HAL
open science

On the Decoding Failure Rate of QC-MDPC Bit-Flipping Decoders

Nicolas Sendrier, Valentin Vasseur

To cite this version:

Nicolas Sendrier, Valentin Vasseur. On the Decoding Failure Rate of QC-MDPC Bit-Flipping Decoders. PQCrypto 2019 - Post-Quantum Cryptography 10th International Conference, May 2019, Chongqing, China. pp.404–416, 10.1007/978-3-030-25510-7_22 . hal-03139797

HAL Id: hal-03139797

https://inria.hal.science/hal-03139797v1

Submitted on 12 Feb 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

On the Decoding Failure Rate of QC-MDPC Bit-Flipping Decoders

Nicolas Sendrier[1] and Valentin Vasseur[1 2]

[1] Inria, Paris, France
FirstName.LastName@inria.fr,
[2] Université Paris Descartes, Sorbonne Paris Cité, France.

Abstract Quasi-cyclic moderate density parity check codes [1] allow the design of McEliece-like public-key encryption schemes with compact keys and a security that provably reduces to hard decoding problems for quasi-cyclic codes.

In particular, QC-MDPC are among the most promising code-based key encapsulation mechanisms (KEM) that are proposed to the NIST call for standardization of quantum safe cryptography (two proposals, BIKE and QC-MDPC KEM).

The first generation of decoding algorithms suffers from a small, but not negligible, decoding failure rate (DFR in the order of 10^{-7} to 10^{-10}). This allows a key recovery attack that exploits a small correlation between the faulty message patterns and the secret key of the scheme [2], and limits the usage of the scheme to KEMs using ephemeral public keys. It does not impact the interactive establishment of secure communications (*e.g.* TLS), but the use of static public keys for asynchronous applications (*e.g.* email) is rendered dangerous.

Understanding and improving the decoding of QCMDPC is thus of interest for cryptographic applications. In particular, finding parameters for which the failure rate is provably negligible (typically as low as 2^{-64} or 2^{-128}) would allow static keys and increase the applicability of the mentioned cryptosystems.

We study here a simple variant of bit-flipping decoding, which we call step-by-step decoding. It has a higher DFR but its evolution can be modelled by a Markov chain, within the theoretical framework of [3]. We study two other, more efficient, decoders. One is the textbook algorithm implemented as in [3]. The other is (close to) the BIKE decoder. For all those algorithms we provide simulation results, and, assuming an evolution similar to the step-by-step decoder, we extrapolate the value of the DFR as a function of the block length. This will give an indication of how much the code parameters must be increased to ensure resistance to the GJS attack.

1 Introduction

Moderate Density Parity Check (MDPC) codes were introduced for cryptography[3] in [1]. They are related to Low Density Parity Check (LDPC) codes, but instead of admitting a sparse parity check matrix (with rows of small constant weight) they admit a somewhat sparse parity check matrix, typically with rows of Hamming weight $O(\sqrt{n})$ and length n. Together with a quasi-cyclic structure they allow the design of a McEliece-like public-key encryption scheme [4] with reasonable key size and a security that provably reduces to generic hard problems over quasi-cyclic codes, namely the hardness of decoding and the hardness of finding low weight codewords.

Because of these features, QC-MDPC have attracted a lot of interest from the cryptographic community. In particular, two key exchange mechanisms "BIKE" and "QC-MDPC KEM" were recently proposed to the NIST call for standardization of quantum safe cryptography [4].

The decoding of MDPC codes can be achieved, as for LDPC codes, with iterative decoders [5] and in particular with the (hard decision) bit flipping algorithm, which we consider here. Using soft decision decoding would improve performance [6], but would also increase the complexity and make the scheme less suitable for hardware and embedded device implementations, which is one of its interesting features [7]. There are several motivations for studying MDPC decoding. First, since QC-MDPC based cryptographic primitives may become a standard, it is worth understanding and improving their engineering and in particular the decoding algorithm, which is the bottleneck of their implementation. The other motivation is security. A correlation was established by Guo, Johansson and Stankovski in [2] between error patterns leading to a decoding failure and the secret key of the scheme: the sparse parity check matrix of a QC-MDPC code. This GJS attack allows the recovery of the secret by making millions of queries to a decryption oracle. To overcome the GJS attack, one must find instances of the scheme for which the Decoding Failure Rate (DFR) is negligible. This is certainly possible by improving the algorithm and/or increasing the code parameters, but the difficulty is not only to achieve a negligible DFR (a conservative target is a failure rate of the same order as the security requirements, that is typically 2^{-128}) but to prove, as formally as possible, that it is negligible when the numbers we consider are out of reach by simulation.

In this work, we recall the context and the state-of-the-art in §2, mostly results of [3] as well as some new properties. In §3 we describe a new decoder, the step-by-step bit flipping algorithm, and its probabilistic model. This algorithm is less efficient than the existing techniques, but, thanks to the model, its DFR can be estimated. Finally in §4 we compare the DFR prediction of our model with the DFR obtained with simulations. We compare this with BIKE decoder

[3] MDPC were previously defined, in a different context, by Ouzan and Be'ery in 2009, `http://arxiv.org/abs/0911.3262`

[4] `https://csrc.nist.gov/Projects/Post-Quantum-Cryptography`

simulation and try to extrapolate its behavior even when the DFR cannot be obtained by simulation.

Notation

- Throughout the paper, the matrix $\mathbf{H} \in \{0,1\}^{r \times n}$ will denote the sparse parity check matrix of a binary MDPC code of length n and dimension $k = n - r$. The rows of $\mathbf{H}$, the parity check equations, have a weight $w = O(\sqrt{n})$, and are denoted eq_i, $0 \leq i < r$. The columns of $\mathbf{H}$, transposed to become row vectors, are denoted h_j, $0 \leq j < n$.
- For any binary vector v, we denote v_i its i-th coordinate and $|\mathrm{v}|$ its Hamming weight. Moreover, we will identify v with its support, that is $i \in \mathrm{v}$ if and only if $v_i = 1$.
- Given two binary vector u and v of same length, we will denote $\mathrm{u} \cap \mathrm{v}$ the set of all indices that belong to both u and v, or equivalently their component-wise product.
- The scalar product of u and v is denoted $\langle \mathrm{u}, \mathrm{v} \rangle \in \{0,1\}$.
- For a random variable X we write $X \sim \mathrm{Bin}(n,p)$ when X follows a binomial distribution:
$$\Pr[X = k] = f_{n,p}(k) = \binom{n}{k} p^k (1-p)^{n-k}.$$
- In a finite state machine, the event of going from a state S to a state T in at most I iterations is denoted:
$$S \xrightarrow{I} T\,.$$

2 Bit Flipping Decoding

The bit flipping algorithm takes as argument a (sparse) parity check matrix $\mathbf{H} \in \{0,1\}^{r \times n}$ and a noisy codeword $\mathrm{y} \in \{0,1\}^n$. It flips a position if the proportion of unsatisfied equations containing that position is above some threshold. A parity check equation eq_i is unsatisfied if the scalar product $\langle \mathrm{eq}_i, \mathrm{y} \rangle = 1$. The proportion of unsatisfied equations involving j is $|\mathrm{s} \cap \mathrm{h}_j| \,/\, |\mathrm{h}_j|$, where $\mathrm{s} = \mathrm{y}\mathbf{H}^T$ denotes the syndrome. In practice (see [1]), the bit flipping decoder of Algorithm 1 is paralell:

Algorithm 1 The Bit Flipping Algorithm

Require: $\mathbf{H} \in \{0,1\}^{(n-k) \times n}$, $\mathrm{y} \in \{0,1\}^n$
  while $\mathrm{y}\mathbf{H}^T \neq 0$ **do**
    $\mathrm{s} \leftarrow \mathrm{y}\mathbf{H}^T$
    $\tau \leftarrow \mathrm{threshold}(\mathit{context})$
    for $j \in \{0, \ldots, n-1\}$ **do**
      if $|\mathrm{s} \cap \mathrm{h}_j| \geq \tau\, |\mathrm{h}_j|$ **then**
        $y_j \leftarrow 1 - y_j$
  return y

the syndrome is updated after all tests and flips. The choice of a particular threshold τ depends on the context, that is $\mathbf{H}$, y, and possibly anything the algorithm can observe or compute. With an appropriate choice of threshold and assuming that $\mathbf{H}$ is sparse enough and y close enough to the code, the algorithm terminates with high probability.

2.1 QC-MDPC-McEliece

Bit flipping decoding applies in particular to quasi-cyclic moderate density parity check (QC-MDPC) codes which can be used in a McEliece-like encryption scheme [1]. In the cryptographic context, those codes are often of rate 1/2, length n, dimension $k = n/2$ (codimension $r = n/2$). The parity matrix has row weight $w = O(\sqrt{n})$ and is regular (all columns have the same weight $d = w/2$). The bit flipping decoding correct $t = O(\sqrt{n})$ errors. Parameters n, r, w, t must be fine-tuned so that the cryptosystem is secure and the decoding failure rate (DFR) low. The implementation of Algorithm 1 for QC-MDPC was considered in several works. Different strategies have been considered so far to choose a good threshold: relying on the value of $\max_j(|\mathrm{s} \cap \mathrm{h}_j| \,/\, |\mathrm{h}_j|)$ [1], using fixed values [8] or using the syndrome weight [3, 9]. The last two strategies require, for each set of parameters, a precomputation based on simulation to extract the proper threshold selection rule. For the parameters of Table 1 those algorithms typically require less than 10 iterations for a DFR that does not exceed 10^{-7}. Table 1 gives the sets of parameters of the BIKE proposal [10] to NIST. The bit flipping decoder of

Table 1. BIKE Parameters (security against classical adversary)

n	r	w	t	security
20 326	10 163	142	134	128
39 706	19 853	206	199	192
65 498	32 749	274	264	256

BIKE is slightly more elaborated. Its DFR appears to be below 10^{-10} but this is unfortunately difficult to establish from mere simulations.

2.2 A Key Recovery Attack

In a recent work, Guo, Johansson, and Stankovski (GJS) [2] were able to exhibit a correlation between faulty error patterns and the secret key of the scheme (the sparse parity check matrix of the QC-MDPC code). An attacker that has access to a decryption oracle for a given secret key, may perform a key recovery attack by collecting and analyzing thousands (at least) of error patterns leading to a failure. This limits the cryptographic usage of QC-MDPC to key encapsulation mechanisms with ephemeral key. To safely extend the usage of the scheme to static keys (allowing one-pass asynchronous key exchange, for instance for email),

one needs to lower the failure rate to something negligible. Depending on the adversarial model the DFR could be a small constant, like 2^{-64}, or even a value which decreases exponentially with the security claim (typically $2^{-\lambda}$ for λ bits of security).

2.3 Bit Flipping Identities

More details about this section can be found in [3, part III]. We assume that the MDPC code is quasi-cyclic and regular. That means that in the parity check matrix $\mathbf{H} \in \{0,1\}^{r \times n}$, every row has the same weight w and every column has the same weight d. If $r = n/2$, which is the most common situation in cryptographic applications, then we have $w = 2d$.

We consider an instance of the decoding algorithm, the noisy codeword is denoted y and is at distance $t = O(\sqrt{n})$ of the code. With probability overwhelmingly close to 1, the corresponding error e is unique. Its syndrome is $\mathrm{s} = \mathrm{y}\mathbf{H}^T = \mathrm{e}\mathbf{H}^T$.

Definition 1. *For a parity matrix* $\mathbf{H}$ *and an error vector* e*, the number of unsatisfied parity check equations involving the position* j *is* $\sigma_j(\mathrm{e}, \mathbf{H}) = |\mathrm{h}_j \cap \mathrm{e}\mathbf{H}^{\mathrm{T}}|$*. We call this quantity a* counter*. The number of equations affected by exactly* ℓ *errors is*

$$E_\ell(\mathrm{e}, \mathbf{H}) = \left| \left\{ i \in \{0, \ldots, r-1\} \colon \ |\mathrm{eq}_i \cap \mathrm{e}| = \ell \right\} \right|.$$

The quantities e and $\mathbf{H}$ are usually clear form the context. We will omit them and simply write σ_j and E_ℓ.

Proposition 1. *The following identities are verified for all* e *and all* $\mathbf{H}$*:*

$$\sum_{\ell \text{ odd}} E_\ell = \left|\mathrm{e}\mathbf{H}^T\right|, \qquad \sum_j \sigma_j = w \left|\mathrm{e}\mathbf{H}^T\right|, \qquad \sum_{j \in \mathrm{e}} \sigma_j = \sum_{\ell \text{ odd}} \ell E_\ell \,.$$

The Counter Distributions. If e is distributed uniformly among the words of weight t, we have $\sigma_j \sim \mathrm{Bin}(d, \pi_{e_j})$ with

$$\pi_1 = \sum_{\ell \text{ even}} \frac{\binom{w-1}{\ell}\binom{n-w}{t-1-\ell}}{\binom{n-1}{t-1}}, \text{ and } \pi_0 = \sum_{\ell \text{ odd}} \frac{\binom{w-1}{\ell}\binom{n-w}{t-\ell}}{\binom{n-1}{t}}. \tag{1}$$

The above distribution is valid on average. However, the following facts are remarked in [3].

1. It does not accurately predict the counter values for an individual value of e. In fact, the counters tend to grow with the syndrome weight.
2. Even if the inital error pattern is uniformly distributed (of fixed weight), this is no longer true after the first iteration and the deviation from (1) is significant.

Conditioning the Counter Distributions with the Syndrome Weight. We denote $t = |\mathrm{e}|$ and $S = \left|\mathrm{e}\mathbf{H}^T\right|$ the syndrome weight. A better to model for σ_j is given by the distribution $\mathrm{Bin}(d, \pi'_{e_j})$ where (see [3])

$$\pi'_1 = \frac{S+X}{dt}, \pi'_0 = \frac{(w-1)S - X}{d(n-t)} \text{ with } X = \sum_{\ell \text{ odd}} (\ell - 1)E_\ell \,. \tag{2}$$

The above formulas depends on the codes parameters n, w, d, on the error weight $t = |\mathrm{e}|$, on the syndrome weight $S = \left|\mathrm{e}\mathbf{H}^T\right|$ but also on the quantity $X = \sum_{\ell>0} 2\ell E_{2\ell+1}$. Here, we wish to obtain an accurate model for the counter distribution which only depends on S and t. We must somehow get rid of X. In practice $X = 2E_3 + 4E_5 + \cdots$ is not dominant in the above formula (for relevant QC-MDPC parameters) and we will replace it by its expected value.

Proposition 2. *When* e *is chosen uniformly at random of weight* t *the expected value of* $X = \sum_{\ell>0} 2\ell E_{2\ell+1}$ *given that* $S = \left|\mathrm{e}\mathbf{H}^T\right|$ *is,*

$$\overline{X}(S,t) = \frac{S\sum_\ell 2\ell\rho_{2\ell+1}}{\sum_\ell \rho_{2\ell+1}} \text{ where } \rho_\ell = \frac{\binom{w}{\ell}\binom{n-w}{t-\ell}}{\binom{n}{t}} \,.$$

Remarks.

- The counter distributions above are extremely close to the observations when the error pattern e is uniformly distributed of fixed weight.
- The model gradually degenerates as the number of iterations grows, but remains relatively accurate in the first few iterations of Algorithm 1, that is even when e is not uniformly distributed.

2.4 Adaptive Threshold

Within this model, a good threshold for Algorithm 1 is $\tau = T/d$ where T is the smallest integer such that (recall that $f_{d,\pi}$ is defined in the notation)

$$t f_{d,\pi'_1}(T) \geq (n-t) f_{d,\pi'_0}(T) \,.$$

We will use this threshold selection rule in the sequel of the paper. Note that it is very consistent with the thresholds that were empirically determined in [9] to optimize Algorithm 1.

2.5 Estimating the Syndrome Weight

The probability for a parity equation eq of weight w to be unsatisfied when the error e is distributed uniformly of weight t is equal to

$$\bar{\rho} = \sum_{\ell \text{ odd}} \Pr[|\mathrm{eq} \cap \mathrm{e}| = \ell] = \sum_{\ell \text{ odd}} \frac{\binom{w}{\ell}\binom{n-w}{t-\ell}}{\binom{n}{t}} = \sum_{\ell \text{ odd}} \rho_\ell$$

The syndrome weight $S = \left|\mathrm{e}\mathbf{H}^T\right|$ is equal to the number of unsatisfied equations and thus its expectation in $\bar{\rho}r$. For a row-regular[5] MDPC code the syndrome weight follows the binomial distribution $\mathrm{Bin}(r, \bar{\rho})$. However, it was remarked in [3] that for a regular MDPC code, there was a dependence between equations and the syndrome weight followed a different distribution.

In the following proposition we give the distribution of the syndrome weight when the error is uniformly distributed of weight t and the matrix $\mathbf{H}$ is regular.

Proposition 3. *Let* $\mathbf{H}$ *be a binary* $r \times n$ *row-regular matrix of row weight* w*. When the error* e *is uniformly distributed of weight* t*, the syndrome weight* $S = \left|\mathrm{e}\mathbf{H}^T\right|$ *follows the distribution*

$$\Pr[S = \ell] = f_{r,\bar{\rho}}(\ell)$$

and if $\mathbf{H}$ *is regular of column weight* d*, we have*

$$P_\ell(t) = \Pr[S = \ell \mid \mathbf{H} \textit{ is regular}] = \frac{f_{r,\bar{\rho}}(\ell)h(\ell)}{\sum_{k\in\{0,\ldots,r\}} f_{r,\bar{\rho}}(k)h(k)} \tag{3}$$

where for $\ell \in \{0, \ldots, r\}$ *(*$*$ *is the discrete convolution operation and* ^{*n} *is the* n*-fold iteration of the convolution with itself)*

$$h(\ell) = g_1^{*\ell} * g_0^{*(r-\ell)}(dt)$$

with, for $k \in \{0, \ldots, w\}$*,*

$$g_1(k) = \begin{cases} \frac{\binom{w}{k}\binom{n-w}{t-k}}{\binom{n}{t}}\frac{1}{\bar{\rho}} & \textit{if } k \textit{ is odd} \\ 0 & \textit{otherwise} \end{cases}; \qquad g_0(k) = \begin{cases} \frac{\binom{w}{k}\binom{n-w}{t-k}}{\binom{n}{t}}\frac{1}{1-\bar{\rho}} & \textit{if } k \textit{ is even} \\ 0 & \textit{otherwise} \end{cases}.$$

The above distribution takes into account the regularity but not the quasi-cyclicity. Nevertheless, experimental observation shows that it is accurate for quasi-cyclic matrices, at least in the range useful for QC-MDPC codes.

3 Step-by-Step Decoding

In Algorithm 1 the positions with a counter value above the threshold are flipped all at once. In Algorithm 2 only one position is flipped at a time. The benefit, in contrast with algorithm 1, is that we can predict the evolution of the decoder. For example, when position j with counter σ_j is flipped, the syndrome weight becomes $|\mathrm{s}| + |\mathrm{h_j}| - 2\sigma_j$. And the error weight is either increased or decreased by one.

To instanciate Algorithm 2, we will use the threshold selection rule described in §2.4 and to sample j, we uniformly pick an unverified equation then a position in this equation, that is

$$i \xleftarrow{\$} \{i, |\mathrm{eq}_i \cap \mathrm{y}| \text{ odd}\}; j \xleftarrow{\$} \mathrm{eq}_i$$

[5] all rows of $\mathbf{H}$ have the same weight w, no condition on the column weight

Algorithm 2 The Step-by-Step Bit Flipping Algorithm

Require: $\mathbf{H} \in \{0,1\}^{(n-k)\times n}$, $\mathrm{y} \in \{0,1\}^n$
  while $\mathrm{y}\mathbf{H}^T \neq 0$ **do**
    $\mathrm{s} \leftarrow \mathrm{y}\mathbf{H}^T$
    $\tau \leftarrow \mathrm{threshold}(\mathit{context})$
    $j \leftarrow \mathrm{sample}(\mathit{context})$
    if $|\mathrm{s} \cap \mathrm{h}_j| \geq \tau\,|\mathrm{h}_j|$ **then**
      $y_j \leftarrow 1 - y_j$
  return y

(where $x \xleftarrow{\$} X$ means we pick x uniformly at random in the set X). With this rule, and using the model for counter distributions given in §2.3, the probability to pick $j \in \mathrm{e}$ is

$$\frac{\sum_{j' \in \mathrm{e}} \sigma_{j'}}{\sum_{j'} \sigma_{j'}} = \frac{S+X}{wS} = \frac{1}{w}\left(1 + \frac{X}{S}\right),$$

where S in the syndrome weight and $X = 2E_2 + 4E_4 + \cdots$ is defined in §2.3. Note that this probability is slightly above $1/w$ and larger in general than t/n, the same probability when j is chosen randomly.

3.1 Modeling the Step-by-step Decoder

We assume here that $\mathbf{H}$ is the sparse parity check matrix of a regular QC-MDPC code, with row weight w and column weight d. We model the step-by-step decoder as a finite state machine (FSM) with a state (S,t) with S the syndrome weight and t the error weight.

We consider one loop of Algorithm 2. The position j is sampled, the corresponding counter value is denoted $\sigma = |\mathrm{s} \cap \mathrm{h}_j|$ and the threshold is $T = \tau\,|\mathrm{h}_j| = \tau d$. There are 3 kinds of transition,

- if $\sigma < T$, then $(S,t) \to (S,t)$, with probability p
- if $\sigma \geq T$ and $j \in \mathrm{e}$, then $(S,t) \to (S+d-2\sigma, t-1)$, with probability p_σ^-
- if $\sigma \geq T$ and $j \notin \mathrm{e}$, then $(S,t) \to (S+d-2\sigma, t+1)$, with probability p_σ^+

and the transition probabilities are given in the following proposition.

Proposition 4.

$$p_\sigma^- = \frac{t\sigma f_{d,\pi_1'}(\sigma)}{wS}, p_\sigma^+ = \frac{(n-t)\sigma f_{d,\pi_0'}(\sigma)}{wS}, p = \sum_{\sigma<T}(p_\sigma^- + p_\sigma^+),$$

where $f_{d,\pi}(i) = \binom{d}{i}\pi^i(1-p)^{d-i}$ *and* π_0', π_1' *are given in* (2) *in §2.3.*

The above machine does not correctly take into account the situation where the algorithm is unable to find a suitable position to flip. We modify it as follows: one step of the new machine will iterate the loop until a flip occurs. We call j the flipped position and σ its counter. The possible transitions are now,

– if no high enough counter is found, then $(S,t) \to \mathrm{L}$, with probability p_L
– if $\sigma \geq T$ and $j \in \mathrm{e}$, then $(S,t) \to (S+d-2\sigma, t-1)$, with probability p'^{-}_{σ}
– if $\sigma \geq T$ and $j \notin \mathrm{e}$, then $(S,t) \to (S+d-2\sigma, t+1)$, with probability p'^{+}_{σ}

where the state L corresponds to the situation where there no position exists with a suitable counter.

Proposition 5.

$$p'^{-}_{\sigma} = p^{-}_{\sigma} \frac{1-p_L}{1-p}, p'^{+}_{\sigma} = p^{+}_{\sigma} \frac{1-p_L}{1-p},$$

where $p, p^{-}_{\sigma}, p^{+}_{\sigma}$ *are given in Proposition 4, and*

$$p_L = \left(\sum_{\sigma<T} f_{d,\pi'_1}(\sigma) \right)^t \cdot \left(\sum_{\sigma<T} f_{d,\pi'_0}(\sigma) \right)^{n-t},$$

where $f_{d,\pi}(i) = \binom{d}{i} \pi^i (1-p)^{d-i}$ *and* π'_0, π'_1 *are given in* (2) *in* §*2.3.*

Note. As mentionned in §2.3, we have replaced X by $\overline{X}$ (Proposition 2) in π'_0, π'_1 in all the results of this section.

3.2 Computing the DFR

To compute the theoretical DFR in our model, we will add another state F corresponding to a decoding failure. We assume the stochastic process we have defined in the previous section is a time-homogeneous Markov chain. For any starting state (S,t) we wish to determine with which probability the FSM reaches the failure state after an infinite number of iterations:

$$\mathrm{DFR}(S,t) = \Pr[(S,t) \xrightarrow{\infty} \mathrm{F}].$$

Since we assumed an infinite number of iterations, we need to fix an error weight above which the decoder fails, say t_{fail}. Similarly, to simplify the computation, we assume that when t is small enough, say below t_{pass} the decoder always succeeds. We have $\forall t \leq t_{\mathrm{pass}}, \Pr[(S,t) \xrightarrow{\infty} \mathrm{F}] = 0$ and $\forall t > t_{\mathrm{fail}}, \Pr[(S,t) \xrightarrow{\infty} \mathrm{F}] = 1$.

Notice that as long as $T \geq \frac{d+1}{2}$ (which is always the case here), $\forall \sigma \geq T, S_\sigma < S$ therefore these probabilities can be computed by induction with S in ascending order. Finally, the probability to successfully decode a vector y noised with a uniformly distributed error of weight t in the model is

$$\mathrm{DFR}(t) = \sum_S P_S(t)\, \mathrm{DFR}(S,t)$$

where $P_S(t)$ is the distribution of the syndrome weight given in Proposition 3.

3.3 Using t Alone for the State is not Enough

In the analysis of LDPC decoding, starting with Gallager's work [5], it is usually assumed that the error pattern remains uniformly distributed throughout the decoding process. This assumption greatly simplifies the analysis. It is correct for LDPC codes, but unfortunately not for MDPC codes.

Assuming uniform distribution of the error during all the decoding is equivalent to adopting a stochastic model in which the decoder state is described by the error weight alone. From our analysis, we easily derive the transition probabilities as

$$\Pr[t \to (t \pm 1)] = \sum_S P_S(t) \sum_{S'} \Pr[(S,t) \to (S', t \pm 1)] \,.$$

The corresponding Markov chain can be computed. We observe a huge discrepancy. For instance, for parameters $(n, r, w, t) = (65500, 32750, 274, 264)$ the observed failure rate is in the order of 10^{-4} for the step-by-step decoder while the model predicts less than 10^{-12}. The difference is even higher for larger block size.

4 Simulation

We simulate here three algorithms:

Algorithm 1: as in [3], using the threshold selection rule of §2.3.
Algorithm 2: step-by-step bit flipping as in the model of §3.
BIKE decoder: adapted from [10].

The parameters are those of BIKE-1 Level 5 ($d = 137$, $w = 274$, $t = 264$) with rate $1/2$ and a varying block size r.

The true BIKE decoder is tuned for $r = 32749$. We adapt it here for variable r. The BIKE decoder starts with one iteration of Algorithm 1 and ends with Algorithm 2 and a threshold $\tau = 0.5$. Between the two there a "gray zone" step described in [10].

Let us point out that the threshold selection rule of §3 (used for Algorithms 1 and 2 and the model is not honest. It assumes that the error weight is known throughout the computation while obviously a real decoder has no access to that information. However the main objective here was to compare the simulation and the model, and both of them "cheat". Moreover, we believe that finding the "good" threshold can always be achieved for an extra computational cost without cheating. Note finally that the BIKE algorithm outperform the others without relying on the knowledge of the error weight.

In figure 1, we compare the DFR derived from our model to the one obtained by Monte Carlo simulations of the three above decoders.

While our model is slighly optimistic, the DFR curve we obtain from it follows the same evolution as the one obtained by simulation. Assuming this stays true for higher block length values, this allows us to observe the evolution of the DFR for block lengths that are out of the reach of simulation (when the DFR becomes too small to be measured). Observing the model behavior beyond the

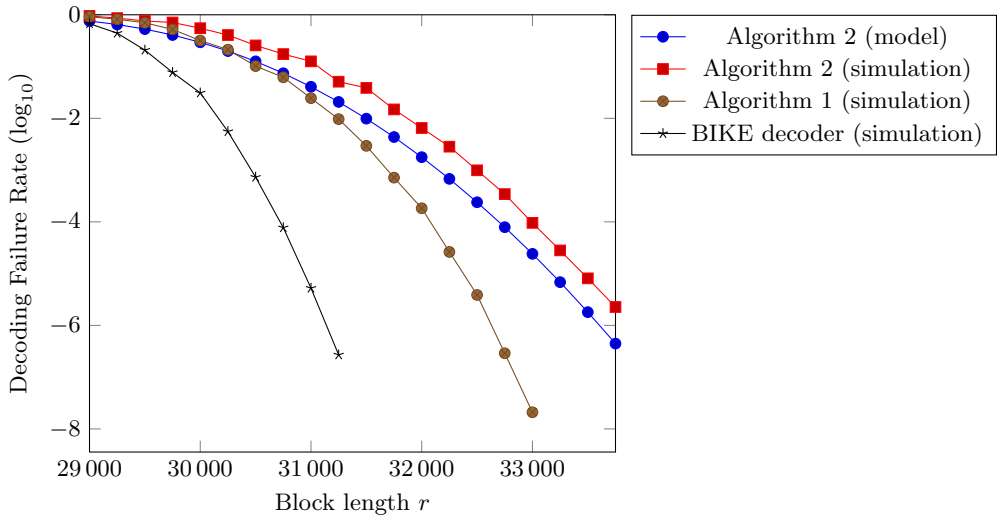

Figure 1. DFR of the step-by-step algorithm in the models and from simulations (infinite number of iterations)

range plotted in Figure 1, we have noticed that the DFR evolves in two phases, in the first phase ($r < 37\,500$) it closely fits a quadratic curve and in the second phase it is linear, which is consistent with the asymptotic analysis in [11].

Table 2. Extrapolating QC-MDPC Parameters

	(a)	(b)	(c)	(d)	(e)	(f)
Algorithm 1	−21.6	−21.6	35 498	39 652	34 889	36 950
BIKE	−48.8	−57.0	33 594	37 149	32 983	34 712
Algorithm 2	−10.4	−11.5	39 190	46 884	37 537	40 952

(a): linearly extrapolated value for $\log_2(p_{\text{fail}}(32\,749))$;
(b): quadratically extrapolated value for $\log_2(p_{\text{fail}}(32\,749))$;
(c): minimal r such that $p_{\text{fail}}(r) < 2^{-64}$ assuming a linear evolution;
(d): minimal r such that $p_{\text{fail}}(r) < 2^{-128}$ assuming a linear evolution;
(e): minimal r such that $p_{\text{fail}}(r) < 2^{-64}$ assuming a quadratic evolution;
(f): minimal r such that $p_{\text{fail}}(r) < 2^{-128}$ assuming a quadratic evolution

We also observe a quadratic evolution with the algorithms that we implemented and tested. We have no indication on when or if the curve changes from quadratic to linear so our model suggests that an optimistic extrapolation of the DFR would be quadratic and a pessimistic one would be linear. We give some of those extrapolations in Table 2. We denote $p_{\text{fail}}(r)$ the DFR for block size r.

5 Conclusion

We have presented here a variant of the bit flipping decoder of QC-MDPC codes, namely the step-by-step decoder. It is less efficient than the existing decoders, but can be accurately modeled by a Markov chain. If we assume that the evolution of the DFR of other related algorithms, and in particular BIKE, follow the same kind of evolution, we are able to give estimates for their DFR and also of the block size we would need to reach a low enough error rate. For BIKE-1 level 5, we estimate the DFR between 2^{-49} and 2^{-57}. As shown in Table 2, the amount by which the block size should be increased to reach a DFR of 2^{-64} or even 2^{-128} seems to be relatively limited, only 1% to 15%. This suggest that with an improvement of the decoder efficiency, the original BIKE parameters might not to far to what is needed for resisting the GJS attack and allow static keys.

References

1. Misoczki, R., Tillich, J.P., Sendrier, N., Barreto, P.S.L.M.: MDPC-McEliece: New McEliece variants from moderate density parity-check codes. In: Proc. IEEE Int. Symposium Inf. Theory - ISIT. (2013) 2069–2073
2. 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. Volume 10031 of LNCS. (2016) 789–815
3. Chaulet, J.: Étude de cryptosystèmes à clé publique basés sur les codes MDPC quasi-cycliques. PhD thesis, University Pierre et Marie Curie (March 2017)
4. McEliece, R.J. In: A Public-Key System Based on Algebraic Coding Theory. Jet Propulsion Lab (1978) 114–116 DSN Progress Report 44.
5. Gallager, R.G.: Low Density Parity Check Codes. M.I.T. Press, Cambridge, Massachusetts (1963)
6. Baldi, M., Santini, P., Chiaraluce, F.: Soft mceliece: MDPC code-based mceliece cryptosystems with very compact keys through real-valued intentional errors. In: Proc. IEEE Int. Symposium Inf. Theory - ISIT, IEEE Press (2016) 795–799
7. Heyse, S., von Maurich, I., Güneysu, T.: Smaller keys for code-based cryptography: QC-MDPC McEliece implementations on embedded devices. In Bertoni, G., Coron, J., eds.: Cryptographic Hardware and Embedded Systems - CHES 2013. Volume 8086 of LNCS., Springer (2013) 273–292
8. Chou, T.: Qcbits: Constant-time small-key code-based cryptography. In Gierlichs, B., Poschmann, A.Y., eds.: CHES 2016. Volume 9813 of LNCS., Springer (2016) 280–300
9. Chaulet, J., Sendrier, N.: Worst case QC-MDPC decoder for mceliece cryptosystem. In: IEEE Conference, ISIT 2016, IEEE Press (2016) 1366–1370
10. Aguilar Melchor, C., Aragon, N., Barreto, P., Bettaieb, S., Bidoux, L., Blazy, O., Deneuville, J.C., Gaborit, P., Gueron, S., Güneysu, T., Misoczki, R., Persichetti, E., Sendrier, N., Tillich, J.P., Zémor, G.: BIKE. first round submission to the NIST post-quantum cryptography call (November 2017)
11. Tillich, J.P.: The decoding failure probability of MDPC codes. In: 2018 IEEE International Symposium on Information Theory, ISIT 2018, Vail, CO, USA, June 17-22, 2018. (2018) 941–945

A Precisions on §2.5

Let us write $\omega_i = |\mathrm{eq}_i \cap \mathrm{e}|$ for all row $0 \leq i < r$. Then $S = \sum_i (\omega_i \mod 2)$. Moreover, for a regular code, the ω_i variables always verify the following property:

$$\sum_{i \in \{0,\ldots,r-1\}} \omega_i = d \cdot |\mathrm{e}| \ .$$

We wish to determine the distribution of $\sum_{i \in \{0,\ldots,r-1\}} \omega_i$ knowing the syndrome weight. We can then apply the Bayes' theorem to obtain the result of proposition 3:

$$\begin{aligned}
\Pr[S = \ell \mid \mathbf{H} \text{ is regular}] &= \Pr[S = \ell \mid \textstyle\sum_{i \in \{0,\ldots,r-1\}} \omega_i = d\,|\mathrm{e}|] \\
&= \frac{\Pr[\sum_{i \in \{0,\ldots,r-1\}} \omega_i = d\,|\mathrm{e}| \mid S = \ell]\,\Pr[S = \ell]}{\Pr[\sum_{i \in \{0,\ldots,r-1\}} \omega_i = d\,|\mathrm{e}|]} \\
&= \frac{\Pr[\sum_{i \in \{0,\ldots,r-1\}} \omega_i = d\,|\mathrm{e}| \mid S = \ell]\,\Pr[S = \ell]}{\sum_{k \in \{0,\ldots,r\}} \Pr[\sum_{i \in \{0,\ldots,r-1\}} \omega_i = d\,|\mathrm{e}| \mid S = k]\,\Pr[S = k]}
\end{aligned}$$

for $\ell \in \{0, \ldots, r\}$.

We can reorder the ω_i such that for $i \in \{0, \ldots, S-1\}$, ω_i is odd and for $i \in \{S, \ldots, r-1\}$, ω_i is even.

For all i, if we see ω_i as a random variable, if $i \in \{0, \ldots, S-1\}$ (resp $i \in \{S, r-1\}$) its probability mass function is g_1 (resp. g_0).

Assuming independence of those random variables, we have:

$$\forall k \in 0, \ldots, wS, \Pr[\textstyle\sum_{i \in \{0,\ldots,S-1\}} \omega_i = k] = g_1^{*S}(k)\,;$$
$$\forall k \in 0, \ldots, w(r-S), \Pr[\textstyle\sum_{i \in \{S,\ldots,r-1\}} \omega_i = k] = g_0^{*(r-S)}(k)\,.$$

Hence the result.

B Proof of Proposition 2

When e is uniformely distributed of weight t, the probability for a parity equation to contain ℓ errors is equal to

$$\rho_\ell = \frac{\binom{w}{\ell}\binom{n-w}{t-\ell}}{\binom{n}{t}}.$$

If we condition this probability on ℓ being odd, we obtain

$$\rho'_\ell = \begin{cases} \frac{\rho_\ell}{\sum_{k \text{ odd}} \rho_k} & \text{when } \ell \text{ is odd;} \\ 0 & \text{otherwise.} \end{cases}$$

We know that exactly S equations have an odd numbers of errors. Thus we find the expected value for X with

$$\overline{X}(S,t) = S \sum_\ell (\ell - 1)\rho'_\ell\,.$$

C Comment on §3

Proposition 4 and Proposition 5 are proven from the identities given in §2.3. The factors $t\sigma/wS$ and $(n-t)\sigma/wS$ derive from the sampling rule.