA2) property of the resulting KEM, equally depend on both the computational hardness of passive message and key recovery attacks, and a reliable Decoding Failure Rate (DFR) estimation technique [36].

2.4 Near codewords

Near codewords [8,58] are error vectors with low Hamming weight that, due to the set positions being related to the structure of quasi-cyclic parity check matrices, prove to be hard to decode by iterative decoders employing such codes. They are defined as follows:

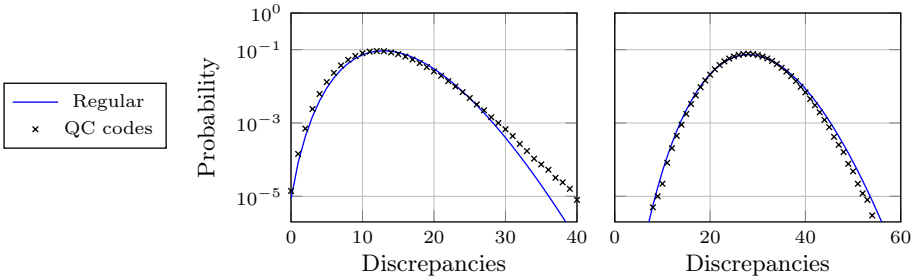
Definition 5 (Near codeword). *Let* $\mathbf{e} = \left[\mathbf{e}_{(0)} \mid \mathbf{e}_{(1)} \mid \ldots \mid \mathbf{e}_{(n_0-1)}\right]$*, where each* $\mathbf{e}_{(j)}$ *is a vector of size* p*. The vector* $\mathbf{e}$ *is a near codeword if it is null save for a single* $\mathbf{e}_{(i_0)}$ *for which the following holds:* $\mathbf{e}_{(i_0)} = (\mathbf{H}_{:,pi_0+l})^{\mathtt{T}}, 0 \leq l \leq p-1$.

If $\mathbf{e}$ is a near codeword, then $\mathtt{wt}(\mathbf{e}) = v$ and $\mathtt{wt}(\mathbf{s}) = \mathtt{wt}(\mathbf{H}\mathbf{e}^{\mathtt{T}}) = v$: this stems from the fact that the result of $\mathbf{H}\mathbf{e}^{\mathtt{T}}$ is obtained multiplying one circulant block of $\mathbf{H}$, which is associated to a polynomial in $\mathbb{F}_2[x]/(x^p-1)$ by the non null, p elements long portion of $\mathbf{e}$, $\mathbf{e}_{(i_0)}$. The same operation can be reinterpreted as a polynomial multiplication in $\mathbb{F}_2[x]/(x^p-1)$ of the polynomial $\mathtt{e}(x)$, having the elements $\mathbf{e}_{(j)}$ as coefficients, by the polynomial associated to the block of $\mathbf{H}$, $\mathtt{h}(x)$. Observe now that, by definition of near codeword, $\mathtt{h}(x)$ can be rewritten as $\mathtt{e}(x)x^l \bmod x^p-1$, therefore the multiplication result is $\mathtt{h}(x)\mathtt{e}(x) = (\mathtt{e}(x))^2x^l$. Recalling that, in $\mathbb{F}_2[x]/(x^p-1)$ taking the square of a polynomial amounts to permuting its coefficients [13] we have that the number of non null monomials in $(\mathtt{e}(x))^2x^l$ and thus the Hamming weight of $\mathbf{s}$ is the same as $\mathbf{H}_{:,pi_0+l}$, i.e., v.

Iterative decoders are fixed point procedures that try to minimize the Hamming weight of the syndrome $\mathbf{s}$, and the aforementioned situation where the syndrome weight matches the one of a single column of $\mathbf{H}$, but the number of columns added together to form the syndrome is v (with an expected weight in the range of v^2) constitutes an unrecoverable pattern. While near codewords are a particularly critical case, errors $\mathbf{e}$ having set terms which share in a significant amount of the positions of set bits with a near codeword still prove tougher to decode than the average ones, by causing the discrepancy vector $\bar{\mathbf{d}}^{(i)}$ to converge to such near codeword [8].

3 Closed Form DFR Estimation for QC-MDPC Codes

In this section, we report a complete closed-form model for the DFR of QC-MDPC codes, decoded with the parallel bit flipping decoder (Alg. 1) having $\mathtt{iterMax} = 3$. To do so, we separately consider the contribution to the overall DFR provided by different error patterns. After listing the set of assumptions on which our DFR model relies, we analyse generic *hard-to-decode* errors (Theorem 1), that trigger a decoding failure not because of the QC structure of the parity check matrix, but because the induced flips cause discrepancies to remain after three decoding iterations. Then, we model the failures induced by *near codewords* (Theorem 2), taking into account the quasi-cyclic structure of the codes employed in BIKE and LEDAcrypt. Moreover, we define a novel class of errors, *half codewords*, studying its impact on the DFR in the floor regime (Theorem 3). With such analysis, we are able to provide a conservative estimate of the DFR of QC-MDPC codes (Theorem 4).



(a) QC-LDPC (b) QC-MDPC

Fig. 2: Discrepancies left after one iteration by (v, w)-regular codes and QC codes with rate $\frac{1}{2}$. Fig. (a): QC-LDPC codes (in weak-keys regime) with $n = 2426$, $v = 9$, $t = 25$, $\mathtt{th}^{(1)} = 5$, Fig. (b): QC-MDPC codes with $n = 27382$, $v = 69$, $t = 130$, $\mathtt{th}^{(1)} = 41$.

3.1 Assumptions

In the derivation of our closed-form DFR estimation model, we rely on two assumptions, related to the outcome of the parity-check equations and of the flipping decisions.

Assumption 1 *The probability that a parity check, involving a specific bit-position, is unsatisfied at iteration* `iter` *is independent from the outcome at iteration* `iter` *of the other* $v - 1$ *parity checks involving the same bit-position.*

Assumption 2 *The probability that a specific bit-position is flipped at iteration* `iter` *is independent from the outcome of the simultaneous flipping decisions taken on other bit-positions at iteration* `iter`*.*

Assumption 1 is the same employed in many DFR estimation model of iterative decoders, including [4,8], with [5] (the extended version of [4]) providing evidence that such assumption yields a conservative failure rate estimation. Assumption 2 is a less stringent version of Assumption 2 in [8], and is also used implicitly in [4]. While Assumption 1 always predicts experimental data accurately, we found that Assumption 2 does not approximate the behaviour of the parallel bit-flipping decoder accurately in the parameter regions where the average DFR is dominated by *weak keys*.

Weak keys are QC sparse codes with significantly higher DFR than the average DFR of (w,v)-regular sparse codes with the same parameters. These keys are characterized by large intersections between columns of the parity check matrix [58,25,59], causing some d-positions to share many parity checks and to be flipped "together", leading to incorrect flips be positively correlated. This phenomenon takes place in QC-LDPC codes with very low weight (e.g., $v \leq 13$), and tends to disappear when the length and density of the parity check matrix increase, as accidental column intersections with large cardinality are more rare. Fig. 2 shows an example of such phenomenon.

Table 1: Numerical results of the test performed on the parameter sets shown in Tab. 2 for long-term keys. $\hat{X}$ indicates the empirical mean of the number of discrepancies left after one iteration (computed over 10^6 samples), $\hat{X}^{\text{ub}}_{0.99}$ is the 99% confidence upper bound on the mean, S^2 is the empirical variance, $S^{2,\text{ub}}_{\chi^2,0.99}$ is the 99% confidence upper bound on the variance computed with a χ^2 test, $S^{2,\text{ub}}_{\text{boot},0.99}$ is the 99% confidence upper bound on the variance computed with bootstrapping (thus without normality assumption on the target distribution). μ and σ^2 indicate mean and variance returned by the model. The last column reports the DFR lower bound given by weak keys [59], computed with 10^8 decoding actions for increasing values of m (with $\epsilon = 1$) until no failure takes place.

λ	n_0	Empirical					Model		Weak
		$\hat{X}$	$\hat{X}^{\text{ub}}_{0.99}$	S^2	$S^{2,\text{ub}}_{\chi^2,0.99}$	$S^{2,\text{ub}}_{\text{boot},0.99}$	μ	σ^2	Keys [59]
128	2	28.38	28.40	26.54	26.63	26.63	28.57	29.23	-146.58
128	3	11.21	11.22	10.82	10.86	10.86	11.39	12.11	-170.70
128	4	6.40	6.41	6.24	6.26	6.26	6.56	8.17	-186.82
128	5	5.66	5.67	6.02	6.04	6.04	5.77	7.29	-208.83
192	2	47.04	47.06	45.70	45.85	45.85	47.22	48.69	-245.66
192	3	18.12	18.13	18.38	18.44	18.44	18.29	19.73	-280.25
192	4	12.24	12.25	13.31	13.35	13.35	12.27	15.63	-361.17
192	5	8.41	8.42	9.16	9.19	9.20	8.49	10.70	-323.20
256	2	59.66	59.68	57.62	57.82	57.82	59.88	64.41	-333.16
256	3	25.97	25.98	27.80	27.89	27.90	26.12	28.49	-427.14
256	4	35.99	36.00	37.66	37.79	37.79	36.30	43.20	-550.69
256	5	9.52	9.53	10.30	10.34	10.34	9.63	10.51	-460.56

In the parameter regimes where weak keys are statistically relevant, the distribution of the number of errors appears to have heavier tails than the one of a (v, w)-regular codes with the same parameters, in turn causing the probability of a high number of errors to be larger (and increasing the average DFR). In particular, the discrepancies distribution exhibits a high *variance*, caused by the *covariance* between flips (which is assumed to be 0 under Assumption 2) being positive, due to the correlated columns. To detect whether weak keys play a significant role on average DFR of QC-MDPC codes with our selected parameters, we thus suggest to analyze the distribution of the number of discrepancies after one decoding iteration (i.e. the distribution of $\mathtt{wt}(\mathbf{d}^{(1)})$).

We performed statistical tests on the number of discrepancies left after one iteration for all the cryptographic-grade parameter sets of interest, showing with at least 99% confidence that the variance of such distribution is strictly smaller that the variance predicted by the closed-form model. This indicates that the behaviour the bit flipping decoder of such codes is well described by Assump-

tion 2. For additional validation, we verified that the DFR lower bound induced by the class of weak keys presented in [59] is smaller than the required DFR for all security levels. The numerical results of these tests can be found in Tab. 1.

The long-term security of the QC-MDPC based cryptosystem employing our chosen parameter sets, for which our model provides a DFR estimate below the required level 2^{λ}, relies on the following additional assumption.

Assumption 3 *The decoding failures of cryptographic-grade QC-MDPC codes decoded with a 3-iterations parallel bit flipping decoder are caused either by hard-to-decode errors, or by convergence to a near codeword or a half codeword.*

This assumption is backed by decades of study and experiments on QC-MDPC codes: while near codewords were the only structure known to be problematic QC-MDPC codes, half codewords constitute the only generalization of such structure that can lead to decoding failure with non-negligible probability. This is particularly true for code parameters that lie in the waterfall regime or at the very beginning of the error floor, coinciding with the parameters of interest for QC-MDPC based cryptosystems. Note that we are not making any assumption about which family of error structures dominates the DFR either in the waterfall or error floor regime. Additionally, with Assumption 3 we are considering the average DFR of QC-MDPC codes to be always higher (i.e., worse) than the average DFR of (v, w)-regular codes.

3.2 Hard-to-decode Errors

In the following, we derive a closed-form probabilistic model for the DFR of the parallel bit-flipping decoder induced by generic *hard-to-decode* errors (i.e. errors that are not related to the QC structure of the parity check matrix). In doing so, we can assume the parity check matrix to be (v, w)-regular, $w = vn_0$, and we build on the model presented in [4], that assesses the DFR of the same decoder with $\texttt{iterMax} = 2$, applied to the same code family. We model the bit flipping decoder employing thresholds depending only on the iteration index, and not on the Hamming weight of the syndrome.

Random Variables Modeling the Bit-flipping Behavior. We represent crucial quantities in the decoder behavior as random variables, and describe how to compute their distributions. To provide a visual intuition of the quantities, we depict in Fig. 3 a running example with one parity check equation $\mathbf{H}_{0,:}$ with $n = 12, w = 8$, together with its corresponding syndrome term $\mathbf{s}_0$ and the history of the discrepancy vector in iterations 0 through 2: $\bar{\mathbf{d}}^{(0)}, \bar{\mathbf{d}}^{(1)}, \bar{\mathbf{d}}^{(2)}$. We consider $\mathcal{W}$ as the r.v. representing the initial syndrome weight $\texttt{wt}(\mathbf{s}^{(0)})$, and we move onto modeling the evolution of the discrepancy vectors $\bar{\mathbf{d}}^{(i)}$. We categorize the d-positions after each iteration i in 2^{i+1} sets, depending on the history of the values taken by them in the discrepancy vectors $\bar{\mathbf{d}}^{(j)}$, $0 \leq j \leq i$. We denote as $\mathbf{J}_{\texttt{cat}}$, where $\texttt{cat}$ is a binary string of length $i + 1$, the set of all d-positions for which the sequence of corresponding values in the discrepancy vectors, in subsequent iterations, matches $\texttt{cat}$. As an example, in Fig. 3 $\mathbf{J}_{00}$ contains the

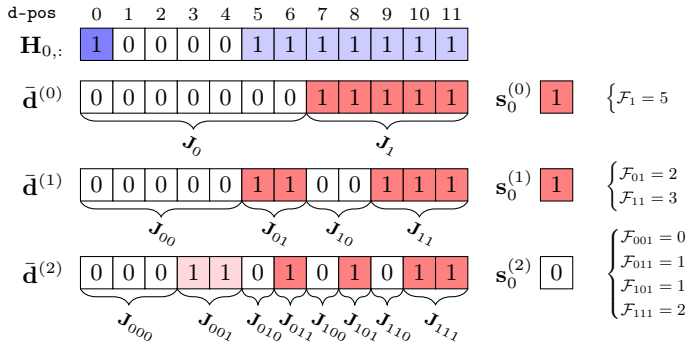


Fig. 3: Graphical representation of the bits in a parity-check with $n = 12$ and $w = 8$, that includes a bit in $\mathbf{J}_{000}$ (highlighted in blue), at the beginning of the decoding process ($\texttt{iter} = 0$) and after the 1st and 2nd iteration. Each entry of $\bar{\mathbf{d}}^{(\texttt{iter})}$ is equal to 1 where $\mathbf{e}$ and $\bar{\mathbf{e}}^{(\texttt{iter})}$ differ and 0 where $\mathbf{e}$ and $\bar{\mathbf{e}}^{(\texttt{iter})}$ match. Curly brackets indicate in which set $\mathbf{J}_a$, $\mathbf{J}_{ab}$ $\mathbf{J}_{abc}$ $(a, b, c \in \{0, 1\})$ each d-position is contained at each iteration.

first five positions from the left for which $\bar{\mathbf{d}}^{(0)}$ and $\bar{\mathbf{d}}^{(1)}$ have null values (i.e., $\mathbf{J}_{00} = \{0, 1, 2, 3, 4\}$), while $\mathbf{J}_{101} = \{8\}$, as $\bar{\mathbf{d}}_8^{(0)} = 1$, $\bar{\mathbf{d}}_8^{(1)} = 0$ and $\bar{\mathbf{d}}_8^{(2)} = 1$. We denote as $\mathcal{E}_{\texttt{cat}}$ the random variables associated with the cardinality of each $\mathbf{J}_{\texttt{cat}}$. Additionally, given a specific parity-check equation, we denote as $\mathcal{F}_{\texttt{cat}}$ the number of d-positions in $\mathbf{J}_{\texttt{cat}}$ that are also involved in the equation. In Fig. 3, we have that $\mathcal{F}_{001} = 0$ since $\mathbf{J}_{001} = \{3, 4\}$, but only d-positions 0 and 5 to 11 are involved in the parity check. The outcome of a specific parity check depends on the parity of $\mathcal{F}_1$ during the first iteration, $\mathcal{F}_{01} + \mathcal{F}_{11}$ during the second iteration, $\mathcal{F}_{001} + \mathcal{F}_{011} + \mathcal{F}_{101} + \mathcal{F}_{111}$ during the third iteration.

Derivation of the Distributions of Random Variables. In the following, we will refer to the results of the analysis of the first two iterations, derived in [4]. Our goal is to compute the probability $p_{\texttt{flip|cat}}$ of flipping the value of a d-position in $\mathbf{J}_{\texttt{cat}}$ during the third iteration, for all three-bit strings $\texttt{cat}$. We report the derivation for $\texttt{cat} = 000$ as the probabilities for the other categories are derived in the same fashion, up to a matter of indexes. In this analysis, we are going to fix the cardinalities of $\mathbf{J}_{01}$, $\mathbf{J}_{11}$ as $\mathcal{E}_{01}{=}\epsilon_{01}$ and $\mathcal{E}_{11}{=}\epsilon_{11}$ (therefore $|\mathbf{J}_{01}| + |\mathbf{J}_{11}| = \epsilon_{01} + \epsilon_{11} = \texttt{wt}(\bar{\mathbf{d}}^{(1)})$), and the cardinalities of $\mathbf{J}_{001}$, $\mathbf{J}_{011}$, $\mathbf{J}_{101}$ and $\mathbf{J}_{111}$ as $\mathcal{E}_{001} = \epsilon_{001}$, $\mathcal{E}_{011} = \epsilon_{011}$, $\mathcal{E}_{101} = \epsilon_{101}$ and $\mathcal{E}_{111} = \epsilon_{111}$ (therefore $|\mathbf{J}_{001}| + |\mathbf{J}_{011}| + |\mathbf{J}_{101}| + |\mathbf{J}_{111}| = \epsilon_{001} + \epsilon_{011} + \epsilon_{101} + \epsilon_{111} = \texttt{wt}(\bar{\mathbf{d}}^{(2)})$).

The first step to model $p_{\texttt{flip|000}}$ is to characterize the behaviour of bits appearing in specific parity-check equations, under assumptions on the outcome of such parity check. This analysis will allow us to determine how parity checks are affected by the flips resulting at the end of the second iteration. We begin with the following proposition.

Proposition 1. *Let $p_{\texttt{u|0}}$ be the probability that a parity check involving a bit in $\mathbf{J}_0$ (i.e., a correct bit) is initially unsatisfied. Moreover, let $p_{\texttt{u|00,s}}$ (resp. $p_{\texttt{u|00,u}}$) be the probability that a parity check, involving a bit in $\mathbf{J}_{00}$ and initially satisfied*

(resp. unsatisfied), becomes unsatisfied after the first iteration (these probabilies are named $p_{\texttt{unsat}|0}$*,* $p_{00,\texttt{BecomeUnsat}}$ *and* $p_{00,\texttt{StayUnsat}}$ *respectively in the extended version of [4]). Given that a bit in* $\mathbf{J}_{00}$ *appears in a parity check that is* satisfied *both before and after the first iteration, the probability that such bit is flipped in the second iteration is* $p_{\texttt{flip}|00,\texttt{ss}} =$

$$\sum_{\mu=0}^{\texttt{th}^{(1)}} \left(\frac{\text{BIN}(v-1, p_{\texttt{u}|0}, \mu)}{\sum_{x=0}^{\texttt{th}^{(1)}} \text{BIN}(v-1, p_{\texttt{u}|0}, x)} \right) \cdot \sum_{\substack{\mu_{\texttt{s}}, \mu_{\texttt{u}} \\ \mu_{\texttt{s}}+\mu_{\texttt{u}} \geq \texttt{th}^{(2)}}} \text{BIN}(\mu, p_{\texttt{u}|00,\texttt{u}}, \mu_{\texttt{u}}) \text{BIN}(v-\mu-1, p_{\texttt{u}|00,\texttt{s}}, \mu_{\texttt{s}})$$

Proof. In order for a correct bit to be flipped in the second iteration, but not in the first, two events must take place simultaneously: 1) its `upc` before the first iteration equals a value μ below $\texttt{th}^{(1)}$, and 2) a total of at least $\texttt{th}^{(2)}$ parity checks involving the bit must become (or remain) unsatisfied after the first iteration. The probability of the first event, under Assumption 1, is $\text{BIN}(v-1, p_{\texttt{u}|0}, \mu)$ normalized over all values of μ smaller than $\texttt{th}^{(1)}$, where the term $v-1$ is justified by the fact that one of the parity checks involving the bit is assumed to be initially satisfied. For the second event, we must distinguish the case for parity equations that remain unsatisfied ($\mu_{\texttt{u}}$) and become unsatisfied ($\mu_{\texttt{s}}$). The probability that a check of the former type remains unsatisfied is $p_{\texttt{u}|00,\texttt{u}}$, and the number of initially unsatisfied checks is μ. The probability that a check of the latter type becomes unsatisfied is $p_{\texttt{u}|00,\texttt{s}}$, and the number of initially satisfied checks is $v-\mu$. However, under the initial hypothesis that one of such checks is satisfied also after the first iteration, the number of checks that can become unsatisfied is $v-\mu-1$. □

The probabilities $p_{\texttt{u}|00,\texttt{su}}$, $p_{\texttt{u}|00,\texttt{us}}$ and $p_{\texttt{u}|00,\texttt{uu}}$ (and the corresponding probabilities for bits $\mathbf{J}_{01}$, $\mathbf{J}_{10}$, $\mathbf{J}_{11}$) can be derived through an analogous line of reasoning. The next step to compute $p_{\texttt{flip}|000}$ is to model the behaviour of parity checks that include a bit in $\mathbf{J}_{000}$, during the *third* iteration.

Proposition 2. *Consider a parity-check equation, involving a bit in* $\mathbf{J}_{000}$*, having* $\mathcal{F}_1 = f_1$*,* $\mathcal{F}_{01} = f_{01}$ *and* $\mathcal{F}_{11} = f_{11}$*. The probability* $\chi^{\texttt{odd}}_{001}(f_1, f_{01}, f_{11})$ *that* $\mathcal{F}_{001}$ *is odd in such parity check is* $\chi^{\texttt{odd}}_{001}(f_1, f_{01}, f_{11}) =$

$$\sum_{f_{001},\texttt{odd}} \text{FNCHYPG}\left(f_{001}; \epsilon_{001}, w-f_1-f_{01}-1, n-t-\epsilon_{01}-1, \widetilde{\rho}, p_{\texttt{flip}|00}\right)$$

where:

$$\widetilde{\rho} = \begin{cases} p_{\texttt{flip}|00,\texttt{ss}} & \text{if } f_1 \text{ even, } f_{01}+f_{11} \text{ even} \\ p_{\texttt{flip}|00,\texttt{su}} & \text{if } f_1 \text{ even, } f_{01}+f_{11} \text{ odd} \\ p_{\texttt{flip}|00,\texttt{us}} & \text{if } f_1 \text{ odd, } f_{01}+f_{11} \text{ even} \\ p_{\texttt{flip}|00,\texttt{uu}} & \text{if } f_1 \text{ odd, } f_{01}+f_{11} \text{ odd} \end{cases}$$

Proof. In a parity check with $\mathcal{F}_1{=}f_1$, $\mathcal{F}_{01}{=}f_{01}$, and $\mathcal{F}_{11}{=}f_{11}$, the total number of bits in $\mathbf{J}_{00}$ involved in the check (excluding the specific bit under analysis) is $w{-}f_1{-}f_{01}{-}1$. Globally, if $\mathcal{E}_{01}{=}\epsilon_{01}$ and $\mathcal{E}_{11} = \epsilon_{11}$, the total number of bits in

$\mathbf{J}_{00}$ (excluding the specific bit under analysis) is $n-t-\epsilon_{01}-1$. During the second iteration, the probability $\widetilde{\rho}$ of flipping a bit in $\mathbf{J}_{00}$ that is also involved in the check, under Assumption 2, is one of $p_{\mathtt{flip}|00,\mathtt{ss}}, \ldots, p_{\mathtt{flip}|00,\mathtt{uu}}$ depending on the outcome of the parity check before and after the first iteration. On the other hand, the probability of flipping a bit in $\mathbf{J}_{00}$ that is not involved in the check is simply $p_{\mathtt{flip}|00}$. After conditioning on the total number of positions in $\mathbf{J}_{00}$ that are incorrectly flipped during the second iteration ($\mathcal{E}_{001} = \epsilon_{001}$), the flips happening within the parity check on d-positions in $\mathbf{J}_{00}$ are no longer independent. This implies that the random variable $\mathcal{F}_{001}$ (modeling the number of such incorrect flips) is bound to Fisher's noncentral hypergeometric distribution, since the total number of incorrect flips is fixed but the d-positions inside and outside the parity check have a different prior probability of being flipped. The probability that $\mathcal{F}_{001}$ is odd can be computed as the sum of such probability distribution over the odd terms. □

The probabilities $\chi^{\mathtt{odd}}_{011}(f_1, f_{01}, f_{11})$, $\chi^{\mathtt{odd}}_{101}(f_1, f_{01}, f_{11})$, $\chi^{\mathtt{odd}}_{111}(f_1, f_{01}, f_{11})$ that the number of discrepancies of each kind appearing in the check is odd, can be derived in the same way. We now have all the tools to compute the following probability.

Proposition 3. *Consider a parity check that involves a bit in $\mathbf{J}_{000}$, and that is satisfied both before and after the first iteration. The probability $p_{\mathtt{u}|000,\mathtt{ss}}$ that such check becomes unsatisfied after the second iteration is:*

$$p_{\mathtt{u}|000,\mathtt{ss}} = \frac{\sum_{\substack{f_1,f_{01},f_{11} \\ f_1\mathtt{even} \\ f_{01}+f_{11}\mathtt{even}}} \Pr(\mathcal{F}_1, \mathcal{F}_{01}, \mathcal{F}_{11} = f_1, f_{01}, f_{11}) \cdot \Psi(f_1, f_{01}, f_{11})}{\sum_{\substack{f_1,f_{01},f_{11} \\ f_1\mathtt{even} \\ f_{01}+f_{11}\mathtt{even}}} \Pr(\mathcal{F}_1, \mathcal{F}_{01}, \mathcal{F}_{11} = f_1, f_{01}, f_{11})}$$

where

$$\Psi(f_1, f_{01}, f_{11}) := \frac{1}{2} - \frac{1}{2}\left(\prod_{\mathtt{cat}\in\{001,011,101,111\}} (1 - 2\cdot\chi^{\mathtt{odd}}_{\mathtt{cat}}(f_1, f_{01}, f_{11}))\right)$$

Proof. During the third decoding iteration, the outcome of a parity check depends on the parity of the number of discrepancies involved, that is, $\mathcal{F}_{001} + \mathcal{F}_{011} + \mathcal{F}_{101} + \mathcal{F}_{111}$. Fixing $\mathcal{F}_1 = f_1$, $\mathcal{F}_{01} = f_{01}$ and $\mathcal{F}_{11} = f_{11}$, the probability of $\mathcal{F}_{\mathtt{cat}}$ being odd, $\mathtt{cat} \in \{001, 011, 101, 111\}$, is $\chi^{\mathtt{odd}}_{\mathtt{cat}}(f_1, f_{01}, f_{11})$ (Proposition 2). Using the formula given in [32, Lemma 1], the probability that the sum of the four terms yields an odd result is $\Psi(f_1, f_{01}, f_{11})$. Assuming $\mathcal{F}_1$ and $\mathcal{F}_{01} + \mathcal{F}_{11}$ to be even, the probability of the parity check being unsatisfied after the second iteration can be computed by averaging $\Psi(f_1, f_{01}, f_{11})$ over all values f_1, f_{01}, f_{11} respecting the hypothesis. The p.m.f. $\Pr(\mathcal{F}_1, \mathcal{F}_{01}, \mathcal{F}_{11} = f_1, f_{01}, f_{11})$ is a corollary of the analysis in the extended version of [4] ([5, Propositions 8-10]). □

The probabilities $p_{\mathtt{u}|000,\mathtt{su}}$, $p_{\mathtt{u}|000,\mathtt{us}}$, $p_{\mathtt{u}|000,\mathtt{uu}}$ can be computed by changing the parity of $\mathcal{F}_1$ and $\mathcal{F}_{01} + \mathcal{F}_{11}$ in the previous statement. Our next goal is to

employ the probabilities $p_{\mathtt{u}|000,\mathtt{ss}}, \ldots, p_{\mathtt{u}|000,\mathtt{uu}}$ to model the distribution of the upc of d-positions in $\mathbf{J}_{000}$, with the aim of computing $p_{\mathtt{flip}|000}$ as the probability that the upc in the third iteration is greater than or equal to the threshold $\mathtt{th}^{(3)}$.

Proposition 4. *Consider a bit in* $\mathbf{J}_{000}$*. Assume the* upc *value in the corresponding d-position is* μ *during the first iteration (*$0 \leq \mu < \mathtt{th}^{(1)}$*). Moreover, let* $\mu_{\mathtt{s}}$ *be the number of satisfied checks involving the bit that* ***become*** *unsatisfied after the first iteration, and* $\mu_{\mathtt{u}}$ *the number of unsatisfied checks that* ***remain*** *unsatisfied after the first iteration. Under Assumption 1, the probability that* $\mu_{\mathtt{ss}}$ *parity checks become unsatisfied after the second iteration, out of* $v - \mu - \mu_{\mathtt{s}}$ *that are satisfied both before and after the first iteration, is:*

$$\Phi_{\mathtt{ss}}(\mu, \mu_{\mathtt{s}}, \mu_{\mathtt{u}}, \mu_{\mathtt{ss}}) := \textsc{Bin}(v - \mu - \mu_{\mathtt{s}}, p_{\mathtt{u}|000,\mathtt{ss}}, \mu_{\mathtt{ss}})$$

If the parity check is unsatisfied either in the first or second iteration, this probability becomes $\Phi_{\mathtt{su}}(\mu, \mu_{\mathtt{s}}, \mu_{\mathtt{u}}, \mu_{\mathtt{su}}) := \textsc{Bin}(\mu_{\mathtt{s}}, p_{\mathtt{u}|000,\mathtt{su}}, \mu_{\mathtt{su}})$*,* $\Phi_{\mathtt{us}}(\mu, \mu_{\mathtt{s}}, \mu_{\mathtt{u}}, \mu_{\mathtt{us}}) := \textsc{Bin}(\mu - \mu_{\mathtt{u}}, p_{\mathtt{u}|000,\mathtt{us}}, \mu_{\mathtt{us}})$*,* $\Phi_{\mathtt{uu}}(\mu, \mu_{\mathtt{s}}, \mu_{\mathtt{u}}, \mu_{\mathtt{uu}}) := \textsc{Bin}(\mu_{\mathtt{u}}, p_{\mathtt{u}|000,\mathtt{uu}}, \mu_{\mathtt{uu}})$*.*

Proposition 5. *Let* $p_{\mathtt{u}|0}$ *be the probability that a parity check involving a bit in* $\mathbf{J}_0$ *(i.e., a correct bit) is initially unsatisfied. Moreover, let* $p_{\mathtt{u}|00,\mathtt{s}}$ *(resp.* $p_{\mathtt{u}|00,\mathtt{u}}$*) be the probability that a parity check, involving a bit in* $\mathbf{J}_{00}$ *and initially satisfied (resp. unsatisfied), becomes unsatisfied after the first iteration. The probability* $p_{\mathtt{flip}|000}$ *of flipping a bit in* $\mathbf{J}_{000}$ *during the third iteration is:*

$$p_{\mathtt{flip}|000} = \frac{\sum\limits_{\substack{\mu,\mu_{\mathtt{s}},\mu_{\mathtt{u}} \\ \mu < \mathtt{th}^{(1)} \\ \mu_{\mathtt{s}}+\mu_{\mathtt{u}} < \mathtt{th}^{(2)}}} \theta(\mu, \mu_{\mathtt{s}}, \mu_{\mathtt{u}}) \cdot \lambda(\mu, \mu_{\mathtt{s}}, \mu_{\mathtt{u}})}{\sum\limits_{\substack{\mu,\mu_{\mathtt{s}},\mu_{\mathtt{u}} \\ \mu < \mathtt{th}^{(1)} \\ \mu_{\mathtt{s}}+\mu_{\mathtt{u}} < \mathtt{th}^{(2)}}} \theta(\mu, \mu_{\mathtt{s}}, \mu_{\mathtt{u}})}$$

where

$$\theta(\mu, \mu_{\mathtt{s}}, \mu_{\mathtt{u}}) := \textsc{Bin}(v, p_{\mathtt{u}|0}, \mu) \cdot \textsc{Bin}(v - \mu, p_{\mathtt{u}|00,\mathtt{s}}, \mu_{\mathtt{s}}) \cdot \textsc{Bin}(\mu, p_{\mathtt{u}|00,\mathtt{u}}, \mu_{\mathtt{u}})$$

$$\lambda(\mu, \mu_{\mathtt{s}}, \mu_{\mathtt{u}}) := \sum_{\substack{\mu_{\mathtt{ss}},\mu_{\mathtt{su}},\mu_{\mathtt{us}},\mu_{\mathtt{uu}} \\ \mu_{\mathtt{ss}}+\mu_{\mathtt{su}}+\mu_{\mathtt{us}}+\mu_{\mathtt{uu}} \geq \mathtt{th}^{(3)}}} \prod_{\mathtt{cat} \in \{\mathtt{ss},\mathtt{su},\mathtt{us},\mathtt{uu}\}} \Phi_{\mathtt{cat}}(\mu, \mu_{\mathtt{s}}, \mu_{\mathtt{u}}, \mu_{\mathtt{cat}})$$

Proof. Under Assumption 1, $\theta(\mu, \mu_{\mathtt{s}}, \mu_{\mathtt{u}})$ is the probability that a correct bit (i.e., a bit in $\mathbf{J}_0$) has an upc value of μ during the first iteration, and that a total of $\mu_{\mathtt{s}}$ (resp. $\mu_{\mathtt{u}}$) of parity checks involving such bit become (resp. stay) unsatisfied after the first iteration. $\lambda(\mu, \mu_{\mathtt{s}}, \mu_{\mathtt{u}})$ is the sum of all the disjoint events that lead to the upc being greater than or equal to $\mathtt{th}^{(3)}$. Given μ, $\mu_{\mathtt{s}}$, and $\mu_{\mathtt{u}}$, $\lambda(\mu, \mu_{\mathtt{s}}, \mu_{\mathtt{u}})$ is thus equal to the probability of flipping the bit in $\mathbf{J}_{000}$. The probability $p_{\mathtt{flip}|000}$ can be derived by averaging $\lambda(\mu, \mu_{\mathtt{s}}, \mu_{\mathtt{u}})$ across all the possible values $\mu, \mu_{\mathtt{s}}, \mu_{\mathtt{u}}$. Since all the bits in $\mathbf{J}_{000}$ are not flipped in the first and second iteration, the possible values for $\mu, \mu_{\mathtt{s}}, \mu_{\mathtt{u}}$ are $\mu < \mathtt{th}^{(1)}$ and $\mu_{\mathtt{s}} + \mu_{\mathtt{u}} < \mathtt{th}^{(2)}$. □

The flip probabilities $p_{\mathtt{flip}|abc}$ (with $a, b, c \in \{0, 1\}$) can be derived with a series of steps which is equivalent to the one of $p_{\mathtt{flip}|000}$ up to a matter of indexes. We can now provide the explicit formulation for the decoding failure probability after three iterations of the parallel bit flipping decoder.

Theorem 1 (Hard-to-Decode Errors). *The decoding failure rate of a three-iterations parallel decoder induced by hard-to-decode errors ($\mathtt{DFR}_{\mathtt{htd}}$) is:*

$$\mathtt{DFR}_{\mathtt{htd}} = \sum_{\substack{y,\epsilon_{01},\epsilon_{11},\\ \epsilon_{001},\epsilon_{011},\\ \epsilon_{101},\epsilon_{111}}} \left(\Pr(\mathcal{W} = y) \cdot \left(\prod_{a\in\{0,1\}} \Pr(\mathcal{E}_{a1} = \epsilon_{a1}) \right) \left(\prod_{a,b\in\{0,1\}} \Pr(\mathcal{E}_{ab1} = \epsilon_{ab1}) \right) \cdot \mathtt{DFR}_{\epsilon} \right)$$

with $\mathtt{DFR}_{\epsilon} = 1 - \left(\prod_{a,b\in\{0,1\}} (1 - p_{\mathtt{flip}|ab0})^{(\epsilon_{ab0})} \right) \left(\prod_{a,b\in\{0,1\}} (p_{\mathtt{flip}|ab1})^{(\epsilon_{ab1})} \right)$,

where $\epsilon_{000}{=}n{-}t{-}\epsilon_{01}{-}\epsilon_{001}$, $\epsilon_{010}{=}\epsilon_{01}{-}\epsilon_{011}$, $\epsilon_{100}{=}t{-}\epsilon_{11}{-}\epsilon_{101}$, *and* $\epsilon_{110}{=}\epsilon_{11}{-}\epsilon_{111}$.

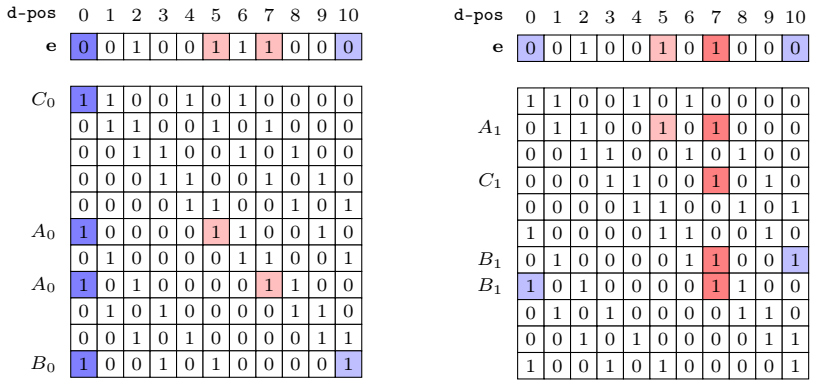
Proof. Under Assumption 2, the probability $\mathtt{DFR}_{\epsilon}$ of performing at least an incorrect flip during the third iteration equals 1 minus the probability of independently maintaining all the bits that are correct after the second iteration (i.e., d-positions in $\mathbf{J}_{ab0}$, $a, b \in \{0, 1\}$) and flipping all the discrepant bits (i.e., d-positions in $\mathbf{J}_{ab1}$, $a, b \in \{0, 1\}$). The overall failure rate $\mathtt{DFR}_{\mathtt{htd}}$ can be calculated by averaging $\mathtt{DFR}_{\epsilon}$ over all the admissible values of the initial syndrome weight and the number of discrepancies after the first and second iteration [4]. □

3.3 Near Codewords

In this section, we characterize the probability $\mathtt{DFR}_{\mathtt{ncw}}$ that the decoding process converges towards a near codeword, causing a decoding failure. By *convergence*, we denote the event where the discrepancy vector $\mathrm{Supp}(\bar{\mathbf{d}}^{(3)})$ obtained after three decoding iterations has a high overlap with a near codeword, caused by a large number of errors in $\mathbf{e} = \mathrm{Supp}(\bar{\mathbf{d}}^{(0)})$ appearing in the support of said near codeword at the beginning of the decoding process. To compute $\mathtt{DFR}_{\mathtt{ncw}}$, we derive the probability $p_{\mathtt{ncw}}$ of converging towards a *specific* near codeword $\boldsymbol{\nu}$.

We define $\mathcal{M}^{(\mathtt{iter})}$ to be the random variable counting the number of elements in the intersection between $\mathrm{Supp}(\bar{\mathbf{d}}^{(\mathtt{iter})})$ and the support $\mathrm{Supp}(\boldsymbol{\nu})$. of the said, specific, near codeword. Since the positions of the t asserted bits in $\bar{\mathbf{d}}^{(0)}$ (i.e., which match the ones in $\mathbf{e}$) are uniformly distributed across the pn_0 bits, and since $\mathrm{Supp}(\boldsymbol{\nu})$ has size v, we have that $\mathcal{M}^{(0)}$ follows a hypergeometric distribution, $\Pr(\mathcal{M}^{(0)}{=}m^{(0)}) = \binom{v}{m^{(0)}}\binom{pn_0-v}{t-m^{(0)}}/\binom{pn_0}{t}$. We denote the amount of common elements between $\mathrm{Supp}(\mathbf{e})$ and $\mathrm{Supp}(\boldsymbol{\nu})$ as $m^{(0)}$, and study the properties of the parity checks that include the $v - m^{(0)}$ null bits of $\mathbf{e}$ and the $m^{(0)}$ asserted bits within $\boldsymbol{\nu}$. We start by recalling the following two facts, proven in [8].

Proposition 6. *Consider the v parity checks where one of the $v - m^{(0)}$ correct bits in the near codeword is involved. Then: i) $m^{(0)}$ of these parity checks also include one **erroneous** bit within the near codeword (denote the checks as type A_0); ii) $v - m^{(0)} - 1$ of these parity checks also include one **correct** bit within the near codeword (denote the checks as type B_0); iii) 1 of these parity checks does not (necessarily) include any other bits within the near codeword (type C_0).*



(a) Correct bit (d-position 0) (b) Incorrect bit (d-position 7)

Fig. 4: Visualization of the near codeword $\boldsymbol{\nu} = \{0, 5, 7, 10\}$ in one block of a QC parity check matrix having $p = 11$, $v = 4$, where $\mathcal{M}^{(0)} = 2$, along with the parity check types of a correct (a) and incorrect (b) bit within the near codeword.

Proposition 7. *Consider the v parity check where one of the $m^{(0)}$ erroneous bits in the near codeword is involved. Then: i) $m^{(0)}-1$ of these parity checks also include one **erroneous** bit within the near codeword (type A_1); ii) $v - m^{(0)}$ of these parity checks also include one **correct** bit within the near codeword (type B_1); iii) 1 of these parity checks does not (necessarily) include any other bits within the near codeword (type C_1).*

Fig. 4 provides a visual representation of Proposition 6 and Proposition 7. In the example, the set of d-positions $\boldsymbol{\nu} = \{0, 5, 7, 10\}$ is, by Definition 5, a near codeword, since it matches the support of a column of the parity check matrix (specifically, column in d-position 0). Within the near codeword $\boldsymbol{\nu}$, where $\mathcal{M}^{(0)} = m^{(0)} = 2$, each correct bit (e.g., the bit in position 0) is included in $m^{(0)}$=2 parity checks of type A_0, $v-m^{(0)}-1$=1 of type B_0, and one of type C_0, while each incorrect bit (e.g., the bit in position 4) is included in $m^{(0)}-1$=1 parity checks of type A_1, $v-m^{(0)}$=2 of type B_1, and one of type C_1.

Proposition 8. *Consider a parity check of the type A_0 (i.e., a check including one correct bit and one incorrect bit within the near codeword).*
The probability p_{A_0} that such parity check is unsatisfied is:

$$p_{A_0} = \frac{\sum_{f=1,\ \mathtt{odd}}^{\min(t,w)} \Pr(\mathcal{F}_1 = f) \cdot \frac{\binom{w-f}{1}\binom{f}{1}}{\binom{w}{2}}}{\sum_{f=0}^{\min(t,w)} \Pr(\mathcal{F}_1 = f) \cdot \frac{\binom{w-f}{1}\binom{f}{1}}{\binom{w}{2}}} = \frac{\sum_{f=1,\ \mathtt{odd}}^{\min(t,w)} \Pr(\mathcal{F}_1 = f) \cdot (w-f)f}{\sum_{f=0}^{\min(t,w)} \Pr(\mathcal{F}_1 = f) \cdot (w-f)f}$$

Analogously:

$$p_{B_0} = \frac{\sum_{f=1,\ \mathtt{odd}}^{\min(t,w)} \Pr(\mathcal{F}_1 = f) \cdot (w-f)(w-f-1)}{\sum_{f=0}^{\min(t,w)} \Pr(\mathcal{F}_1 = f) \cdot (w-f)(w-f-1)}, \quad p_{C_0} = \frac{\sum_{f=1,\ \mathtt{odd}}^{\min(t,w)} \Pr(\mathcal{F}_1 = f) \cdot (w-f)}{\sum_{f=0}^{\min(t,w)} \Pr(\mathcal{F}_1 = f) \cdot (w-f)}$$

The same calculations can be performed for parity checks including an incorrect bit. We have that $p_{B_1} = p_{A_0}$, since both probabilities assume one correct and one incorrect bit to be included in the parity check, and:

$$p_{A_1} = \frac{\sum_{f=1,\ \mathtt{odd}}^{\min(t,w)} \Pr(\mathcal{F}_1 = f) \cdot (f-1)f}{\sum_{f=0}^{\min(t,w)} \Pr(\mathcal{F}_1 = f) \cdot (f-1)f}, \quad p_{C_1} = \frac{\sum_{f=1,\ \mathtt{odd}}^{\min(t,w)} \Pr(\mathcal{F}_1 = f) \cdot f}{\sum_{f=0}^{\min(t,w)} \Pr(\mathcal{F}_1 = f) \cdot f}$$

Knowing the unsatisfaction probability for each type of parity check, we can compute the probability $p_{\mathtt{flip}|0}$ of flipping a correct bit within the support of a near codeword. Correct bits appear in $m^{(0)}$ parity checks with unsatisfaction probability p_{A_0}, $v - m^{(0)} - 1$ parity checks with unsatisfaction probability p_{B_0} and 1 parity check with unsatisfaction probability p_{C_0}, therefore the probability that their $\mathtt{upc}$ is greater than the flipping threshold $\mathtt{th}^{(1)}$ is $p_{\mathtt{flip}|0} =$

$$\sum_{\substack{\mu_{A_0},\ \mu_{B_0},\ \mu_{C_0} \\ \mu_{A_0}+\mu_{B_0}+\mu_{C_0} \geq \mathtt{th}^{(1)}}} \mathrm{BIN}(m^{(0)}, p_{A_0}, \mu_{A_0}) \mathrm{BIN}(v - m^{(0)} - 1, p_{B_0}, \mu_{B_0}) \mathrm{BIN}(1, p_{C_0}, \mu_{C_0})$$

Following the same line of reasoning, we have that the probability $p_{\neg\mathtt{flip}|1}$ of maintaining an incorrect bit within the near codeword is $p_{\neg\mathtt{flip}|1} =$

$$\sum_{\substack{\mu_{A_1},\ \mu_{B_1},\ \mu_{C_1} \\ \mu_{A_1}+\mu_{B_1}+\mu_{C_1} < \mathtt{th}^{(1)}}} \mathrm{BIN}(m^{(0)} - 1, p_{A_1}, \mu_{A_1}) \mathrm{BIN}(v - m^{(0)}, p_{B_1}, \mu_{B_1}) \mathrm{BIN}(1, p_{C_1}, \mu_{C_1})$$

We now compute the probability that a parity check of the type A_0 is unsatisfied, given that the correct bit included in the check has been flipped ($p_{A_0|\mathtt{flip}}$) or not ($p_{A_0|\neg\mathtt{flip}}$). We provide the formulation for $p_{A_0|\mathtt{flip}}$: the same line of thought can be employed to derive $p_{A_0|\neg\mathtt{flip}}$, and the same probabilities for other parity checks of type B_0, A_1 and B_1.

Proposition 9. $p_{A_0|\mathtt{flip}} =$

$$\frac{\sum_{\substack{\mu_{A_0},\ \mu_{B_0},\ \mu_{C_0} \\ \mu_{A_0}+\mu_{B_0}+\mu_{C_0} \geq \mathtt{th}^{(1)}-1}} (p_{A_0} \cdot \alpha(\mu_{A_0},\ \mu_{B_0},\ \mu_{C_0}))}{\sum_{\substack{\mu_{A_0},\ \mu_{B_0},\ \mu_{C_0} \\ \mu_{A_0}+\mu_{B_0}+\mu_{C_0} \geq \mathtt{th}^{(1)}-1}} (p_{A_0} \cdot \alpha(\mu_{A_0},\ \mu_{B_0},\ \mu_{C_0})) + \sum_{\substack{\mu_{A_0},\ \mu_{B_0},\ \mu_{C_0} \\ \mu_{A_0}+\mu_{B_0}+\mu_{C_0} \geq \mathtt{th}^{(1)}}} ((1-p_{A_0}) \cdot \alpha(\mu_{A_0},\ \mu_{B_0},\ \mu_{C_0}))}$$

where:

$$\alpha(\mu_{A_0}, \mu_{B_0}, \mu_{C_0}) = \mathrm{BIN}(m^{(0)} - 1, p_{A_0}, \mu_{A_0}) \mathrm{BIN}(v - m^{(0)} - 1, p_{B_0}, \mu_{B_0}) \mathrm{BIN}(1, p_{C_0}, \mu_{C_0})$$

Proof. $\alpha(\mu_{A_0}, \mu_{B_0}, \mu_{C_0})$ is the probability that, among the parity check equations different from the one under analysis, μ_{A_0} of the type A_0, μ_{B_0} of the type B_0, and μ_{C_0} of the type C_0 are unsatisfied. Assuming that the parity check under analysis is *satisfied*, we have that $\mu_{A_0} + \mu_{B_0} + \mu_{C_0}$ must be greater than $\mathtt{th}^{(1)}$ for the bit to be (incorrectly) flipped. If the parity check under analysis is *unsatisfied*, $\mu_{A_0} + \mu_{B_0} + \mu_{C_0}$ must be greater than $\mathtt{th}^{(1)} - 1$. $p_{A_0|\mathtt{flip}}$ is thus the probability sum of all the events that lead to the parity check being unsatisfied *and* the correct bit being flipped, normalized over the probability of all the events that lead to the correct bit being flipped. □

Since the bits within the near codeword share pairwise dependencies, we cannot model the flipping decisions in the near codeword as independent. To account for this fact, we consider the flipping action applied to each bit of the near codeword, one at a time, and model the probability of flipping or maintaining the bit observed at each step.

Definition 6. *The r.v. $\mathcal{K}_{01,i}$ models the number of correct bits that have been (erroneously) flipped, after i correct bits have been observed, $0 \leq i \leq v - m^{(0)}$. The r.v. $\mathcal{K}_{11,j}$ models the number of incorrect bits that have been (erroneously) maintained, after j incorrect bits have been observed, $0 \leq j \leq m^{(0)}$.*

Proposition 10. *Let $\mathcal{K}_{01,i} = k_{01,i}$ and $\mathcal{K}_{11,j} = k_{11,j}$, and consider the event of observing the $(i+1)$-th correct bit within the near codeword. Then:*

$$\Pr(\mathcal{K}_{01,i+1} = k_{01,i} + 1) = \Pr(\mathcal{U}_{A_0} + \mathcal{U}_{B_0} + \mathcal{U}_{C_0} + \mathcal{U}_{A_0|\mathtt{f}} + \mathcal{U}_{B_0|\mathtt{f}} + \mathcal{U}_{A_0|\neg\mathtt{f}} + \mathcal{U}_{B_0|\neg\mathtt{f}} \geq \mathtt{th}^{(1)})$$

and $\Pr(\mathcal{K}_{01,i+1}{=}k_{01,i}) = 1 - \Pr(\mathcal{K}_{01,i+1}{=}k_{01,i} + 1)$,

$$\begin{cases} \mathcal{U}_{A_0} \sim \textsc{Bin}(m^{(0)} - j, p_{A_0}) \\ \mathcal{U}_{B_0} \sim \textsc{Bin}(v - m^{(0)} - 1 - i, p_{B_0}) \\ \mathcal{U}_{C_0} \sim \textsc{Bin}(1, p_{C_0}) \end{cases} \qquad \begin{cases} \mathcal{U}_{A_0|\mathtt{f}} \sim \textsc{Bin}(j - k_{11,j}, p_{B_1|\mathtt{flip}}) \\ \mathcal{U}_{B_0|\mathtt{f}} \sim \textsc{Bin}(k_{01,i}, p_{B_0|\mathtt{flip}}) \\ \mathcal{U}_{A_0|\neg\mathtt{f}} \sim \textsc{Bin}(k_{11,j}, p_{B_1|\neg\mathtt{flip}}) \\ \mathcal{U}_{B_0|\neg\mathtt{f}} \sim \textsc{Bin}(i - k_{01,i}, p_{B_0|\neg\mathtt{flip}}) \end{cases}$$

Proof. The event $\mathcal{K}_{01,i} = k_{01,i}$ implies that, when observing the $(i+1)$-th correct bit (out of $v - m^{(0)}$), $k_{01,i}$ correct bits have been observed and incorrectly flipped, $i - k_{01,i}$ have been observed and correctly maintained, and $v - m^{(0)} - i - 1$ have yet to be observed. The parity checks shared between the current correct bit and one of the $k_{01,i}$ incorrectly flipped bits have probability $p_{B_0|\mathtt{flip}}$ of being unsatisfied. The parity checks shared between the current correct bit and one of the $i - k_{01,i}$ correctly maintained bits have probability $p_{B_0|\neg\mathtt{flip}}$ of being unsatisfied. The other $v - m^{(0)} - 1 - i$ parity checks of the type B_0 have probability p_{B_0} of being unsatisfied. Applying the same reasoning to parity checks of the type A_0, we can characterize the behavior of all the v parity checks involving the currently observed correct bit. The probability of flipping such bit (causing $\mathcal{K}_{01,i+1} = k_{01,i} + 1$) is thus equal to the probability that the sum of all the unsatisfied checks is greater than or equal to $\mathtt{th}^{(1)}$. □

The probabilities $\Pr(\mathcal{K}_{11,j+1} = k_{11,j} + 1)$ and $\Pr(\mathcal{K}_{11,j+1} = k_{11,j})$, modeling the behaviour of an observed incorrect bit, can be derived in an analogous way. Starting from the trivial distribution of $\mathcal{K}_{01,0}$ and $\mathcal{K}_{11,0}$ (both equal to 0 with probability 1), we use Proposition 10 recursively to compute the distribution of $\mathcal{K}_{01,v-m^{(0)}}$ and $\mathcal{K}_{11,m^{(0)}}$. Consequently, we can derive the distribution of $\mathcal{M}^{(1)}$, the number of discrepancies in the near codeword at the end of the first iteration.

Proposition 11. $\Pr(\mathcal{M}^{(1)} = m^{(1)}) =$

$$= \sum_{y, m^{(0)}, x} \Pr(\mathcal{W} = y) \frac{\binom{v}{m^{(0)}} \binom{pn_0 - v}{t - m^{(0)}}}{\binom{pn_0}{t}} \Pr(\mathcal{K}_{01,v-m^{(0)}} + \mathcal{K}_{11,m^{(0)}} = m^{(1)})$$

We now provide an analysis for the iterations following the first. We indicate the current iteration index as $\mathtt{i}$ (starting from $\mathtt{i}=1$, since the first iteration has already been studied), and analyse how the overlap between the near codeword and the discrepancy vector evolves during iteration $\mathtt{i}+1$. Since our goal is to predict the error floor caused by near codewords, we are interested in a probabilistic model that is particularly precise for the parameter sets where the decoding failures caused by generic *hard-to-decode* errors are negligible with respect to failures caused by near codewords. Under this assumption, we can safely neglect the influence on the convergence probability of discrepancies that lie outside the support of a near codeword after the first decoding iteration. This observation will allow us to avoid making the simplifying assumption that failures are caused by a single near codeword (employed in, e.g., [8]).

Proposition 12. *Let the number of discrepancies left in the near codeword after* $\mathtt{i}$ *iterations be equal to* $m^{(\mathtt{i})}$*. Then:*

$$p_{A_0} = \sum_{f=0,\mathtt{even}}^{m^{(\mathtt{i})}-1} \frac{\binom{v-2}{f}\binom{p-v}{m^{(\mathtt{i})}-1-f}}{\binom{p-2}{m^{(\mathtt{i})}-1}}, \quad p_{B_0} = \sum_{f=1,\mathtt{odd}}^{m^{(\mathtt{i})}} \frac{\binom{v-2}{f}\binom{p-v}{m^{(\mathtt{i})}-f}}{\binom{p-2}{m^{(\mathtt{i})}}}, \quad p_{C_0} = \sum_{f=1,\mathtt{odd}}^{m^{(\mathtt{i})}} \frac{\binom{v-1}{f}\binom{p-v}{m^{(\mathtt{i})}-f}}{\binom{p-1}{m^{(\mathtt{i})}}}$$

$$p_{A_1} = \sum_{f=1,\mathtt{odd}}^{m^{(\mathtt{i})}-2} \frac{\binom{v-2}{f}\binom{p-v}{m^{(\mathtt{i})}-2-f}}{\binom{p-2}{m^{(\mathtt{i})}-2}}, \quad p_{B_1} = \sum_{f=0,\mathtt{even}}^{m^{(\mathtt{i})}-1} \frac{\binom{v-2}{f}\binom{p-v}{m^{(\mathtt{i})}-1-f}}{\binom{p-2}{m^{(\mathtt{i})}-1}}, \quad p_{C_1} = \sum_{f=0,\mathtt{even}}^{m^{(\mathtt{i})}-1} \frac{\binom{v-1}{f}\binom{p-v}{m^{(\mathtt{i})}-1-f}}{\binom{p-1}{m^{(\mathtt{i})}-1}}$$

Proof. The probability of a parity check being unsatisfied is equal to the probability sum of the disjoint events that lead to the number of erroneous bits (whose total is $m^{(\mathtt{i})}$) included in the parity check to be odd. Following Proposition 6, parity checks of type A_0 include one correct bit and one incorrect bit, therefore such parity equation is unsatisfied iff an *even* number of discrepancies, among the $m^{(\mathtt{i})}-1$ remaining ones, are also involved in the check. The same reasoning can be applied to the other parity check types, starting from Proposition 6 and Proposition 7. □

Applying the probabilities derived in Proposition 12 to Proposition 9, Proposition 10 and Proposition 11, we can compute the p.m.f. of $\mathcal{M}^{(\mathtt{i}+1)}$, starting from the one of $\mathcal{M}^{(\mathtt{i})}$. This allows us to derive the distribution of $\mathcal{M}^{(\mathtt{i})}$ up to $\mathcal{M}^{(3)}$. The probability that discrepancies remaining within the near codeword cause a decoding failure after three iterations, $p_{\mathtt{ncw}}$, can be computed as $p_{\mathtt{ncw}} = \Pr(\mathcal{M}^{(3)} \geq 1)$.

This formulation of $p_{\mathtt{ncw}}$ allows us to count the decoding failures caused by a "partial" convergence towards a near codeword, including the cases where the final discrepancy pattern does not match the support of the near codeword completely. We now compute an upper bound on the probability $\mathtt{DFR}_{\mathtt{ncw}}$ of converging to *any* near codeword, without assumptions on the number of near codewords leading to a decoding failure.

Theorem 2 (Near Codewords). $\mathtt{DFR}_{\mathtt{ncw}} \leq pn_0 \cdot p_{\mathtt{ncw}}$

3.4 Half Codewords

QC-LDPC and QC-MDPC codes exhibit codewords of Hamming weight $2v$. Indeed, consider a QC code with a parity check matrix $\mathbf{H}$ built by tiling n_0 circulant blocks, and let $\mathbf{e} = [\mathbf{e}_{(0)} \mid \mathbf{e}_{(1)} \mid \ldots \mid \mathbf{e}_{(n_0-1)}]$, where each $\mathbf{e}_{(j)}$ is a vector of size p. Consider two blocks $0 \leq i_0 < i_1 \leq n_0 - 1$, a position between said blocks $0 \leq l \leq p-1$, and $\mathbf{e}$ such that $\mathbf{e}_{(i_0)} = (\mathbf{H}_{:,pi_1+l})^{\mathtt{T}}$, $\mathbf{e}_{(i_1)} = (\mathbf{H}_{:,pi_0+l})^{\mathtt{T}}$, and $\mathbf{e}_{(j)} = \mathbf{0}$ for $j \neq i_0, i_1$. We have that $\mathtt{wt}(\mathbf{e}) = 2v$ and $\mathbf{s} = \mathbf{H}\mathbf{e}^{\mathtt{T}} = \mathbf{0}$, meaning that $\mathtt{wt}(\mathbf{e})$ is a codeword. The probability of a decoding failure caused by the convergence to a codeword different from the zero vector has been shown to be negligible [12]. We now introduce an error structure which can cause decoding failures with higher probability than a whole codeword.

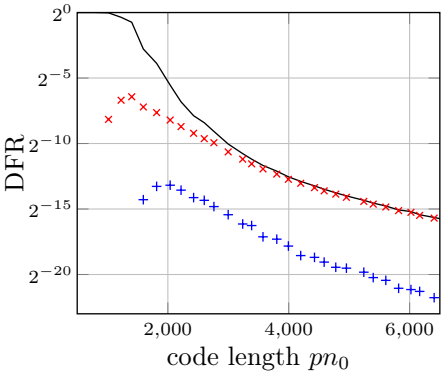


Fig. 5: Decoding failure probability for QC-LDPC codes with $n_0 = 2$, $v = 7$, $t = 18$, parallel bit flipping decoder with $\mathtt{th}^{(1)} = \mathtt{th}^{(2)} = \mathtt{th}^{(3)} = \lceil \frac{v+1}{2} \rceil$. In the plot, —— denotes the total DFR, + denotes convergence to near codewords, × denotes convergence to half codewords.

Definition 7 (Half codeword). *An error vector* $\mathbf{e}$ *is a half codeword if it is null, save for a single* $\mathbf{e}_{(i_0)}$ *for which:* $\mathbf{e}_{(i_0)} = (\mathbf{H}_{:,pi_1+l})^{\mathtt{T}}$, $i_1 \neq i_0$, $0 \leq l \leq p-1$

Half codewords can be thought as a generalization of near codewords, since decoding converges to a near codeword whenever one block $\mathbf{e}_{(i_0)}$ of the error vector has a high overlap with one column of $\mathbf{H}$ in the *same* block i_0, while decoding converges to a half codewords whenever one block $\mathbf{e}_{(i_0)}$ of the error vector has a high overlap with one column of $\mathbf{H}$ in a *different* block $i_1 \neq i_0$. Indeed, if we consider two half codewords $\widehat{\mathbf{e}}$ and $\widetilde{\mathbf{e}}$ such that $\widehat{\mathbf{e}} \oplus \widetilde{\mathbf{e}}$ is a codeword (e.g., where l is the same and i_0 and i_1 are swapped), we have that $\mathbf{s}{=}\mathbf{H}\widehat{\mathbf{e}}^T{=}\mathbf{H}\widetilde{\mathbf{e}}^T$, meaning that the decoder cannot distinguish which of the two error vectors generated the given syndrome. If the error $\mathbf{e}$ is not exactly equal to either $\widehat{\mathbf{e}}$ or $\widetilde{\mathbf{e}}$, but has a high number of asserted bits in common with one of the two vectors, then a sequential decoder such as the step-by-step decoder [8] can possibly detect the erroneous positions by only flipping the bits in the block containing the errors. On the other hand, a parallel decoder will flip the "suspect" positions in both blocks i_0 and i_1 simultaneously, converging towards a cycle of period 2 between $\bar{\mathbf{d}} = \widehat{\mathbf{e}}$ and $\bar{\mathbf{d}} = \widetilde{\mathbf{e}}$ and causing a failure. This phenomenon cannot occur in sequential decoders, since it is inherently caused by *simultaneous* flips. Fig. 5 shows the DFR of QC-LDPC codes, distinguishing failures caused by convergence to near and half codewords. Under such specific decoding strategy, the waterfall region is dominated by hard-to-decode errors (as expected), but the error floor is dominated by half codewords. This example shows that taking into account

the parallel behaviour of the analyzed decoder is necessary to characterize the error floor correctly.

In this section, we compute the probability that the decoding process converges towards a half codeword, causing a failure ($\mathtt{DFR}_{\mathtt{hcw}}$). To this end, we derive the probability $p_{\mathtt{hcw}}$ of converging towards a *specific* couple of half codewords $(\boldsymbol{\nu}, \boldsymbol{\eta})$. We define $\mathcal{M}_0^{(\mathtt{iter})}$ and $\mathcal{M}_1^{(\mathtt{iter})}$ to be the random variables corresponding to the asserted bits in $\bar{\mathbf{d}}^{(\mathtt{iter})}$ (i.e., the incorrect bits after `iter` iterations) that lie in the support of the two halves of said codeword, distinguishing the number of incorrect bits in $\boldsymbol{\nu}$ ($\mathcal{M}_0^{(\mathtt{iter})}$) and $\boldsymbol{\eta}$ ($\mathcal{M}_1^{(\mathtt{iter})}$). Since the t positions of the discrepancies in $\bar{\mathbf{d}}^{(0)}$ (i.e., the asserted bits in $\mathbf{e}$) are uniformly distributed across the pn_0 bits, and since the support of each codeword half has size v, we have that the joint r.v. $(\mathcal{M}_0^{(0)}, \mathcal{M}_1^{(0)})$ follows a multivariate hypergeometric distribution, $\Pr(\mathcal{M}_0^{(0)} = m_0^{(0)}, \mathcal{M}_1^{(0)} = m_1^{(0)}) = \binom{v}{m_0^{(0)}}\binom{v}{m_1^{(0)}}\binom{pn_0-2v}{t-m_0^{(0)}-m_1^{(0)}} / \binom{pn_0}{t}$. We now fix the intersection between $\mathbf{e}$ and the support of the two codeword halves to $(m_0^{(0)}, m_1^{(0)})$, and study the properties of the parity checks that include the $v - m_0^{(0)}$ correct bits and $m_0^{(0)}$ incorrect bits within the **first** half codeword.

Proposition 13. *Consider the v parity check where one of the bits in the **first** half of the codeword (either correct or incorrect) is involved. Then: i) $m_1^{(0)}$ of these parity checks also include one **erroneous** bit from the other half of the codeword (type A_0 for correct bits, type A_1 for incorrect bits); ii) $v - m_1^{(0)}$ of these parity checks also include one **correct** bit from the other half of the codeword (type B_0 for correct bits, type B_1 for incorrect bits).*

Proof. Let $\mathbf{a}$ be the vector corresponding to the support of the first column of $\mathbf{H}$ (i.e., $\mathrm{Supp}(\mathbf{H}_{:,0}) = \{\mathbf{a}_i \mid 0 \le i \le v-1\}$), and let $\mathbf{z}$ be the vector corresponding to the support of the p-th column of $\mathbf{H}$ (i.e., $\mathrm{Supp}(\mathbf{H}_{:,p}) = \{\mathbf{z}_i \mid 0 \le i \le v-1\}$). Since $\mathbf{H}$ is built by circulant blocks of size p, we have that for $0 \le j \le p-1$ and $p \le \widetilde{j} \le 2p-1$:

$$\mathrm{Supp}(\mathbf{H}_{:,j}) = \{\mathbf{a}_i + j \bmod p \mid 0 \le i \le v-1\}$$

$$\mathrm{Supp}(\mathbf{H}_{:,\widetilde{j}}) = \{\mathbf{z}_i + \widetilde{j} \bmod p \mid 0 \le i \le v-1\}$$

Let j_0 be the position of a bit, such that that $0 \le j_0 \le p-1$ (i.e., the bit lies within the first circulant block of $\mathbf{H}$). Consider the following vectors of positions, located (without loss of generality) in the first and second circulant block, respectively:

$$\boldsymbol{\nu} = \{j_0, j_0 + \mathbf{z}_1 - \mathbf{z}_0 \bmod p, \ldots, j_0 + \mathbf{z}_k - \mathbf{z}_0 \bmod p, \ldots, j_0 + \mathbf{z}_{v-1} - \mathbf{z}_0 \bmod p\}$$

$$\boldsymbol{\eta} = \{(j_0 - \mathbf{z}_0 + \mathbf{a}_0 \bmod p) + p, \ldots (j_0 - \mathbf{z}_0 + \mathbf{a}_k \bmod p) + p, \ldots, (j_0 - \mathbf{z}_0 + \mathbf{a}_{v-1} \bmod p) + p\}$$

Letting $\widetilde{j_0} = j_0 - \mathbf{z}_0 \bmod p$, the values in $\boldsymbol{\nu}$ correspond to the support of $\mathbf{H}_{:,\widetilde{j_0}+p}$, and (after an appropriate reduction mod p) the values in $\boldsymbol{\eta}$ correspond to the support of $\mathbf{H}_{:,\widetilde{j_0}}$. As a consequence, $\boldsymbol{\nu}$ and $\boldsymbol{\eta}$ constitute the support of the two

halves of a codeword. Let $\mathbf{b}$ be the vector corresponding to the support of $\mathbf{H}_{:,j_0}$, with $\mathbf{b}_i = \mathbf{a}_i + j_0 \bmod p$. For every value $0 \leq k \leq v-1$, we have $\mathrm{Supp}(\mathbf{H}_{:,\boldsymbol{\eta}_k}) =$

$$= \{\mathbf{z}_i + \boldsymbol{\eta}_k \bmod p \mid 0 \leq i \leq v-1\} = \{\mathbf{a}_k + j_0 + (\mathbf{z}_i - \mathbf{z}_0) \bmod p \mid 0 \leq i \leq v-1\}$$

As a consequence, $\mathbf{b}_k \in \mathrm{Supp}(\mathbf{H}_{:,j_0})$ (by definition) and $\mathbf{b}_k = \mathbf{a}_k + j_0 \bmod p = \mathbf{a}_k + j_0 + (\mathbf{z}_0 - \mathbf{z}_0) \bmod p \in \mathrm{Supp}(\mathbf{H}_{:,\boldsymbol{\eta}_k})$, meaning that the parity check $\mathbf{b}_k$ includes both j_0 and $\boldsymbol{\eta}_k$. If $m_1^{(0)}$ of the positions in $\boldsymbol{\eta}$ are erroneous, then there are $m_1^{(0)}$ values of k for which $\mathbf{e}_{\boldsymbol{\eta}_k} = 1$, meaning that the bit in position j_0 shares $m_1^{(0)}$ parity checks with erroneous bits (types A_0 and A_1, depending on the value of $\mathbf{e}_{j_0}$). Moreover, $v - m_1^{(0)}$ parity checks are shared with correct bits (types B_0 and B_1, depending on the value of $\mathbf{e}_{j_0}$). □

Proposition 14. *Consider a parity check of the type A_0, i.e., a check including one correct bit from one of the two halves (say $\boldsymbol{\nu}$) and one incorrect bit from the other half (say $\boldsymbol{\eta}$) within the codeword. The probability p_{A_0} that such parity check is unsatisfied is:*

$$p_{A_0} = \frac{\sum_{f=1,\ \mathtt{odd}}^{\min(t,w)} \Pr(\mathcal{F}_1 = f) \cdot \frac{\binom{w-f}{1}\binom{f}{1}}{\binom{w}{2}}}{\sum_{f=0}^{\min(t,w)} \Pr(\mathcal{F}_1 = f) \cdot \frac{\binom{w-f}{1}\binom{f}{1}}{\binom{w}{2}}} = \frac{\sum_{f=1,\ \mathtt{odd}}^{\min(t,w)} \Pr(\mathcal{F}_1 = f) \cdot (w-f)f}{\sum_{f=0}^{\min(t,w)} \Pr(\mathcal{F}_1 = f) \cdot (w-f)f}$$

The same calculations can be performed for p_{B_0}, p_{A_1} and p_{B_1}. We have that $p_{B_1} = p_{A_0}$, since both probabilities assume one correct and one incorrect bit to be included in the parity check, and:

$$p_{B_0} = \frac{\sum_{f=1,\ \mathtt{odd}}^{\min(t,w)} \Pr(\mathcal{F}_1 = f) \cdot (w-f)(w-f-1)}{\sum_{f=0}^{\min(t,w)} \Pr(\mathcal{F}_1 = f) \cdot (w-f)(w-f-1)}, \quad p_{A_1} = \frac{\sum_{f=1,\ \mathtt{odd}}^{\min(t,w)} \Pr(\mathcal{F}_1 = f) \cdot (f-1)f}{\sum_{f=0}^{\min(t,w)} \Pr(\mathcal{F}_1 = f) \cdot (f-1)f}$$

Knowing the unsatisfaction probability for each type of parity check, we can compute the probability $p_{\mathtt{flip}|0}$ of flipping a correct bit within the **first** half of the codeword. Correct bits appear in $m_1^{(0)}$ parity checks with unsatisfaction probability p_{A_0} and $v - m_1^{(0)}$ parity checks with unsatisfaction probability p_{B_0}, therefore the probability that their $\mathtt{upc}$ is greater than the threshold $\mathtt{th}^{(1)}$ is

$$p_{\mathtt{flip}|0} = \sum_{\substack{\mu_{A_0},\ \mu_{B_0} \\ \mu_{A_0}+\mu_{B_0} \geq \mathtt{th}^{(1)}}} \mathrm{BIN}(m_1^{(0)}, p_{A_0}, \mu_{A_0}) \mathrm{BIN}(v - m_1^{(0)}, p_{B_0}, \mu_{B_0})$$

Following the same line of reasoning, we have that the probability $p_{\neg\mathtt{flip}|1}$ of maintaining an incorrect bit within the codeword half is

$$p_{\neg\mathtt{flip}|1} = \sum_{\substack{\mu_{A_1},\ \mu_{B_1} \\ \mu_{A_1}+\mu_{B_1} < \mathtt{th}^{(1)}}} \mathrm{BIN}(m_1^{(0)}, p_{A_1}, \mu_{A_1}) \mathrm{BIN}(v - m_1^{(0)}, p_{B_1}, \mu_{B_1})$$

We can now derive the distribution of the number of errors present in the **first** half of the codeword after the first iteration.

Proposition 15. $\Pr(\mathcal{M}_0^{(1)} = m_0^{(1)} \mid m_0^{(0)}, m_1^{(0)}) =$

$$= \textstyle\sum_x \textsc{Bin}(v - m_0^{(0)}, p_{\mathtt{flip}|0}, x) \cdot \textsc{Bin}(m_0^{(0)}, p_{\neg\mathtt{flip}|1}, m^{(1)} - x)$$

The distribution of the number of incorrect flipping decision within the **second** half of the codeword, $\Pr(\mathcal{M}_1^{(1)} = m_1^{(1)} \mid m_0^{(0)}, m_1^{(0)})$, can be derived by switching $m_0^{(1)}$ and $m_1^{(1)}$ in the formulations of $p_{\mathtt{flip}|0}$, $p_{\neg\mathtt{flip}|1}$, and Proposition 15. The joint p.m.f. of $(\mathcal{M}_0^{(1)}, \mathcal{M}_1^{(1)})$ can be derived using the law of total probability: $\Pr(\mathcal{M}_0^{(1)} = m_0^{(1)},\ \mathcal{M}_1^{(1)} = m_1^{(1)}) =$

$$\sum_{y, m_0^{(0)}, m_1^{(0)}} \Pr(\mathcal{W} = y) \cdot \frac{\binom{v}{m_0^{(0)}}\binom{v}{m_1^{(0)}}\binom{pn_0 - 2v}{t - m_0^{(0)} - m_1^{(0)}}}{\binom{pn_0}{t}} \cdot \Pr(\mathcal{M}_0^{(1)} = m_0^{(1)} \mid m_0^{(0)},\ m_1^{(0)}) \cdot \Pr(\mathcal{M}_1^{(1)} = m_1^{(1)} \mid m_0^{(0)},\ m_1^{(0)})$$

We now provide an analysis for the iterations following the first. Under the same assumptions described in the previous section, we can safely neglect the influence on the convergence probability of errors that lie outside the support of the codeword after the first decoding iteration.

Proposition 16. *Let the number of discrepancies left in the two halves of the codeword after* $\mathtt{i}$ *iteration be equal to* $(m_0^{(\mathtt{i})}, m_1^{(\mathtt{i})})$*. The parity check unsatisfaction probability for correct and incorrect bits in the* ***first*** *half is:*

$$p_{A_0} = \sum_{\substack{f_0,\ f_1 \\ f_0+f_1\ \mathtt{even}}} \frac{\binom{v-1}{f_0}\binom{p-v}{m_0^{(\mathtt{i})}-f_0}\binom{v-1}{f_1}\binom{p-v}{m_1^{(\mathtt{i})}-1-f_1}}{\binom{p-1}{m_0^{(\mathtt{i})}}\binom{p-1}{m_1^{(\mathtt{i})}-1}}, \quad p_{B_0} = \sum_{\substack{f_0,\ f_1 \\ f_0+f_1\ \mathtt{odd}}} \frac{\binom{v-1}{f_0}\binom{p-v}{m_0^{(\mathtt{i})}-f_0}\binom{v-1}{f_1}\binom{p-v}{m_1^{(\mathtt{i})}-f_1}}{\binom{p-1}{m_0^{(\mathtt{i})}}\binom{p-1}{m_1^{(\mathtt{i})}}}$$

$$p_{A_1} = \sum_{\substack{f_0,\ f_1 \\ f_0+f_1\ \mathtt{odd}}} \frac{\binom{v-1}{f_0}\binom{p-v}{m_0^{(\mathtt{i})}-1-f_0}\binom{v-1}{f_1}\binom{p-v}{m_1^{(\mathtt{i})}-1-f_1}}{\binom{p-1}{m_0^{(\mathtt{i})}-1}\binom{p-1}{m_1^{(\mathtt{i})}-1}}, \quad p_{B_1} = \sum_{\substack{f_0,\ f_1 \\ f_0+f_1\ \mathtt{even}}} \frac{\binom{v-1}{f_0}\binom{p-v}{m_0^{(\mathtt{i})}-1-f_0}\binom{v-1}{f_1}\binom{p-v}{m_1^{(\mathtt{i})}-f_1}}{\binom{p-1}{m_0^{(\mathtt{i})}-1}\binom{p-1}{m_1^{(\mathtt{i})}}}$$

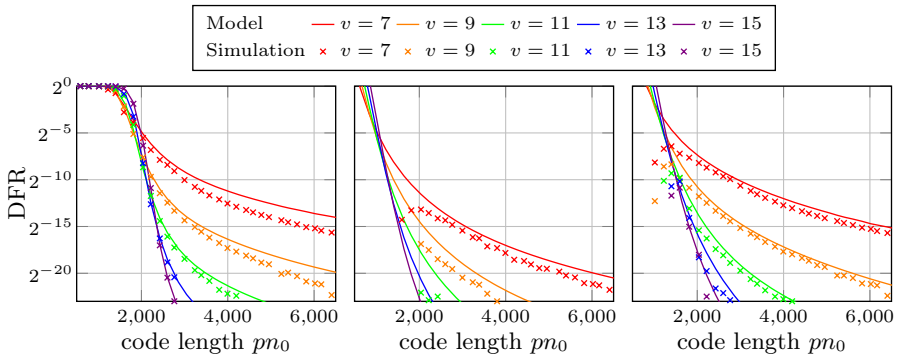
For correct and incorrect bits in the **second** half, it is sufficient to exchange f_0 with f_1 and $m_0^{(\mathtt{i})}$ with $m_1^{(\mathtt{i})}$. Moreover, the formulations of $p_{\mathtt{flip}|0}$ and $p_{\neg\mathtt{flip}|1}$ is analogous to the one derived during the study of the first iteration. Following the same line of reasoning as Proposition 15, we get:

$$\Pr(\mathcal{M}_0^{(\mathtt{i}+1)} = m_0^{(\mathtt{i}+1)} \mid m_0^{(\mathtt{i})},\ m_1^{(\mathtt{i})}) = \sum_x \textsc{Bin}(v - m_0^{(\mathtt{i})}, p_{\mathtt{flip}|0}, x) \cdot \textsc{Bin}(m_0^{(\mathtt{i})}, p_{\neg\mathtt{flip}|1}, m_0^{(\mathtt{i}+1)} - x)$$

The equivalent probability for the second half, $\Pr(\mathcal{M}_1^{(\mathtt{i}+1)} = m_1^{(\mathtt{i}+1)} \mid m_0^{(\mathtt{i})},\ m_1^{(\mathtt{i})})$, can be derived in the same way. Then, $\Pr(\mathcal{M}_0^{(\mathtt{i}+1)} = m_0^{(\mathtt{i}+1)},\ \mathcal{M}_1^{(\mathtt{i}+1)} = m_1^{(\mathtt{i}+1)}) =$

$$\sum_{m_0^{(\mathtt{i})}, m_1^{(\mathtt{i})}} \Pr(\mathcal{M}_0^{(\mathtt{i})} = m_0^{(\mathtt{i})},\ \mathcal{M}_1^{(\mathtt{i})} = m_1^{(\mathtt{i})}) \cdot \Pr(\mathcal{M}_0^{(\mathtt{i}+1)} = m_0^{(\mathtt{i}+1)} \mid m_0^{(\mathtt{i})},\ m_1^{(\mathtt{i})}) \cdot \Pr(\mathcal{M}_1^{(\mathtt{i}+1)} = m_1^{(\mathtt{i}+1)} \mid m_0^{(\mathtt{i})},\ m_1^{(\mathtt{i})})$$

Once the joint p.m.f. of $(\mathcal{M}_0^{(\mathtt{i})}, \mathcal{M}_1^{(\mathtt{i})})$ up to $(\mathcal{M}_0^{(3)}, \mathcal{M}_1^{(3)})$ has been derived, we can compute the probability $p_{\mathtt{hcw}}$ of a decoding failure caused by the half codewords $(\boldsymbol{\nu}, \boldsymbol{\eta})$ as $p_{\mathtt{hcw}} = \Pr(\mathcal{M}_0^{(3)} + \mathcal{M}_1^{(3)} \geq 1)$. With $p_{\mathtt{hcw}}$, we can derive an estimate on the probability $\mathtt{DFR}_{\mathtt{hcw}}$ of the decoder converging to *any* half codeword, without any assumption on the number of half codewords causing a decoding failure.



(a) Decoding failure rate (b) Near codewords (c) Half codewords

Fig. 6: Comparison between our model and numerical simulations for three iterations $\mathtt{DFR}_{\mathtt{QC}}$, $\mathtt{DFR}_{\mathtt{ncw}}$ and $\mathtt{DFR}_{\mathtt{hcw}}$ values for QC-LDPCs. Parameters: $n_0 = 2$, $v \in \{7, 9, 11, 13, 15\}$, $t = 18$, parallel decoder with $\mathtt{th}^{(1)} = \mathtt{th}^{(2)} = \mathtt{th}^{(3)} = \lceil \frac{v+1}{2} \rceil$. Fig. (a): DFR of QC-LDPC codes, Fig. (b): probability of convergence to a near codeword, Fig. (c): probability of convergence to a half codeword.

Theorem 3 (Half Codewords). $\mathtt{DFR}_{\mathtt{hcw}} \leq \binom{n_0}{2} p \cdot p_{\mathtt{hcw}}$

With the use of Theorem 1, Theorem 2 and Theorem 3, under Assumption 3 we can derive a conservative closed form estimation for $\mathtt{DFR}_{\mathtt{QC}}$, the DFR of the three-iterations parallel bit flipping decoder applied to QC-MDPC codes.

Theorem 4 (Decoding Failure Rate for QC-MDPC codes).

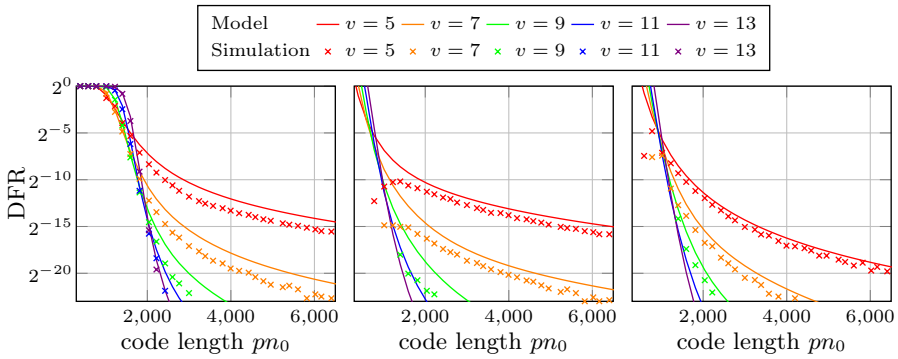
$$\mathtt{DFR}_{\mathtt{QC}} \leq \mathtt{DFR}_{\mathtt{htd}} + \mathtt{DFR}_{\mathtt{ncw}} + \mathtt{DFR}_{\mathtt{hcw}}$$

4 Experimental Validation

We now provide a numerical validation of our closed-form DFR estimation technique. [1] Fig. 6(a) reports the results of our closed-form model compared against Monte-Carlo simulations of DFR obtained decoding 10^8 random error vectors with fixed weight. We analyzed QC-LDPC codes with rate $\frac{1}{2}$ and 5 different densities $\frac{v}{n}$, where the random sampling of codes has been shaped so that the empirical variance of the distribution of discrepancies after one iteration matches the one of regular LDPC codes, property that is verified to hold for QC-MDPC codes of interest in cryptographic applications. Our model provides a precise or slightly conservative (i.e., higher) DFR estimate both in the *waterfall* and *floor* regimes.

Employing QC-LDPC codes allows us to distinguish failures caused by near codewords and half codewords, since whenever a failure is caused by near/half

[1] Our implementation of the DFR model is available at https://crypto.deib.polimi.it/model_DFR_3it.zip



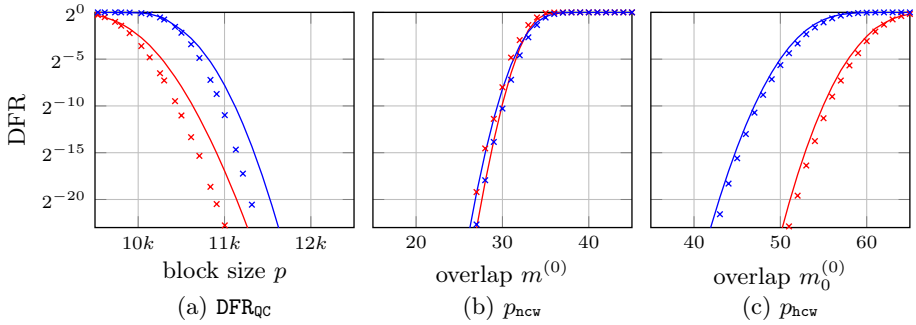
(a) Decoding failure rate (b) Near codewords (c) Half codewords

Fig. 7: Comparison between our model and numerical simulations for three iterations $\mathtt{DFR}_{\mathtt{QC}}$, $\mathtt{DFR}_{\mathtt{ncw}}$ and $\mathtt{DFR}_{\mathtt{hcw}}$ values for QC-LDPCs. Parameters: $n_0 = 4$, $v \in \{5, 7, 9, 11, 13\}$, $t = 18$, parallel decoder with $\mathtt{th}^{(1)} = \lceil \frac{v+3}{2} \rceil$, $\mathtt{th}^{(2)} = \mathtt{th}^{(3)} = \lceil \frac{v+1}{2} \rceil$. Fig. (a): DFR of QC-LDPC codes, Fig. (b): probability of convergence to a near codeword, Fig. (c): probability of convergence to a half codeword.

codewords the discrepancy vector completely converges to such structures after three iterations. While this criterion does not match the definition of $\mathtt{DFR}_{\mathtt{ncw}}$ and $\mathtt{DFR}_{\mathtt{hcw}}$ exactly, it provides a similar characterization of the convergence phenomenon. Fig. 6(b) and Fig. 6(c) report the decoding failures caused by near codewords and half codewords, compared to the corresponding failure probability predicted by our model. For our choice of thresholds, the error floor is dominated by half codewords rather than near codewords. Nevertheless, our model is precise in the floor region, where it predicts the dominance of failures caused by half codewords (with respect to hard-to-decode errors), and provides a conservative estimation for the convergence probability towards near codewords.

In Fig. 7, we provide the numerical results of a second set of tests, performed on QC codes with rate $\frac{3}{4}$ and a different threshold selection strategy to further validate our DFR estimates. Changing the parameters of the decoder and of the underlying code influences the family of errors dominating the error floor. In this instance, a mixture of hard-to-decode errors and near codewords dominate the error floor, with our model correctly estimating their contribution on the DFR.

We also test the results of our model against numerical simulations on cryptographic grade QC-MDPC codes. Fig. 8(a) shows the beginning of the waterfall region, for $n_0 = 2$, $v = 69$, $t = 130$, and different values of p. We tested the model on the threshold we found optimal for NIST category 1 (blue) and a set of sub-optimal, conservative threshold, that constitute the case in which the model in [8] provides an optimistic DFR estimate. As it can be seen, our model provides a conservative estimation for the DFR of QC-MDPC codes, regardless of the threshold choice. Willing to validate the near codewords and half codewords model, we compute the probability of converging towards such structures in conditions where the failure rate can be simulated. Fig. 8(b) shows the probability of converging to a near codeword (i.e., having at least one discrepancy within



(a) $\mathtt{DFR}_{\mathtt{QC}}$ (b) $p_{\mathtt{ncw}}$ (c) $p_{\mathtt{hcw}}$

Fig. 8: Comparison between our model (—) and numerical simulations (×) for three iterations $\mathtt{DFR}_{\mathtt{QC}}$, $p_{\mathtt{ncw}}$ and $p_{\mathtt{hcw}}$ values for QC-MDPCs. Parameters: n_0=2, v=69, t=130. Thresholds: $\mathtt{th}^{(1)}$=41, $\mathtt{th}^{(2)}$=35, $\mathtt{th}^{(3)}$=35 (ours, blue markers); $\mathtt{th}^{(1)}$=43, $\mathtt{th}^{(2)}$=37, $\mathtt{th}^{(3)}$=37 (conservative, red markers). Fig. (a): DFR for $9,500 \leq p \leq 12,500$, Fig. (b): $p = 13613$, prob. of convergence to a near codeword with fixed number of incorrect bits $m^{(0)}$, Fig. (c): $p = 13613$, prob. of convergence to a half codeword with fixed number of incorrect bits in one block $m_0^{(0)}$.

the support of the near codeword after three iterations, as in the definition of $p_{\mathtt{ncw}}$) after fixing the initial overlap between such near codeword and the error vector $\mathbf{e}$, denoted as $\mathcal{M}^{(0)}$ in Section 3.3, to $m^{(0)}$. Fig. 8(c) shows the probability of converging to a half codeword (as per the definition of $p_{\mathtt{hcw}}$) after fixing the overlap between one half of such codeword and the error vector, denoted as $\mathcal{M}_0^{(0)}$ in Section 3.4, to $m_0^{(0)}$. For the optimal threshold set, our model provides an accurate estimate of the convergence probability of such structures. For the conservative threshold set, the near codewords model provides an optimistic estimate for the same reason as in [8], namely, because the statistical dependencies between iterations are simplified. We find this not to be an issue, as the vast majority of failures for these codes are caused by hard-to-decode errors (that are instead modeled correctly), so the overall DFR estimate is still conservative. Moreover, the discrepancy only arises for non-optimal threshold sets, while for the values we select in our cryptosystem the model is remarkably accurate.

5 Parameter Design and Performance Evaluation

We design cryptosystem parameters matching the security requirements specified by NIST [46], for ease of comparison against the existing literature, i.e., such that breaking the public key cryptosystem takes as much as an exhaustive search on AES keys. This computational effort corresponds to 2^{λ} AES operations, with $\lambda \in \{128, 192, 256\}$, i.e. 2^{λ_C} Boolean operations, where $\lambda_C \in \{143, 207, 272\}$, or 2^{λ_Q} quantum operations, with $\lambda_Q \in \{154, 219, 283\}$, measured in terms of quantum gate count times depth of a quantum circuit.

We estimate the concrete computational effort of solving message and key recovery instances by computing cost of Information Set Decoding (ISD) solvers. To compute our estimates, we rely on three tools: i) *LEDAtools* (LT) [10] offers

finite-regime estimates for ISD variants up to the BJMM variant [15], under a logarithmic memory access cost model, with the recent version [48] correcting earlier cost inaccuracies and incorporating quantum cost models from [49,50,51]; `ii`) *Cryptographic Estimators* (CE) [29] provides estimates of the count of Boolean operations performed on bit vectors (counting each vector operation as one operation), with constant, logarithmic, square- and cube-root memory access cost models; `iii`) *CryptAttack Tester (CAT)* [18] provides a Boolean gate count of the operations of the ISD variants neglecting signal propagation costs, thus providing a lower bound on the computational effort. The computational effort estimates provided by LT, CE and CAT are paired with the DFR estimates from our closed form model.

To design cryptosystem parameters, we repeat the same procedure for each $(n_0, \lambda) \in \{2,3,4,5\}\times\{128,192,256\}$. The procedure generates a set $\mathcal{S}$ of tuples (n_0, λ, p, v, t), where p is a prime in the range $4,000<p<70,000$, and v, t are chosen such that the corresponding ISD weight w yields an asymptotic complexity estimate within a $\pm 20\%$ band around the classical effort, with the criterion in [56]. We verified a posteriori that no selected parameters lie on the boundary of the explored region, avoiding under-exploration of the parameters.

The procedure is repeated over three iterations. Each iteration consists of an evaluation step, computing the cost of message and key recovery attacks, and the DFR for all tuples in $\mathcal{S}$, and a pruning step, where: *i)* tuples failing to meet complexity or DFR thresholds are discarded; *ii)* among tuples with identical (n_0, p, v), we retain the one with smallest t in the first two iterations, and the one with smallest p in the final iteration. Each iteration uses increasingly accurate estimation techniques: **1)** a coarse evaluation using *LT*, based on the Stern [54] and quantum Prange [50] attacks, with DFR estimated using fixed thresholds: $\lfloor\frac{v+1}{2}\rfloor+\lfloor\frac{v}{12}\rfloor$, $\lfloor\frac{v+1}{2}\rfloor+\lfloor\frac{v}{24}\rfloor$, and $\lfloor\frac{v+1}{2}\rfloor+2$; **2)** a refined evaluation using the BJMM [15] and the quantum Lee-Brickell attack [51], with DFR estimated via greedy threshold optimization: starting from majority thresholds, each threshold is incrementally adjusted to minimize DFR, one at a time; **3)** a high-fidelity validation using *LT*, *CE*, and *CAT*, with DFR computed exhaustively over all admissible threshold triples.

We report in Tab. 2 the parameters and three-iterations decoder thresholds obtained during our exploration, together with the computational cost of running KRAs and MRAs against them, counted in number of AES-computations for AES-λ equivalent security, $\lambda \in \{128,192,256\}$. The *LT-C* and *LT-Q* columns report attack complexities obtained using *LT*, based on classical BJMM with logarithmic memory access cost and quantum Lee–Brickell strategies, respectively. The *CE* column reports the range of computational costs of the most efficient ISDs among all the variants implemented in the *Cryptographic Estimators* (CE) framework, assuming non-constant memory access (e.g., logarithmic, square root, or cubic).

Tab. 2 reports parameter sets where the computational effort matches the requirements of 2^λ AES computations when estimated with LT, whereas CE estimates do not account for the hidden factor, namely, the length of the vectors

Table 2: Parameters for a Niederreiter-style KEM, and relative security margin and DFR predictions. The value of the security parameter λ and its estimations refer to a computational security level equivalent to 2^λ AES-λ runs.

λ	Key Use	n_0	p	v	t	upc threshold 1st	2nd	3rd	Estimated λ LT-C	LT-Q	CE	CAT	$-\log_2(\mathbf{DFR})$
128	Eph.	2	12,413	69	131	42	36	35	128.7	145.1	128.2–130.0	129.6	64.4
		3	9,203	79	83	47	42	41	128.8	144.7	128.4–130.3	130.2	64.9
		4	7,757	81	66	49	42	41	128.7	143.2	125.7–129.1	127.1	65.3
		5	6,947	83	57	50	43	43	128.5	143.2	125.4–129.1	126.4	64.1
128	Long	2	13,613	69	130	41	35	35	128.2	144.5	127.6–129.3	128.8	131.6
		3	10,037	77	83	45	41	40	128.0	143.3	126.7–128.9	127.5	128.2
	Term	4	8,627	81	66	48	41	41	128.8	143.6	126.5–129.3	127.1	128.7
		5	7,829	83	57	49	42	42	128.8	143.7	125.6–129.3	126.4	130.3
192	Long	2	26,813	103	195	60	53	52	192.2	212.0	192.7–194.4	193.3	194.6
		3	19,709	115	124	66	60	59	193.1	212.0	192.5–195.2	193.2	193.8
	Term	4	16,829	121	98	70	62	61	192.2	211.4	192.0–195.0	192.5	192.1
		5	15,373	125	85	71	66	64	193.3	212.0	192.3–196.4	193.0	193.0
256	Long	2	43,451	135	261	77	69	69	257.5	279.8	258.2–260.5	258.9	256.5
		3	32,069	153	165	87	78	78	257.4	280.0	258.7–260.8	259.1	256.3
	Term	4	27,437	161	131	91	82	82	257.5	279.4	257.8–261.1	258.2	256.1
		5	24,733	165	113	93	85	84	257.6	278.9	256.6–260.7	256.7	256.4

involved in the computation, which is at least p. The estimated values of λ with CAT are either matching the requirements or at most short by 1.6, i.e., the estimate of the attack reports a computation $2^{1.6}$ times faster than expected. We consider this acceptable, as the values reported by CAT ignore the cost of signal transmission delays, which dominate the construction of a large cryptanalytic machine [18]. We report parameters designed for both long-term key usage (i.e., a keypair generated once and reused across multiple sessions), and ephemeral key usage (e.g., in TLS, where perfect forward secrecy is required). Ephemeral keys relax the stringent DFR requirements to a level where decoding failures are negligible from an engineering perspective (2^{-64}), improving performance.

Tab. 4 compares the performance of BIKE, HQC [44], and LEDAcrypt, paired with that of a new KEM obtained by applying the BIKE [7] and LEDAcrypt [13] construction to a Niederreiter-style PKE, based on QC-MDPC codes using our proposed parameter sets. The table reports the figures for $\lambda = 128$, figures for $\lambda \in \{192, 256\}$ are available in Appendix C and follow analogous trends. The $128^\dagger$ marker for BIKE indicates tha the optimized implementation of BIKE currently available employs the decoder from [59] (for which the authors report a DFR $\geq 2^{-116}$), and not the BIKE-flip decoder described in the latest specification [7], although the two decoders are expected to have similar runtimes. Timings are expressed in number of clock cycles for implementations employing Intel AVX2 as the instruction set extension of choice. Timings for BIKE and HQC are taken from the SUPERCOP [20] project performing independent benchmarking, on a Haswell CPU (NIST's ISA of choice for benchmarking) at 3.5 GHz (`titan0` machine), while timings for our design are obtained adapting

Table 3: Comparison of computational performances (kilo-Clock Cycles, kCC) of existing code-based post-quantum KEMs. LEDAcrypt, BIKE and our approach employ the same construction to turn a PKE into a KEM.

λ	KEM	n_0	pk size (B)	ctx size (B)	Single primitive Keygen (kCC)	Encrypt (kCC)	Decrypt (kCC)	Static KEM (kCC)	Ephem. KEM (kCC)
128	Our	2	1,702	1,734	541	107	617	724	1,266
128	Our	3	2,510	1,287	477	110	745	854	1,331
128	Our	4	3,237	1,111	427	120	890	1,010	1,437
128	Our	5	3,916	1,011	281	99	839	938	1,219
128	Our (Eph.)	2	1,552	1,585	512	85	231	-	828
128	Our (Eph.)	3	2,302	1,183	479	66	246	-	791
128	Our (Eph.)	4	2,910	1,002	270	47	248	-	565
128	Our (Eph.)	5	3,476	901	251	46	252	-	549
128†	BIKE	2	1,541	1,573	598	118	1,806	1,924	2,522
128	LEDAcrypt	2	3,536	3,560	4,573	221	1,578	1,799	6,372
128	LEDAcrypt	3	4,928	2,488	3,989	204	1,526	1,730	5,719
128	LEDAcrypt	4	6,096	2,056	3,861	225	1,502	1,727	5,588
128	HQC	-	2,249	4,433	94	246	419	665	759

the AVX2 optimized implementations from LEDAcrypt and BIKE, compiling them targetting a Haswell ISA and running on an Intel Core i7-12700K, at 3.8 GHz without Turbo Boost.

Concerning single primitives, we observe that our design allows large speedups, outperforming BIKE in keygen, encryption and decryption, both with parameters tuned for static keypair use and ephemeral keypair use. We observe that our design is between 1.9× and 2.7× faster than BIKE in static key KEM scenarios (i.e., considering encryption+decryption latencies) and achieves similar gains against the original LEDAcrypt design. Considering ephemeral keypair use, (e.g., to provide forward secrecy in TLS) we achieve a 4.6× speedup when compared to BIKE (key generation+encryption+decryption latency being the metric). Concerning ciphertext size, our design allows to fit into the 1280 B threshold [39] imposed by IPv6 lack of fragmentation, both in the ephemeral and static keypair scenario, making it a fitting candidate for a post-quantum Wireguard VPN [39]. Our improvements in ciphertext size increase the gap between HQC and Niederreiter-based QC-MDPC cryptosystems, bringing it to 4.4× for static keypairs. The bandwidth gain is significant, at 2.2×, even for ephemeral keypairs, where the figure of merit is the sum of public key and ciphertext sizes.

6 Concluding Remarks

In this work, we presented a closed-form decoding failure rate model for the three-iterations parallel decoder applied to QC-MDPC codes. We address the quasi-cyclic structure of the employed codes by accounting for near codewords and their generalization, presenting a full characterization of the *exact* decoder

employed in our QC-MDPC cryptosystem. We provide numerical validation of our model, showing that it correctly predicts both the waterfall and error floor regimes of QC codes. Having a sound estimate of the DFR allows to design Niederreiter-style cryptosystems with provable IND-CCA2 security [36].

With the use of our model, we show that the three-iterations decoder provides an significant reduction in public key and ciphertext size with respect to LEDAcrypt-KEM, the previous QC-MDPC based cryptosystem with bounded DFR. Our approach outperforms BIKE in ciphertext size and execution time, with speedups up to 4.6×, while improving security guarantees for long term keys. Moreover, we provide bandwidth reductions between 2.2× and 4.4× with respect to the NIST standardized HQC, with similar execution timings.

References

1. Alagic, G., Bros, M., Ciadoux, P., Cooper, D., Dang, Q., Dang, T., Kelsey, J., Lichtinger, J., Liu, Y.K., Miller, C., Moody, D., Peralta, R., Perlner, R., Robinson, A., Silberg, H., Smith-Tone, D., Waller, N.: Status Report on the Fourth Round of the NIST Post-Quantum Cryptography Standardization Process. https://doi.org/10.6028/NIST.IR.8545 (March 2025)
2. Albrecht, M.R., Bernstein, D.J., Chou, T., Cid, C., Gilcher, J., Lange, T., Maram, V., von Maurich, I., Misoczki, R., Niederhagen, R., Paterson, K.G., Persichetti, E., Peters, C., Schwabe, P., Sendrier, N., Szefer, J., Tjhai, C.J., Tomlinson, M., Wang, W.: Classic McEliece. https://csrc.nist.gov/projects/post-quantumcryptography/round-4-submissions (2022)
3. Alrabiah, O., Ananth, P., Christ, M., Dodis, Y., Gunn, S.: Ideal pseudorandom codes. In: Koucký, M., Bansal, N. (eds.) Proceedings of the 57th Annual ACM Symposium on Theory of Computing, STOC 2025, Prague, Czechia, June 23-27, 2025. pp. 1638–1647. ACM (2025). https://doi.org/10.1145/3717823.3718309
4. Annechini, A., Barenghi, A., Pelosi, G.: Bit-Flipping Decoder Failure Rate Estimation for (v, w)-Regular Codes. In: IEEE International Symposium on Information Theory, ISIT 2024, Athens, Greece, July 7-12, 2024. pp. 3374–3379. IEEE (2024). https://doi.org/10.1109/ISIT57864.2024.10619629
5. Annechini, A., Barenghi, A., Pelosi, G.: Estimating the Decoding Failure Rate of Binary Regular Codes Using Iterative Decoding. CoRR **abs/2401.16919** (2024). https://doi.org/10.48550/ARXIV.2401.16919
6. Apon, D., Perlner, R.A., Robinson, A., Santini, P.: Cryptanalysis of LEDAcrypt. In: Micciancio, D., Ristenpart, T. (eds.) Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part III. Lecture Notes in Computer Science, vol. 12172, pp. 389–418. Springer (2020). https://doi.org/10.1007/978-3-030-56877-1_14
7. Aragon, N., Barreto, P.S.L.M., Bettaieb, S., Bidoux, L., Blazy, O., Deneuville, J.C., Gaborit, P., Ghosh, S., Gueron, S., Güneysu, T., Melchor, C.A., Misoczki, R., Persichetti, E., Richter-Brockmann, J., Sendrier, N., Tillich, J.P., Vasseur, V., Zémor, G.: BIKE: Bit Flipping Key Encapsulation. Round 4 Submission - Version 5.2 - 10/10/2024. [Online] Available: https://bikesuite.org/files/v5.2/BIKE_Spec.2024.10.10.1.pdf (2025)

8. Arpin, S., Lau, J.B., Mesnard, A., Perlner, R.A., Robinson, A., Tillich, J., Vasseur, V.: Error floor prediction with markov models for QC-MDPC codes. In: Kalai, Y.T., Kamara, S.F. (eds.) Advances in Cryptology - CRYPTO 2025 - 45th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2025, Proceedings, Part I. Lecture Notes in Computer Science, vol. 16000, pp. 221–252. Springer (2025). https://doi.org/10.1007/978-3-032-01855-7_8
9. Baldi, M., Barenghi, A., Chiaraluce, F., Pelosi, G., Santini, P.: LEDAkem: A Post-quantum Key Encapsulation Mechanism Based on QC-LDPC Codes. In: Lange, T., Steinwandt, R. (eds.) Post-Quantum Cryptography - 9th International Conference, PQCrypto 2018, Fort Lauderdale, FL, USA, April 9-11, 2018, Proceedings. Lecture Notes in Computer Science, vol. 10786, pp. 3–24. Springer (2018). https://doi.org/10.1007/978-3-319-79063-3_1
10. Baldi, M., Barenghi, A., Chiaraluce, F., Pelosi, G., Santini, P.: A Finite Regime Analysis of Information Set Decoding Algorithms. Algorithms **12**(10), 209 (2019). https://doi.org/10.3390/A12100209
11. Baldi, M., Barenghi, A., Chiaraluce, F., Pelosi, G., Santini, P.: A Failure Rate Model of Bit-flipping Decoders for QC-LDPC and QC-MDPC Code-based Cryptosystems. In: Samarati, P., di Vimercati, S.D.C., Obaidat, M.S., Ben-Othman, J. (eds.) Proceedings of the 17th International Joint Conference on e-Business and Telecommunications, ICETE 2020 - Volume 2: SECRYPT, Lieusaint, Paris, France, July 8-10, 2020. pp. 238–249. ScitePress (2020). https://doi.org/10.5220/0009891702380249
12. Baldi, M., Barenghi, A., Chiaraluce, F., Pelosi, G., Santini, P.: Performance Bounds for QC-MDPC Codes Decoders. In: Wachter-Zeh, A., Bartz, H., Liva, G. (eds.) Code-Based Cryptography - 9th International Workshop, CBCrypto 2021, Munich, Germany, June 21-22, 2021 Revised Selected Papers. Lecture Notes in Computer Science, vol. 13150, pp. 95–122. Springer (2021). https://doi.org/10.1007/978-3-030-98365-9_6
13. Baldi, M., Barenghi, A., Chiaraluce, F., Pelosi, G., Santini, P.: LEDAcrypt - version 3.0 Specification (2020). [Online] Available: https://www.ledacrypt.org/documents/LEDAcrypt_v3.pdf (2025)
14. Baldi, M., Bodrato, M., Chiaraluce, F.: A New Analysis of the McEliece Cryptosystem Based on QC-LDPC Codes. In: Ostrovsky, R., Prisco, R.D., Visconti, I. (eds.) Security and Cryptography for Networks, 6th International Conference, SCN 2008, Amalfi, Italy, September 10-12, 2008. Proceedings. Lecture Notes in Computer Science, vol. 5229, pp. 246–262. Springer (2008). https://doi.org/10.1007/978-3-540-85855-3_17
15. Becker, A., Joux, A., May, A., Meurer, A.: Decoding Random Binary Linear Codes in $2^{n/20}$: How $1 + 1 = 0$ Improves Information Set Decoding. In: Pointcheval, D., Johansson, T. (eds.) Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7237, pp. 520–536. Springer (2012). https://doi.org/10.1007/978-3-642-29011-4_31
16. Berlekamp, E.R., McEliece, R.J., van Tilborg, H.C.A.: On the inherent intractability of certain coding problems (Corresp.). IEEE Trans. Inf. Theory **24**(3), 384–386 (1978). https://doi.org/10.1109/TIT.1978.1055873
17. Bernstein, D.J.: Grover vs. McEliece. In: Sendrier, N. (ed.) Post-Quantum Cryptography, Third International Workshop, PQCrypto 2010, Darmstadt, Germany,

May 25-28, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6061, pp. 73–80. Springer (2010). https://doi.org/10.1007/978-3-642-12929-2_6
18. Bernstein, D.J., Chou, T.: CryptAttackTester: high-assurance attack analysis. In: Reyzin, L., Stebila, D. (eds.) Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part VI. Lecture Notes in Computer Science, vol. 14925, pp. 141–182. Springer (2024). https://doi.org/10.1007/978-3-031-68391-6_5
19. Bernstein, D.J., Lange, T., Peters, C.: Smaller Decoding Exponents: Ball-Collision Decoding. In: Rogaway, P. (ed.) Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings. Lecture Notes in Computer Science, vol. 6841, pp. 743–760. Springer (2011). https://doi.org/10.1007/978-3-642-22792-9_42
20. Bernstein, D., Lange, T.: eBACS: ECRYPT Benchmarking of Cryptographic Systems. https://bench.cr.yp.to/supercop.html (2025)
21. Bonnetain, X., Bricout, R., Schrottenloher, A., Shen, Y.: Improved Classical and Quantum Algorithms for Subset-Sum. In: Moriai, S., Wang, H. (eds.) Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part II. Lecture Notes in Computer Science, vol. 12492, pp. 633–666. Springer (2020). https://doi.org/10.1007/978-3-030-64834-3_22
22. Both, L., May, A.: Optimizing BJMM with Nearest Neighbors : Full Decoding in 2 2 n / 21 and McEliece Security (2017)
23. Both, L., May, A.: Decoding Linear Codes with High Error Rate and Its Impact for LPN Security. In: Lange, T., Steinwandt, R. (eds.) Post-Quantum Cryptography - 9th International Conference, PQCrypto 2018, Fort Lauderdale, FL, USA, April 9-11, 2018, Proceedings. Lecture Notes in Computer Science, vol. 10786, pp. 25–46. Springer (2018). https://doi.org/10.1007/978-3-319-79063-3_2
24. Chevignard, C., Fouque, P., Schrottenloher, A.: Reducing the Number of Qubits in Quantum Information Set Decoding. In: Chung, K., Sasaki, Y. (eds.) Advances in Cryptology - ASIACRYPT 2024 - 30th International Conference on the Theory and Application of Cryptology and Information Security, Kolkata, India, December 9-13, 2024, Proceedings, Part VIII. Lecture Notes in Computer Science, vol. 15491, pp. 299–329. Springer (2024). https://doi.org/10.1007/978-981-96-0944-4_10
25. Drucker, N., Gueron, S., Kostic, D.: On constant-time QC-MDPC decoding with negligible failure rate. IACR Cryptol. ePrint Arch. p. 1289 (2019)
26. Ducas, L., Esser, A., Etinski, S., Kirshanova, E.: Asymptotics and Improvements of Sieving for Codes. In: Joye, M., Leander, G. (eds.) Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part VI. Lecture Notes in Computer Science, vol. 14656, pp. 151–180. Springer (2024). https://doi.org/10.1007/978-3-031-58754-2_6
27. Dumer, I.: On Minimum Distance Decoding of Linear Codes. In: Proc. 5th Joint Soviet-Swedish Int. Workshop Inform. Theory. pp. 50–52. Moscow (1991)
28. Esser, A.: Revisiting nearest-neighbor-based information set decoding. In: Quaglia, E.A. (ed.) Cryptography and Coding - 19th IMA International Conference, IMACC 2023, London, UK, December 12-14, 2023, Proceedings. Lecture Notes in Computer Science, vol. 14421, pp. 34–54. Springer (2023). https://doi.org/10.1007/978-3-031-47818-5_3

29. Esser, A., Bellini, E.: Syndrome Decoding Estimator. In: Hanaoka, G., Shikata, J., Watanabe, Y. (eds.) Public-Key Cryptography - PKC 2022 - 25th IACR International Conference on Practice and Theory of Public-Key Cryptography, Virtual Event, March 8-11, 2022, Proceedings, Part I. Lecture Notes in Computer Science, vol. 13177, pp. 112–141. Springer (2022). https://doi.org/10.1007/978-3-030-97121-2_5
30. Esser, A., Ramos-Calderer, S., Bellini, E., Latorre, J.I., Manzano, M.: An Optimized Quantum Implementation of ISD on Scalable Quantum Resources. IACR Cryptol. ePrint Arch. p. 1608 (2021)
31. Finiasz, M., Sendrier, N.: Security bounds for the design of code-based cryptosystems. In: Matsui, M. (ed.) Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings. Lecture Notes in Computer Science, vol. 5912, pp. 88–105. Springer (2009). https://doi.org/10.1007/978-3-642-10366-7_6
32. Gallager, R.G.: Low-density parity-check codes. IRE Trans. Inf. Theory **8**(1), 21–28 (1962). https://doi.org/10.1109/TIT.1962.1057683
33. Grover, L.K.: A Fast Quantum Mechanical Algorithm for Database Search. In: Miller, G.L. (ed.) Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996. pp. 212–219. ACM (1996). https://doi.org/10.1145/237814.237866
34. Guo, Q., Johansson, T., Nguyen, V.: A New Sieving-Style Information-Set Decoding Algorithm. IEEE Trans. Inf. Theory **70**(11), 8303–8319 (2024). https://doi.org/10.1109/TIT.2024.3457150
35. Guo, Q., Johansson, T., Stankovski, P.: A Key Recovery Attack on MDPC with CCA Security Using Decoding Errors. In: Cheon, J.H., Takagi, T. (eds.) Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10031, pp. 789–815 (2016). https://doi.org/10.1007/978-3-662-53887-6_29
36. Hofheinz, D., Hövelmanns, K., Kiltz, E.: A Modular Analysis of the Fujisaki-Okamoto Transformation. In: Kalai, Y., Reyzin, L. (eds.) Theory of Cryptography - 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10677, pp. 341–371. Springer (2017). https://doi.org/10.1007/978-3-319-70500-2_12
37. Kachigar, G., Tillich, J.: Quantum Information Set Decoding Algorithms. In: Lange, T., Takagi, T. (eds.) Post-Quantum Cryptography - 8th International Workshop, PQCrypto 2017, Utrecht, The Netherlands, June 26-28, 2017, Proceedings. Lecture Notes in Computer Science, vol. 10346, pp. 69–89. Springer (2017). https://doi.org/10.1007/978-3-319-59879-6_5
38. Kirshanova, E.: Improved Quantum Information Set Decoding. In: Lange, T., Steinwandt, R. (eds.) Post-Quantum Cryptography - 9th International Conference, PQCrypto 2018, Fort Lauderdale, FL, USA, April 9-11, 2018, Proceedings. Lecture Notes in Computer Science, vol. 10786, pp. 507–527. Springer (2018). https://doi.org/10.1007/978-3-319-79063-3_24
39. Lafourcade, P., Mahmoud, D., Ruhault, S., Taleb, A.R.: A Tale of Two Worlds, a Formal Story of WireGuard Hybridization. Cryptology ePrint Archive, Paper 2025/1179, short presentation at 34th USENIX Security Symposium (2025) (2025), https://eprint.iacr.org/2025/1179

40. Lee, P.J., Brickell, E.F.: An Observation on the Security of McEliece's Public-Key Cryptosystem. In: Günther, C.G. (ed.) Advances in Cryptology - EUROCRYPT '88, Workshop on the Theory and Application of of Cryptographic Techniques, Davos, Switzerland, May 25-27, 1988, Proceedings. Lecture Notes in Computer Science, vol. 330, pp. 275–280. Springer (1988). https://doi.org/10.1007/3-540-45961-8_25
41. Leon, J.S.: A probabilistic algorithm for computing minimum weights of large error-correcting codes. IEEE Trans. Inf. Theory **34**(5), 1354–1359 (1988). https://doi.org/10.1109/18.21270
42. May, A., Meurer, A., Thomae, E.: Decoding Random Linear Codes in $\widetilde{\mathcal{O}}(2^{0.054n})$. In: Lee, D.H., Wang, X. (eds.) Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings. Lecture Notes in Computer Science, vol. 7073, pp. 107–124. Springer (2011). https://doi.org/10.1007/978-3-642-25385-0_6
43. May, A., Ozerov, I.: On Computing Nearest Neighbors with Applications to Decoding of Binary Linear Codes. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I. Lecture Notes in Computer Science, vol. 9056, pp. 203–228. Springer (2015). https://doi.org/10.1007/978-3-662-46800-5_9
44. Melchor, C.A., Aragon, N., Bettaieb, S., Bidoux, L., Blazy, O., Bos, J., Deneuville, J.C., Dion, A., Gaborit, P., Lacan, J., Persichetti, E., Robert, J.M., Véron, P., Zémor, G.: Hamming Quasi-Cyclic (HQC) - Fourth round version. https://pqc-hqc.org/doc/hqc-specification_2024-02-23.pdf (2025)
45. Misoczki, R., Tillich, J., Sendrier, N., Barreto, P.S.L.M.: MDPC-McEliece: New McEliece Variants from Moderate Density Parity-Check Codes. IACR Cryptol. ePrint Arch. p. 409 (2012)
46. National Institute of Standards and Technology: Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process. https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf (2016)
47. Niederreiter, H.: Knapsack-type cryptosystems and algebraic coding theory. Problems of Control and Information Theory (1986), https://vdoc.pub/download/knapsack-type-cryptosystems-and-algebraic-coding-theory-2tgbc23plpdg
48. Perriello, S., Barenghi, A., Lunardi, E.: LEDAtools. https://github.com/tigerjack-crypto/LEDAtools/commit/4650fbfb24085e8da5e11aeb5faf0b55b1a47655
49. Perriello, S., Barenghi, A., Pelosi, G.: A Complete Quantum Circuit to Solve the Information Set Decoding Problem. In: Müller, H.A., Byrd, G., Culhane, C., Humble, T.S. (eds.) IEEE International Conference on Quantum Computing and Engineering, QCE 2021, Broomfield, CO, USA, October 17-22, 2021. pp. 366–377. IEEE (2021). https://doi.org/10.1109/QCE52317.2021.00056
50. Perriello, S., Barenghi, A., Pelosi, G.: Improving the Efficiency of Quantum Circuits for Information Set Decoding. ACM Transactions on Quantum Computing **4**(4) (2023). https://doi.org/10.1145/3607256
51. Perriello, S., Barenghi, A., Pelosi, G.: Quantum Circuit Design for the Lee-Brickell Based Information Set Decoding. In: Andreoni, M. (ed.) Applied Cryptography and Network Security Workshops - ACNS 2024 Satellite Workshops, AIBlock, AIHWS, AIoTS, SCI, AAC, SiMLA, LLE, and CIMSS, Abu Dhabi, United Arab Emirates,

March 5-8, 2024, Proceedings, Part II. Lecture Notes in Computer Science, vol. 14587, pp. 8–28. Springer (2024). https://doi.org/10.1007/978-3-031-61489-7_2
52. Prange, E.: The Use of Information Sets in Decoding Cyclic Codes. IEEE Transactions on Information Theory **8**(5), 5–9 (sep 1962). https://doi.org/10.1109/TIT.1962.1057777
53. Robinson, A.: FIPS 207: HQC-KEM. https://csrc.nist.gov/Presentations/2025/fips-207-hqc-kem
54. Stern, J.: A method for finding codewords of small weight. In: Cohen, G.D., Wolfmann, J. (eds.) Coding Theory and Applications, 3rd International Colloquium, Toulon, France, November 2-4, 1988, Proceedings. Lecture Notes in Computer Science, vol. 388, pp. 106–113. Springer (1988). https://doi.org/10.1007/BFB0019850
55. Tillich, J.: The Decoding Failure Probability of MDPC Codes. In: 2018 IEEE International Symposium on Information Theory, ISIT 2018, Vail, CO, USA, June 17-22, 2018. pp. 941–945. IEEE (2018). https://doi.org/10.1109/ISIT.2018.8437843
56. Torres, R.C., Sendrier, N.: Analysis of Information Set Decoding for a Sub-linear Error Weight. In: Takagi, T. (ed.) Post-Quantum Cryptography - 7th International Workshop, PQCrypto 2016, Fukuoka, Japan, February 24-26, 2016, Proceedings. Lecture Notes in Computer Science, vol. 9606, pp. 144–161. Springer (2016). https://doi.org/10.1007/978-3-319-29360-8_10
57. Vardy, A.: The intractability of computing the minimum distance of a code. IEEE Trans. Inf. Theory **43**(6), 1757–1766 (1997). https://doi.org/10.1109/18.641542
58. Vasseur, V.: Post-quantum cryptography: a study of the decoding of QC-MDPC codes. (Cryptographie post-quantique : étude du décodage des codes QC-MDPC). Ph.D. thesis, University of Paris, France (2021)
59. Wang, T., Wang, A., Wang, X.: Exploring Decryption Failures of BIKE: New Class of Weak Keys and Key Recovery Attacks. In: Handschuh, H., Lysyanskaya, A. (eds.) Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part III. Lecture Notes in Computer Science, vol. 14083, pp. 70–100. Springer (2023). https://doi.org/10.1007/978-3-031-38548-3_3

A Information Set Decoding

The best approach to solve message recovery (i.e., the Syndrome Decoding Problem, SDP) and key recovery (i.e., the Codeword Finding Problem, CFP) is the class of algorithms known as *Information Set Decoding (ISD)*. This technique encompasses a set of Las Vegas algorithm, that is, they always return a correct result but with a probabilistic runtime, given by the reciprocal of the probability of success of a single iteration.

Given an input instance $(\mathbf{H}, \mathbf{s}, w)$, the ISD technique begins by selecting uniformly at random a subset $\overline{\mathcal{I}} \subset [n]$ of size r. This selection can be formalized via a permutation matrix $\mathbf{P} \in \mathbb{F}_2^{n \times n}$, which reorders the coordinates of $\mathbf{H}$ so that the columns corresponding to $\overline{\mathcal{I}}$ are placed last. Applying this transformation yields $s = \mathbf{H}\mathbf{e} = (\mathbf{H}\mathbf{P})\left(\mathbf{P}^{-1}\mathbf{e}\right)$, where, by denoting $\widehat{\mathbf{H}} = \mathbf{H}\mathbf{P}$ and $\widehat{\mathbf{e}} = \mathbf{P}^{-1}\mathbf{e}$, we obtain the permuted system $\widehat{\mathbf{H}}\widehat{\mathbf{e}} = s$. The permuted error vector $\widehat{\mathbf{e}}$ is then partitioned as $\widehat{\mathbf{e}} = \left[\mathbf{e}^1 \mid \mathbf{e}^2\right]$, where $\mathbf{e}^1$ and $\mathbf{e}^2$ correspond to $\mathbf{e}_{|\mathcal{I}}$ and $\mathbf{e}_{|\overline{\mathcal{I}}}$, respectively. The next step applies Gauss-Jordan Elimination (GJE) to the augmented matrix $[\widehat{\mathbf{H}} \mid s]$ in an effort to bring $\widehat{\mathbf{H}}$ to systematic form. This procedure, represented through the matrix $\mathbf{U} \in \mathbb{F}_2^{r \times r}$, yields $\left[\widetilde{\mathbf{H}} \mid \tilde{\mathbf{s}}\right] = \mathbf{U}\left([\widehat{\mathbf{H}} \mid s]\right)$, where $\widetilde{\mathbf{H}}$ is further decomposed as $\widetilde{\mathbf{H}} = [\mathbf{W} \mid \mathbf{V}]$, with $\mathbf{W} \in \mathbb{F}_2^{r \times r}$ and $\mathbf{V} \in \mathbb{F}_2^{r \times k}$. When $\mathbf{W} = \mathbf{I}_r$ the system reduces to:

$$\tilde{\mathbf{s}} = \mathbf{V}\mathbf{e}^1 \oplus \mathbf{e}^2. \tag{1}$$

After this stage, the ISD variants differ in how the error weight is assumed to distribute across $\mathbf{e}^1$ and $\mathbf{e}^2$, and how the search for the weight-t error is carried out. In the following we proceed to describe the different ISD algorithms following the categorization provided in [18].

ISD-0. The ISD-0 group relies on plain linear-algebra techniques and enumeration strategies. In this setting, we assume to have a small number p of errors to reside within $\mathbf{e}^1$, and a zero-window of length ℓ within $\mathbf{e}^2$. The case $\ell = 0$, $p = 0$ corresponds to the original Prange's work. Such configuration leads to a simplification of Eq. (1) as $\tilde{\mathbf{s}} = \mathbf{e}^2$, and therefore the search for the error vector reduces to verify whether $\mathtt{wt}(\tilde{\mathbf{s}}) = t$ after the GJE procedure. The case $\ell = 0$, $p > 0$ corresponds to the Lee-Brickell variant [40], in which an exhaustive search is performed over all $\binom{k}{p}$ choices of p positions in $\mathbf{V}$. For each candidate, the algorithm checks whether the remaining syndrome has Hamming weight $t - p$, i.e., whether $\mathtt{wt}(\tilde{\mathbf{s}}) = t - p$. Finally, when $\ell > 0$, the row-wise split on the error vector $\mathbf{e}^2$ as $\left[\mathbf{0}_\ell \mid \mathbf{e}^{2b}\right]$ induces a similar split into $\tilde{\mathbf{s}}$ and $\mathbf{V}$, allowing to express Eq. (1) as: $\left[\tilde{\mathbf{s}}^a \,\middle|\, \tilde{\mathbf{s}}^b\right] = \left[\mathbf{V}^a \,\middle|\, \mathbf{V}^b\right]\mathbf{e}^1 \oplus \left[\mathbf{0}_\ell \,\middle|\, \mathbf{e}^{2b}\right]$, which separates into the pair of linear systems:

$$\tilde{\mathbf{s}}^a = \mathbf{V}^a\mathbf{e}^1, \tag{2}$$

$$\tilde{\mathbf{s}}^b = \mathbf{V}^b\mathbf{e}^1 \oplus \mathbf{e}^{2b}. \tag{3}$$

Such scheme corresponds to Leon's improvement [41], which performs a two-stage check: Eq. (2) is verified first, and only when it holds, Eq. (3) is evaluated, and if both checks pass the full error vector can be reconstructed.

For this group of algorithms, the probability that a randomly selected configuration yields the correct error vector is given by $\frac{\binom{r-\ell}{t-p}\binom{k-\ell}{p}}{\binom{n}{t}}$.. While the original algorithm by Prange incurs a per-iteration cost primarily dictated by the GJE procedure, the Leon and Lee–Brickell variants increase the computational cost per iteration. This additional cost is justified by a significantly higher probability of successfully identifying the correct error configuration, thereby improving the overall trade-off between time complexity and decoding success rate.

ISD-1. The ISD-1 family of algorithms extends earlier decoding strategies by introducing an additional parameter p' and partitioning the error vector $\mathbf{e}^1$ row-wise as $\left[\mathbf{e}^1_L \mid \mathbf{e}^1_R\right]$. Correspondingly, the submatrices $\mathbf{V}^a$ and $\mathbf{V}^b$ are further partitioned column-wise to match the structure of the error vector, allowing Eq. (2) to be rewritten as $\tilde{\mathbf{s}}^a = \mathbf{V}^a_L\mathbf{e}^1_L \oplus \mathbf{V}^a_R\mathbf{e}^1_R$.

The idea of this family of algorithms is that the searching for the correct vector can be done more efficiently than naive exhaustive enumeration, and instead solve an instance of the generalized k-sum problem through a meet-in-the-middle approach. The decoding algorithm assumes indeed a balanced error distribution between the two halves of the error vector, i.e., $\mathtt{wt}(\mathbf{e}^1_L) = \mathtt{wt}(\mathbf{e}^1_R) = p'$. The search for valid solutions is transformed into a collision-finding problem, and the algorithm precomputes values of $\tilde{\mathbf{s}} \oplus \mathbf{V}^a_L\mathbf{e}^1_L$ for all possible choices of $\mathbf{e}^1_L$ and then matches them against dynamically computed values of $\mathbf{V}^a_R\mathbf{e}^1_R$ for all $\mathbf{e}^1_R$ of the same weight.

As described, this technique corresponds to Stern's algorithm [54]. A similar and asymptotically equivalent algorithm was independently introduced by Dumer [27], who changes the size of $\mathbf{e}^1$ to $k+\ell$, and therefore avoids the discussion on the ℓ-window of zeros in $\mathbf{e}^2$. This strategy allows to reduce the structural requirements on the matrix $\mathbf{W}$, that now needs to only contain the identity submatrix $\mathbf{I}_{r-\ell}$ in its bottom-most portion, allowing for a partial GJE on $\widehat{\mathbf{H}}$. Consequently, the random selection of $\overline{\mathcal{I}}$ is performed over $\binom{[n]}{r-\ell}$, rather than $\binom{[n]}{r}$. In this grouping also fall the further refinements introduced in [19], where the authors proposed redistributing part of the error weight into the ℓ-length window.

A defining characteristic of ISD-1 algorithms is the use of super-polynomial memory. The collision search is optimized by precomputing and storing one half of the search space in a lookup table D, which is then queried during the enumeration of the remaining half. This design significantly reduces the time complexity of collision search, at the cost of memory usage dominated by the storage of $\binom{(k-l)/2}{p'}$ entries. Finally, ISD-1 algorithms increase the number of admissible error patterns per iteration, thereby improving the overall probability of success. The success probability of a single iteration is $\frac{\binom{r-\ell}{t-p}\left(\binom{(k-\ell)/2}{p'}\right)^2}{\binom{n}{t}}$.

ISD-2. The ISD-2 class of algorithms refines earlier meet-in-the-middle approaches by introducing additional parameters and deeper decompositions, enabling more efficient collision-based searches. Specifically, $\tilde{\mathbf{s}}^a$ is partitioned into two parts of length l_0 and l_1, with $l = l_0 + l_1$, which corresponds to a row-wise split of the matrix $\mathbf{V}^a$ into submatrices $\mathbf{V}^{aa} \in \mathbb{F}_2^{l_0 \times k}$ and $\mathbf{V}^{ab} \in \mathbb{F}_2^{l_1 \times k}$. Concurrently, the error vector halves $\mathbf{e}_L^1$ and $\mathbf{e}_R^1$ are each further split into two parts, yielding four subvectors of length $(k+l)/4$, each constrained to have Hamming weight p''. This setting defines a depth-2 meet-in-the-middle strategy. At the first level, candidate error vectors of length $(k+l)/4$ and weight p'' are combined in pairs to identify partial collisions. At the second level, these pairs are merged into length-$(k+l)/2$ vectors of weight p' that are tested for consistency with the syndrome. As with ISD-1 algorithms, the approach relies on exponential-sized data structures: one half of the search space is precomputed and stored for fast lookup, while the complementary half is exhaustively enumerated. This strategy was first instantiated in the MMT algorithm [42], which assumes the relation $p' = 2p''$. The subsequent BJMM algorithm [15] generalizes this setting by allowing $p' < 2p''$, and incorporates an additional level of list merging to further reduce time complexity.

A key insight in both MMT and BJMM is the use of approximate matching: rather than requiring full equality, lists are merged when partial agreement is detected in selected coordinates. This relaxed criterion introduces ambiguity (multiple error vectors may yield the same partial sum) but enlarges the collision space, increasing the probability of success and allowing for more aggressive pruning strategies. The resulting algorithms achieve improved asymptotic performance at the cost of increased memory consumption and algorithmic complexity.

Nearest-Neighbor Techniques. Nearest-neighbor (NN) search techniques were introduced as a means to reduce the cost of solving the approximate matching problem introduced in ISD-2 algorithms. The seminal work of May and Ozerov [43] reinterprets the collision-search step of the ISD-2 as instances of the nearest-neighbor problem in the Hamming metric. Their approach replaces the traditional exhaustive merging of lists with significantly faster NN queries, leveraging data structures designed to efficiently locate closely matching vectors.

Subsequent work by Both-May [23] applied this perspective to optimize the performance of advanced ISD algorithms such as MMT and BJMM. Additionally, in [29] the authors demonstrated that also ISD-1 variants can be reanalyzed through the lens of nearest-neighbor techniques, providing a unifying framework for understanding and improving their performance.

Sieving Techniques. Inspired by strategies developed in lattice-based cryptanalysis, sieving techniques have recently been adapted to the decoding setting. In [34] the authors introduce a sieving algorithm tailored for ISD, which incrementally constructs a list of candidate error vectors by exploiting the linearity of the code and the structure of the parity-check matrix.

The method begins with a randomly generated set of low-weight vectors that partially satisfy the parity-check constraints. The sieving process then iteratively

refines this list: at each step, pairs of vectors whose sum maintains the target weight are selected, and their sum is tested for increased compliance with the parity-check equations. Due to the additive closure of linear codes, these newly generated vectors remain valid candidates, and (by design) offer better alignment with the decoding goal.

In [26] the authors extend this framework by incorporating locality-sensitive hashing and filtering techniques to enhance the efficiency of the sieving process. Their asymptotic analysis demonstrates a performance improvement over BJMM, though the approach remains less competitive than the current best known nearest-neighbor-based algorithm of Both and May [23].

Quantum attacks. The first work showing an asymptotic speedup when using a quantum devices is presented in [17], in which the author demonstrated that Prange's algorithm can benefit from Grover's amplitude amplification [33], achieving a quadratic speed-up over the classical runtime. The theoretical framework resulted in several quantum circuit implementations [49,30,50,51,24], providing finite-regime complexity metrics in terms of number of gates, number of qubits and circuit depth.

Efforts to extend more sophisticated ISD algorithms and Stern to the quantum domain have produced only modest asymptotic improvements [37,38,21,26]. A major bottleneck in these approaches is their reliance on quantum random access memory (QRAM) of exponential size, an assumption deemed highly unrealistic in both current and future quantum hardware architectures. As a result, no explicit quantum circuit implementations for these advanced variants are currently known, and such models remain primarily of theoretical interest.

B Optimization Techniques for DFR Estimation

The mathematical model presented for estimating the decoding failure rate of a 3-iterations bit-flipping decoder is computationally demanding, and it cannot be implemented *as-is* to calculate the DFR of large instances. To obtain the decoding failure rate estimations for cryptographic-grade parameter sets, we employ a set of techniques and approximations to avoid unnecessary calculations while maintaining precise results. All the presented techniques have been introduced and validated incrementally, in order to assert their correctness.

The first optimization aims at reducing the computational load brought by the dependence of all the probabilities and p.m.f.s on the initial syndrome weight $\mathcal{W}$. We reduce the number of required calculations by averaging the probabilities and the p.m.f.s regarding the first iteration for each value of $\mathcal{W} = y$ (with $0 \leq y \leq p$), and performing the analysis of the second and third iteration with the averaged probabilities. This optimization removes the need for repeating the analysis of the second and third iteration for each possible value of y, providing a speedup of $\mathcal{O}(p)$ without impacting on the final result significantly.

The goal of the second optimization is to reduce the number of computations required to estimate the third iteration flipping probabilities $p_{\mathtt{flip}|abc}$, $a, b, c \in$

$\{0, 1\}$. To do so, during the analysis of the third iteration we simplify the behavior of the `upc` random variables, by removing their dependence on the *initial* state of each parity check (i.e. their parity during the first iteration). This modification allows for a speedup proportional to $\mathcal{O}(v^3)$.

The third approximation is based on the fact that, for every value k sufficiently large, we have that for the large majority of values $(a_{001}, a_{011}, a_{101}, a_{111})$ and $(b_{001}, b_{011}, b_{101}, b_{111})$ for $(\mathcal{E}_{001}, \mathcal{E}_{011}, \mathcal{E}_{101}, \mathcal{E}_{111})$ such that $a_{001} + a_{011} + a_{101} + a_{111} = b_{001} + b_{011} + b_{101} + b_{111} = k$, the following approximation holds:

$$\texttt{DFR}_\epsilon(a_{001}, a_{011}, a_{101}, a_{111}) \approx \texttt{DFR}_\epsilon(b_{001} + b_{011} + b_{101} + b_{111})$$

This relation corresponds to the fact that, if the *total number* of discrepancies k is sufficiently large after the second iteration (i.e. $k \geq 20$), the DFR estimate is almost the same regardless of the actual bit category of each discrepancy. This approximation allows us to compute the DFR estimate only once for each value k, selecting as concrete values of $(\epsilon_{001}, \epsilon_{011}, \epsilon_{101}, \epsilon_{111})$ (where $\epsilon_{001} + \epsilon_{011} + \epsilon_{101} + \epsilon_{111} = k$) the ones maximizing $\prod_{a,b\in\{0,1\}} \Pr(\mathcal{E}_{ab1} = \epsilon_{ab1})$, (i.e. the values with maximum likelihood). This modification, if also applied with the same line of reasoning to ϵ_{01} and ϵ_{11}, allows for a total speedup of $\mathcal{O}(nt^3)$.

All the techniques described in this section allow the model to be employed for DFR estimation even in the case of cryptographic-grade parameter sets, without requiring excessive computational resources and without impacting the accuracy of results.

C Complete Performance Comparison

Table 4: Comparison of computational performances (kilo-Clock Cycles, kCC) of existing code-based post-quantum key encapsulation methods. LEDAcrypt-KEM, BIKE and our approach are employing the same construction to turn a PKE into a KEM.

Security Parameter λ	**Primitive**	n_0	**pk Size**	**ctx Size**	**Keygen (kCC)**	**Encrypt (kCC)**	**Decrypt (kCC)**
128	Our	2	1,702	1,734	541	107	617
128	Our	3	2,510	1,287	477	110	745
128	Our	4	3,237	1,111	427	120	890
128	Our	5	3,916	1,011	281	99	839
128	Our (Eph.)	2	1,552	1,584	512	85	231
128	Our (Eph.)	3	2,302	1,183	479	66	246
128	Our (Eph.)	4	2,910	1,002	270	47	248
128	Our (Eph.)	5	3,476	901	251	46	252
128†	BIKE	2	1,541	1,573	598	118	1,806
128	LEDAcrypt	2	3,536	3,560	4,573	221	1,578
128	LEDAcrypt	3	4,928	2,488	3,989	204	1,526
128	LEDAcrypt	4	6,096	2,056	3,861	225	1,502
128	HQC	–	2,249	4,433	94	246	419
192	Our	2	3,352	3,384	1,625	252	1,866
192	Our	3	4,928	2,496	1,499	272	2,254
192	Our	4	6,312	2,136	1,318	302	2,665
192	Our	5	7,688	1,954	732	228	2,462
192†	BIKE	2	3,083	3,115	1,742	275	5,840
192	LEDAcrypt	2	6,584	6,616	9,947	486	4,131
192	LEDAcrypt	3	9,168	4,616	6,623	477	4,109
192	LEDAcrypt	4	11,568	3,888	7,293	635	4,487
192	HQC	–	4,522	8,978	220	557	877
256	Our	2	5,474	5,506	4,123	538	4,459
256	Our	3	8,018	4,041	1,949	379	4,223
256	Our	4	10,416	3,462	1,911	434	5,046
256	Our	5	12,368	3,124	1,838	495	5,821
256	LEDAcrypt	2	10,290	10,488	22,683	1,217	10,838
256	LEDAcrypt	3	14,544	7,312	15,710	1,198	10,546
256	LEDAcrypt	4	18,144	6,088	12,405	866	10,218
256	HQC	–	7,245	14,421	457	1,118	1,769