About Low DFR for QC-MDPC Decoding

Nicolas Sendrier, Valentin Vasseur

About Low DFR for QC-MDPC Decoding*

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

[1] Inria, Paris, France
FirstName.LastName@inria.fr,
[2] Université de Paris, Paris, France.

Abstract McEliece-like code-based key exchange mechanisms using QC-MDPC codes can reach IND-CPA security under hardness assumptions from coding theory, namely quasi-cyclic syndrome decoding and quasi-cyclic codeword finding. To reach higher security requirements, like IND-CCA security, it is necessary in addition to prove that the decoding failure rate (DFR) is negligible, for some decoding algorithm and a proper choice of parameters. Getting a formal proof of a low DFR is a difficult task. Instead, we propose to ensure this low DFR under some additional security assumption on the decoder. This assumption relates to the asymptotic behavior of the decoder and is supported by several other works. We define a new decoder, Backflip, which features a low DFR. We evaluate the Backflip decoder by simulation and extrapolate its DFR under the decoder security assumption. We also measure the accuracy of our simulation data, in the form of confidence intervals, using standard techniques from communication systems.

1 Introduction

Moderate Density Parity Check (MDPC) codes were introduced for cryptography[3] in [17]. 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 [16] 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, the BIKE suite of key exchange mechanisms has been selected to the second round of the NIST call for standardization of quantum safe cryptographic primitives [4]. The second round BIKE document [1]

* This work was supported by the ANR CBCRYPT project, grant ANR-17-CE39-0007 of the French Agence Nationale de la Recherche.

[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

mentions the Backflip decoder, a new variant of bit flipping decoding, as well as claims about its DFR. The low DFR is an essential feature to achieve IND-CCA security, and incidentally to resist to the GJS key recovery attack [11] which exploits decoding failures.

The Backflip algorithm and its DFR claims were never fully described in an academic work. We provide here the rationale and a precise description of Backflip as well as a justification and a description of the simulation methodology and assumptions that were used for estimating the DFR.

The decoding of MDPC codes can be achieved, as for LDPC codes, with iterative decoders [10] and in particular with the (hard decision) bit flipping algorithm. The Backflip algorithm will introduce soft information (*i.e.* reliability information) by flipping coordinates for a limited time which depends on the confidence we have in each flipping decision. This confidence is measured from quantities that were already computed in bit flipping decoders and are thus available at no extra cost. This way, the new decoder will use soft decision decoding, as in [2,14] for instance, while keeping the very simple logic and arithmetic which makes it suited to hardware and embedded device implementations [13].

No theoretical argument is known to guaranty a low DFR for the Backflip decoder. We will resort to simulation. However proving a very low DFR (*e.g.* 2^{-128}) cannot be achieved by simulation alone. Instead, we will use simulation data to extrapolate the DFR in a region of parameters where it is too small to be estimated by simulation. This extrapolation technique for the DFR is valid under an additional assumption on the asymptotic behavior of the decoder.

The paper is organized as follows. The §2 will state and comment the security assumption related to decoding. The §3 will describe the Backflip algorithm and explain its rationale. The §4 will explain, under the decoder security assumption, how to obtain DFR estimates with accurate simulation data.

Notation. 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 vectors 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 as vectors.

1.1 Previous Works

A binary Quasi-Cyclic Moderate Density Parity Check (QC-MDPC) code, is a quasi-cyclic code which admits a parity check matrix of density proportional to $1/\sqrt{n}$ where n is the code length. A QC-MDPC code can efficiently correct an error of Hamming weight t proportional to $\sqrt{n}$ thanks to bit flipping decoding (Algorithm 1). A $(2r, r, w, t)$-QC-MDPC-McEliece is an instance of the McEliece scheme [16] using an QC-MDPC code of index 2 correcting t errors. Such a code admits a parity check matrix consisting of two sparse circulant blocks of size $r \times r$ and row weight $w/2$ proportional to $\sqrt{n}$.

We denote $\mathcal{R} = \mathbb{F}_2[x]/(x^r - 1)$. The ring $\mathcal{R}$ is isomorphic to $r \times r$ circulant matrices. The scheme is fully described by the knowledge of the error weight t

and of two polynomials h_0, h_1 of $\mathcal{R}$ of Hamming weight $w/2$. Its security relates to the following hard problems.

Problem 1. $(2,1)$-QC Syndrome Decoding
Instance: s, h in $\mathcal{R}$, an integer $t > 0$.
Property: There exists e_0, e_1 in $\mathcal{R}$ such that $|e_0| + |e_1| \leq t$ and $e_0 + e_1 h = s$.

Problem 2. $(2,1)$-QC Codeword Finding
Instance: h in $\mathcal{R}$, an integer $w > 0$.
Property: There exists h_0, h_1 in $\mathcal{R}$ such that $|h_0| + |h_1| = w$ and $h_1 + h_0 h = 0$.

In the rest of §1.1 we will consider an instance of a $(2r, r, w, t)$-QC-MDPC-McEliece scheme. The code length is $n = 2r$, its dimension is $k = r$, and we will denote $d = w/2$.

Security Assumptions. The security of QC-MDPC-McEliece for QC codes of index 2 (and rate 1/2) relies on two assumptions.

Assumption 1 *Problem 1 is hard on average over s, h in $\mathcal{R}$.*

Assumption 2 *Problem 2 is hard on average over h in $\mathcal{R}$.*

The above assumptions are enough to guaranty the one-wayness of the underlying encryption primitive. With the ad-hoc conversion they will also be enough to prove that the related Key Encapsulation Mechanism (KEM) is IND-CPA (see [1]). To go further and design and prove an IND-CCA KEM, a further assumption on the decoding failure rate (DFR) is required. This will be examined later in the paper.

Tightness and Best Known Attacks. The security proofs for QC-MDPC code-based schemes are tight in the following sense: the proofs require the decisional versions of Problem 1 and 2 to be hard on average for the size (r, t) and (r, w) while the best known attacks only require to solve the search version of either Problem 1 or 2 for the same size (r, t) or (r, w). Note that there is a search to decision reduction for Syndrome Decoding [9] but it has not been transferred so far to the quasi-cyclic case. The best solvers for Problem 1 and 2 use Information Set Decoding (ISD). As explained in [17], it is possible to make use of the quasi-cyclicity together with the multitarget variant of ISD [20] to slightly improve the decoding. If $\mathrm{WF}(n, k, t)$ is the expected cost of the best ISD solver for the decoding t errors in a binary linear $[n, k]$ code, the cost of the best solver for Problem 1 and Problem 2 is upper bounded respectively by $\frac{\mathrm{WF}(2r,r,t)}{\sqrt{r}}$ and $\frac{\mathrm{WF}(2r,r,w)}{r}$. When $t \ll r$, which is the case here, it was shown in [4] that asymptotically the complexity exponent of all variants of ISD was equivalent to the complexity exponent of Prange algorithm [18], that is $\mathrm{WF}(n, k, t) = 2^{t \log_2 \frac{n}{n-k}(1+o(1))}$. In particular, the value $\mathrm{WF}(2r, r, t) = 2^{t(1+o(1))}$ does not depend, for its first order term, on the block size r.

QC-MDPC-McEliece Practical Security. The security of an instance of the $(2r, r, w, t)$-QC-MDPC-McEliece scheme reduces to Problem 1 with parameters (r, t) and Problem 2 with parameters (r, w). We give in Table 1 the security exponents for the message and key securities, respectively $\frac{\mathrm{WF}(2r,r,t)}{\sqrt{r}}$ and $\frac{\mathrm{WF}(2r,r,w)}{r}$ when the workfactor is computed for the BJMM variant of ISD [3] using the methodology described in [12]. We remark that, as expected, the security exponent grows very slowly with the block size r. The parameters of Table 1 are those of the NIST proposal BIKE [1]. For each security level, the first and second rows correspond respectively to the IND-CPA and IND-CCA variants.

	(r, w, t)	Problem 2 key security	Problem 1 message security
BIKE level 1	$(10163, 142, 134)$	129.5	128.6
	$(11779, 142, 134)$	129.8	128.9
BIKE level 3	$(19853, 206, 199)$	191.6	192.1
	$(24821, 206, 199)$	192.4	193.0
BIKE level 5	$(32749, 274, 264)$	258.0	255.9
	$(40597, 274, 264)$	258.8	256.9

Table 1. Security exponent of $(2r, r, w, t)$-QC-MDPC-McEliece (BIKE parameters)

Bit Flipping Decoding. All decoders for QC-MDPC codes derive from the bit flipping decoder given in Algorithm 1 in its syndrome decoding variant. In Algorithm 1, the counter $|\mathrm{s}' \cap \mathrm{h}_j|$ is the number of unsatisfied equations involving j. Positions with high counter values are flipped. If $\mathrm{s}' = \mathrm{s} - \mathrm{e}\mathbf{H}^\intercal$ for some $(\mathrm{e}, \mathrm{s}')$, with $\mathbf{H}$, s' and e sparse enough, then the algorithm return e with high probability. The variant presented allows noisy syndrome with $u > 0$ (as needed for BIKE-3),

Algorithm 1 Bit Flipping Algorithm, (Noisy-)Syndrome Decoding Variant

Require: $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$, $\mathrm{s} \in \mathbb{F}_2^{n-k}$, integer $u \geq 0$ //$u > 0$ *for noisy syndrome*
Ensure: $\left|\mathrm{s} - \mathrm{e}'\mathbf{H}^\intercal\right| \leq u$ or time $>$ max_time
  $\mathrm{e}' \leftarrow \mathbf{0}$; time $\leftarrow 1$
  while $\left|\mathrm{s} - \mathrm{e}'\mathbf{H}^\intercal\right| > u$ **and** time $\leq$ max_time **do**
    time $\leftarrow$ time $+ 1$
    $\mathrm{s}' \leftarrow \mathrm{s} - \mathrm{e}'\mathbf{H}^\intercal$
    $T \leftarrow$ `threshold`(*context*)
    for $j \in \{0, \ldots, n-1\}$ **do**
      if $|\mathrm{s}' \cap \mathrm{h}_j| \geq T$ **then** //h_j *the* j*-th column of* $\mathbf{H}$
        $e'_j \leftarrow 1 - e'_j$
  return e'

else if $u = 0$, it defines the usual QC-MDPC decoding (used by BIKE-1/2).

Threshold Selection. Selecting the proper threshold is an essential step of the bit flipping algorithm. In the current state of the art [5, 21] the optimal threshold is given as a function of the syndrome weight and of the error weight. We consider an execution of Algorithm 1. At any time, let e denote the (remaining) error vector, the syndrome is $\mathrm{s}' = \mathrm{e}\mathbf{H}^\intercal = \mathrm{s} - \mathrm{e}'\mathbf{H}^\intercal$. The optimal threshold is defined as

$T = \texttt{threshold}(S, t')$ is the smallest integer T such that

$$(n-t')\binom{d}{T}\pi_0^T(1-\pi_0)^{d-T} \leq \begin{cases} t'\binom{d}{T}\pi_1^T(1-\pi_1)^{d-T} & \text{if } \pi_1 < 1 \\ 1 & \text{else} \end{cases}$$

where

$$\pi_1 = \frac{S + X(S,t')}{t'd}, \pi_0 = \frac{(w-1)S - X(S,t')}{(n-t')d}$$

and

$$X(S,t') = \frac{S\sum_{\ell \text{ odd}} (\ell-1)\rho_\ell(t')}{\sum_{\ell \text{ odd}} \rho_\ell(t')} \text{ with } \rho_\ell(t') = \frac{\binom{w}{\ell}\binom{n-w}{t'-\ell}}{\binom{n}{t'}}.$$

Figure 1. Threshold function

in Figure 1 with the call $\texttt{threshold}(|\mathrm{e}\mathbf{H}^\intercal|, |\mathrm{e}|)$. Note that the syndrome weight $S = |\mathrm{e}\mathbf{H}^\intercal| = |\mathrm{s} - \mathrm{e}'\mathbf{H}^\intercal|$ is always known by the the decoder while the error weight $t' = |\mathrm{e}|$ is only known at the first iteration, since $|\mathrm{e}| = t$ by design. Later on the exact error weight is unknown and a value for t' has to be chosen somehow. One possibility is to guess it by using the fact that the expected value of S is a fonction of t', $\mathbb{E}(S) = r\sum \rho_{2\ell+1}(t')$. Though this identity is only exact at the first iteration, it provides a good enough estimate of t' as a function of S. Finally, even though the procedure for computing the threshold seems involved, it is not the case in practice. For a given set of parameters, the threshold is a function of S which can be precomputed and is usually well approximated by an affine function.

Attacks on the Decoder. The bit flipping algorithm is iterative and probabilistic. In particular, it has a small but positive Decoding Failure Rate (DFR). This is not an issue if the scheme uses ephemeral keys (*e.g.* TLS using BIKE specification) but creates a threat when static keys are used. It was shown in [11] how to exploit the decoding failures to recover the secret key. This stresses the importance of reducing the DFR to a negligible value. This is mandatory to reach CCA security and requires an evolution of the decoder, an increase of the parameters, an accurate estimate of the DFR, and arguments to support the accuracy of this estimate.

The GJS technique was later extended [8] to efficiently recover the secret key if the adversary has access to the number of effective decoding iterations. The

latter attack stresses the need of a constant-time implementation when static keys are used. Allowing constant-time implementation may in turn require an evolution of the decoder and of the system parameters.

1.2 Related Works

The Backflip decoding algorithm and claims about its DFR were given in [1]. The purpose of this work is to detail and support those claims. A simplified bit flipping variant, the step-by-step decoder, is modelled with a Markov chain in [21], the model has a DFR which decreases provably exponentially with the block size. The asymptotic analysis of [22] of QC-MPDC also predicts an exponential decrease in the range of interest for cryptography, but the analysis is made in a specific setting and cannot be directly applied to practical BIKE decoder and parameters. Another recent work [7] explores another decoder variant for BIKE to reach simultaneously a low DFR and a constant-time implementation.

2 An Additional Security Assumption

Preliminary: Tangent Extrapolation. When observing the plot of the logarithm of the simulated DFR versus the block size r (the other parameters w and t are fixed), one observes that it is always concave. It seems rather natural to assume that it will remain so and to extrapolate the DFR accordingly. The strategy will then consist in making a simulation for the largest possible r to accurately measure the tangent of the lowest possible point of the curve. For instance in

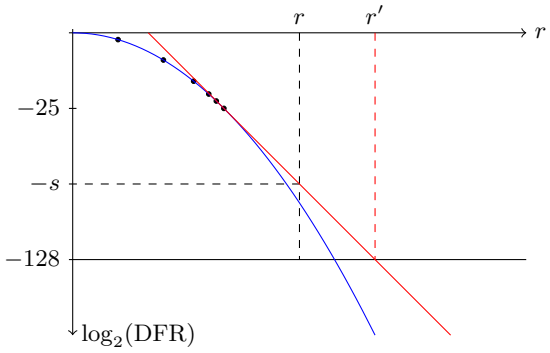

Figure 2. DFR Tangent Extrapolation

Figure 2, suppose the low curve (blue) is giving the $\log_2(\text{DFR})$ and we are able to make accurate simulation as long as the DFR is above 2^{-25} (black dots). Taking the tangent at the last point gives us the red line from which we derive an upper bound r' for a block size with a DFR below 2^{-128} as well as an upper bound 2^{-s} for the DFR for a given block size r.

2.1 Target Parameters

We will consider here three levels of security named according to the NIST postquantum security nomenclature. For each security level λ below, we denote r_λ^{CPA} the block size of the IND-CPA variants of BIKE (1 and 2).

Level 1: $(w,t) = (142, 134)$ for $\lambda = 128$ bits of classical security, $r_\lambda^{\text{CPA}} = 10\,163$
Level 3: $(w,t) = (206, 199)$ for $\lambda = 192$ bits of classical security, $r_\lambda^{\text{CPA}} = 19\,853$
Level 5: $(w,t) = (274, 264)$ for $\lambda = 256$ bits of classical security, $r_\lambda^{\text{CPA}} = 32\,749$

As mentionned previously, the security of the $(2r, r, w, t)$-QC-MDPC-McEliece scheme only marginally depends of the block size r. To reach IND-CCA security the block size must be increased slightly, at most 25% [1]. To allow constant-time implementation, the current state-of-art [7] suggests an extra 10%. We thus expect that for any security level λ the values of interest for the block size r lie in the interval $[r_\lambda^{\text{CPA}}, 2r_\lambda^{\text{CPA}}]$.

2.2 The Decoder Security Assumption

By decoder, say we denote it $\mathcal{D}$, we mean a family of decoding algorithms which can be applied to QC-MDPC codes corresponding to various security levels λ, including the three levels above, and to any block size $r_\lambda^{\text{CPA}}/2 \leq r \leq 2r_\lambda^{\text{CPA}}$. For a given security level λ, corresponding to a value of (w,t), we will denote $\text{DFR}_{\mathcal{D},\lambda}(r)$ the decoding failure rate when the decoder $\mathcal{D}$ is applied to an instance of $(2r, r, w, t)$-QC-MDPC-McEliece.

Assumption 3 *For a given decoder $\mathcal{D}$, and a given security level λ, the function $r \mapsto \log(\text{DFR}_{\mathcal{D},\lambda}(r))$ is decreasing and is concave if $\text{DFR}_{\mathcal{D},\lambda}(r) \geq 2^{-\lambda}$.*

2.3 Validity of the Concavity Assumption

Error Floors for QC-MDPC. The mapping $r \mapsto \log(\text{DFR}_{\mathcal{D},\lambda}(r))$ cannot be concave in the whole range $r \in [0, \infty[$. As explained in appendix, there is an additive term $P_\lambda(r)$ in $\text{DFR}_{\mathcal{D},\lambda}(r)$, coming from the code weight distribution, whose logarithm is asymptotically equivalent to $C_\lambda - (w/2-1)\log_2 r$. This term will dominate when r grows but only for very large values of r. We have

$$\lambda = 128, \log_2 P_\lambda(r_\lambda^{\text{CPA}}) = -396.8, \text{ and } \log_2 P_\lambda(r) \approx 535.0 - 70\log_2 r$$
$$\lambda = 192, \log_2 P_\lambda(r_\lambda^{\text{CPA}}) = -618.5, \text{ and } \log_2 P_\lambda(r) \approx 837.8 - 102\log_2 r$$
$$\lambda = 256, \log_2 P_\lambda(r_\lambda^{\text{CPA}}) = -868.7, \text{ and } \log_2 P_\lambda(r) \approx 1171.2 - 136\log_2 r$$

and this will not affect the DFR for values of r relevant for Assumption 3.

Theoretical Models for the Decoder. In [21] A Markovian model is given for a simple variant of bit flipping, the step-by-step decoder. This decoder corrects less errors than other bit flipping variants, however it uses the same ingredients: computing counters and flipping the corresponding positions if they are above

some threshold. The model can be computed for arbitrary large values of r and we observe that in the range of interest for r the log(DFR) is first strictly concave and eventually decreases linearly with r. This observation is consistent with Assumption 3. Note that the model does not capture the contribution of the weight distribution to the DFR.

Another work explores the asymptotic behavior of QC-MDPC decoding [22]. The asymptotic formula it provides for the DFR cannot be used directly because the setting is different (w and t vary with r), and also the conditions under which it can be proven are not relevant for decoders and parameters of practical interest. However the indication provided by the formula is consistent, the dominant term in the exponent decreases linearly with r.

To conclude this section, the Assumption 3 is and remains an assumption in the current state-of-the-art. We point out though that, for all variants of bit flipping decoding, every related theoretical and simulation results are consistent with it.

3 Backflip: a New Decoder for QC-MDPC Codes Using Reliability

Design Rationale: Positions with higher counters in Algorithm 1 have higher probabilities to be erroneous. Positions are flipped when the counter is above a threshold, how much above doesn't matter and a part of the reliability information is lost. Better performance are achieved with soft-decision decoders such as the belief propagation algorithm for LDPC codes. These decoders work by propagating probabilities back and forth between variable nodes and check nodes in the Tanner graph until the confidence on all values is high enough. Their logic and arithmetic are more complex though. See [2, 14] for examples of soft-decision MDPC decoding. The idea of Backflip is to use the reliability information while keeping the simplicity of the bit flipping decoder.

Among the flip decisions, most are good (an error is removed) and some are bad (an error is added). Bad decisions tend to induce more bad decisions and may lead to a failure. To exploit the reliability information a decoder could lessen the impact of the least reliable decisions and strengthen the impact of the most reliable ones. We propose Backflip, a new bit flipping algorithm which uses time to leverage the reliability information given by the counters on each flip. Every flip gets a (finite) time-to-live (an iteration count). When its time is over, the flip is canceled. Positions with a higher counter stay flipped for a longer time than positions with a counter just above the threshold. The design of Backflip is based on the following principles:

- the most reliable decisions will have more influence in the decoding process,
- all bad decisions will be cancelled at some point,
- conservative threshold selection hinders bad decisions in cascade.

In addition, it is readily seen that, compared to Algorithm 1, the Algorithm 2 only requires a few more operations (in blue) to manage a delay table D. Moreover, as

for the `threshold`, the `ttl` is very well approximated by an affine fonction for any fixed set of parameters and its computation has a negligible cost in practice.

Algorithm 2 Backflipping Algorithm

Require: $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$, $\mathrm{s} \in \mathbb{F}_2^{n-k}$, integer $u \geq 0$ *//$u > 0$ for noisy syndrome*
Ensure: $\left|\mathrm{s} - \mathrm{e}'\mathbf{H}^\intercal\right| \leq u$ or time $>$ max_time
  $\mathrm{e}' \leftarrow \mathbf{0}$; time $\leftarrow 1$; D $\leftarrow \mathbf{0}$ *//D_j = time-of-death of j*
  while $\left|\mathrm{s} - \mathrm{e}'\mathbf{H}^\intercal\right| > u$ **and** time $\leq$ max_time **do** *//here* max_time *is 100, 10 or 11*
    for j such that $D_j = \text{time}$ **do** $e'_j \leftarrow 0$ *//Undo flips at time-of-death*
    time $\leftarrow$ time $+ 1$
    $\mathrm{s}' \leftarrow \mathrm{s} - \mathrm{e}'\mathbf{H}^\intercal$
    $T \leftarrow \texttt{threshold}(|\mathrm{s}'|, t - |\mathrm{e}'|)$
    for $j \in \{0, \ldots, n-1\}$ **do**
      if $|\mathrm{s}' \cap \mathrm{h}_j| \geq T$ **then** *//h_j the j-th column of* $\mathbf{H}$
        $e'_j \leftarrow 1 - e'_j$; $D_j \leftarrow \text{time} + \texttt{ttl}(|\mathrm{s}' \cap \mathrm{h}_j| - T)$
  return e'

Threshold Selection Rule `threshold`(S, t'). As the time-to-live of a flip is always finite, a bad flip will always be canceled eventually. However, it is necessary to avoid adding more bad flips during the period during which it remains flipped. To achieve this, thresholds from Figure 1 are used with $S = |\mathrm{s}'|$ and $t' = t - |\mathrm{e}'|$. This is the best case estimate for the error weight, it supposes that every flip removed an error. When many errors were added, the corresponding threshold is higher than for the usual bit flipping algorithm, this will slow down the decoding process, leaving time to cancel the bad decisions while making only very reliable new flips. In the typical case, most flip decisions were good, the threshold is close to optimal, and the decoding converges quickly.

Time-to-live Rule `ttl`(δ). Empirically, it appears that the time-to-live should be increasing with the difference δ between the position's counter and the iteration threshold. It should also be finite because otherwise outlier counter values could lead to adding errors that are harder to detect: correct positions with a high counter will become errors with a low counter once flipped, their counter will have to change drastically before it is corrected by an algorithm relying solely on a threshold. The `ttl` function depends on the code parameters (especially w and t) as well as the maximum number of iterations of the decoder. In practice, a saturating affine function in δ can be used.

$$\texttt{ttl}(\delta) = \max(1, \min(\text{max\_ttl}, \lfloor \alpha\,\delta + \beta \rfloor))\,.$$

To determine a suitable function, w, t, and the number of iterations are fixed. The block size r is chosen so that a sufficiently precise measure of the DFR can be made with a reasonable number of samples ($\approx 10^8$). A nonlinear optimization

method (such as Nelder-Mead's) is then used to find values for α and β that minimize the DFR.

Iteration count	λ	(w,t)	(α,β)	max_ttl
100	128	$(142,134)$	$(0.45,1.1)$	5
	192	$(206,199)$	$(0.36,1.41)$	5
	256	$(274,264)$	$(0.45,1)$	5
10,11	128	$(142,134)$	$(1,1)$	5
	192	$(206,199)$	$(1,1)$	5
	256	$(274,264)$	$(1,1)$	5

Table 2. `ttl` function parameters

Complexity and Constant Time Implementation. Backflip was primarily designed to work with a maximum of 100 iterations. Reducing this number to 10 is possible and requires an adjustment to the `ttl` function. However it increases significantly the estimated DFR (see §4). Nevertheless, in both cases, the average number of iterations is much smaller, between 2.03 for $(r,w,t) = (24821,206,199)$ and 4.38 for $(r,w,t) = (32749,274,264)$.

The interest of reducing max_time is to allow constant time implementation. The Backflip iteration can be implemented in constant time [7], but to mask the effective number of iterations and keep the DFR claims, the algorithm has to execute exactly max_time iterations.

4 Estimating the DFR from Simulation

Under Assumption 3 for a decoder $\mathcal{D}$ and a security level λ, we may extrapolate the DFR by accurately estimating the tangent of the function $r \mapsto \log_2(\mathrm{DFR}(r))$ for some value of r. We obtain an estimate of the tangent by taking the line joining the values for two points $r_1 < r_2$. Note that, except for a possible lack of accuracy (discussed below), this will provide upper bounds for the extrapolated DFRs. Results are presented in Table 3, we denote $r_{\mathcal{D},\lambda}$ the smallest r such that $\mathrm{DFR}_{\mathcal{D},\lambda}(r) \leq 2^{-\lambda}$. We denote r_λ^{CPA} and r_λ^{CCA} the blocks sizes in BIKE for CPA and CCA security. Known asymptotic analysis [21, 22] indicate that the $\log_2(\mathrm{DFR})$ is ultimately decreasing linearly, but this linear regime probably starts much beyond the simulated region. Thus it is best to choose r_1, r_2 as large as possible, but not too large else we would decrease the accuracy.

Finally note that a significant computational effort was needed to compute the data of Table 4, a total of several years of CPU time (on a single core).

Accurary of Simulated DFRs. The decoding failure is a Bernoulli trial of probability p. If we observe F failures out of N trials our estimate is $\hat{p} = F/N$. The

#iter	λ	r_1	r_2	$\log_2(p_1)$	$\log_2(p_2)$	$r_{\mathcal{D},\lambda}$	r_λ^{CCA}	$\log_2(r_\lambda^{\mathrm{CCA}})$	r_λ^{CPA}	$\log_2(r_\lambda^{\mathrm{CPA}})$
	128	9200	9350	-21.4	-27.7	11717	11779	-130.7	10163	-62.2
100	192	18200	18300	-23.0	-25.6	24665	24821	-196.1	19853	-66.2
	256	30250	30400	-23.3	-26.2	42418	40597	-221.2	32749	-71.1
	128	10000	10050	-22.7	-24.6	12816	11779	-89.2	10163	-28.8
10	192	19550	19650	-23.5	-25.7	26939	24821	-143.7	19853	-30.4
	256	32250	32450	-22.9	-26.6	44638	40597	-180.0	32749	-32.3
	128	10000	10050	-25.1	-27.1	12573	11779	-96.3	10163	-31.6
11	192	19550	19650	-25.9	-28.6	25580	24821	-171.1	19853	-34.2
	256	32250	32450	-25.1	-29.5	42706	40597	-209.4	32749	-36.1

Table 3. DFR estimation for Backflip limited to max_time iterations.

normal distribution gives a good approximation of this distribution in which the standard deviation for F is $\sqrt{p(1-p)N}$. For $p \ll 1$ (the case of interest) we have $\left|\frac{\hat{p}-p}{p}\right| \leq \varepsilon = z/\sqrt{pN}$ with probability $1-\alpha \approx 0.68, 0.95, 0.997$ for $z = 1, 2, 3$ respectively. We observe that the precision decreases as $z/\sqrt{F}$ where F is the number of failures observed and z will be determined by the confidence we wish to achieve. Note that for the same confidence, $|\log \hat{p} - \log p| \leq \varepsilon$. In our case, we use Clopper–Pearson intervals [6] which are exact (they use the correct binomial distribution and not an approximation). Those intervals are not symmetric, the confidence interval is ε^- below and ε^+ above the measured values. In the

#iter	λ	r_1	F_1	N_1	$\log_2 p_1$	ε^-	ε^+	r_2	F_2	N_2	$\log_2 p_2$	ε^-	ε^+
	128	9200	1253	$3.45\,10^9$	-21.4	0.107	0.104	9350	102	$2.30\,10^{10}$	-27.7	0.390	0.361
100	192	18200	499	$4.13\,10^9$	-23.0	0.171	0.165	18300	90	$4.57\,10^9$	-25.6	0.416	0.383
	256	30250	282	$2.96\,10^9$	-23.3	0.229	0.219	30400	80	$6.14\,10^9$	-25.3	0.443	0.407
	128	10000	1074	$7.29\,10^9$	-22.7	0.115	0.113	10050	282	$6.99\,10^9$	-24.6	0.229	0.219
10	192	19550	440	$5.08\,10^9$	-23.5	0.182	0.176	19650	81	$4.55\,10^9$	-25.7	0.440	0.404
	256	32250	513	$3.91\,10^9$	-22.9	0.168	0.163	32450	37	$3.83\,10^9$	-26.6	0.673	0.591
	128	10000	200	$7.29\,10^9$	-25.1	0.274	0.259	10050	48	$6.99\,10^9$	-27.1	0.584	0.522
11	192	19550	83	$5.08\,10^9$	-25.9	0.435	0.399	19650	11	$4.55\,10^9$	-28.6	1.348	1.054
	256	32250	109	$3.91\,10^9$	-25.1	0.376	0.350	32450	5	$3.83\,10^9$	-29.5	2.214	1.501

Table 4. Raw simulation data with confidence intervals ($\alpha = 0.01$)

simulation for max_time = 10 we let the decoder run up to 50 iterations and store the number of effective iterations. We are thus able to measure the DFR for 11 iterations of Backflip. We observe in Table 3 a significant improvement in the DFR, but a lower confidence (Table 4) because the block sizes were chosen for 10 iterations. Nevertheless, this suggests that increasing max_time could

provide interesting trade-offs between complexity and DFR for constant time implementations.

Additional Comments. In [21], the BIKE round 1 algorithm was estimated to have a DFR around $2^{-47.5}$ for $(r, w, t) = (32\,749, 274, 264)$. A significant improvement is made with Backflip as its DFR is estimated around $2^{-71.1}$ for the same parameters, with a smaller complexity on average.

Finally, note that the suggested parameters for the CCA variant of BIKE Level 5 ($\lambda = 256$) have not been correctly estimated. The extrapolated block size to reach a DFR of 2^{-256} is 42418 rather than 40597 in [1]. This is due to the imprecision of the measures at the time. To mitigate this issue, it is very likely that the tangent we are using is pessimistic and that the actual DFR is much lower than the extrapolated value given here.

5 Conclusion

We have given in this paper the description and the rationale of the Backflip decoder of BIKE [1]. We also explain how the DFR claims were obtained by extrapolating simulation data. To justify the extrapolation technique we introduce a new security assumption, related to the decoder, under which the DFR claims are valid. The assumption is supported by other works analyzing the asymptotic behavior of the bit flipping decoding for QC-MDPC codes. Under this additional assumption, it is possible to prove that the BIKE KEMs, derived from QC-MDPC codes, are IND-CCA. Doing this requires extensive simulations in order to obtain accurate simulation data.

Backflip with 100 iterations would hardly produce efficient constant time implementations. Reducing the number of iterations to 10 increases the DFR and would require larger block size to reach a low enough DFR for IND-CCA security. This was remarked in another independent work [7] which considers another variant of the bit flipping algorithm, closer to the round 1 BIKE decoder, and which is more efficient when the number of iterations is bounded to a small number. The methodology we develop here is valid for other variants of bit flipping and can be used to justify the conclusions of [7]: we may produce efficient constant time variants of BIKE with provably low DFR (under Assumption 3) but it requires a small increase of the block size, in the order of 5% to 10%.

Finally, there is one extra feature of the tangent extrapolation technique. With a larger amount a computational effort for the simulation, it should be possible, under the same assumptions, to get the same security guaranty (*e.g.* IND-CCA) for a smaller block size.

References

1. Carlos Aguilar Melchor, Nicolas Aragon, Paulo Barreto, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Shay Gueron, Tim Güneysu, Rafael Misoczki, Edoardo Persichetti, Nicolas Sendrier, Jean-Pierre Tillich, and Gilles Zémor. BIKE. Second round submission to the NIST post-quantum cryptography call, April 2019.
2. Marco Baldi, Paolo Santini, and Franco Chiaraluce. Soft mceliece: MDPC code-based mceliece cryptosystems with very compact keys through real-valued intentional errors. In *Proc. IEEE Int. Symposium Inf. Theory - ISIT*, pages 795–799. IEEE Press, 2016.
3. Anja Becker, Antoine Joux, Alexander May, and Alexander Meurer. Decoding random binary linear codes in $2^{n/20}$: How $1 + 1 = 0$ improves information set decoding. In *Advances in Cryptology - EUROCRYPT 2012*, LNCS. Springer, 2012.
4. Rodolfo Canto-Torres and Nicolas Sendrier. Analysis of information set decoding for a sub-linear error weight. In *Post-Quantum Cryptography 2016*, LNCS, pages 144–161, Fukuoka, Japan, February 2016.
5. Julia Chaulet. *Étude de cryptosystèmes à clé publique basés sur les codes MDPC quasi-cycliques*. PhD thesis, University Pierre et Marie Curie, March 2017.
6. C. J. Clopper and E. S. Pearson. The Use of Confidence or Fiducial Limits Illustrated in the case of the Binomial. *Biometrika*, 26(4):404–413, 12 1934.
7. Nir Drucker, Shay Gueron, and Dusan Kostic. On constant-time QC-MDPC decoding with negligible failure rate. Cryptology ePrint Archive, Report 2019/1289, 2019. https://eprint.iacr.org/2019/1289.
8. Edward Eaton, Matthieu Lequesne, Alex Parent, and Nicolas Sendrier. QC-MDPC: A timing attack and a CCA2 KEM. In *Post-Quantum Cryptography - 9th International Conference, PQCrypto 2018, Fort Lauderdale, FL, USA, April 9-11, 2018, Proceedings*, pages 47–76, 2018.
9. Jean-Bernard Fischer and Jacques Stern. An efficient pseudo-random generator provably as secure as syndrome decoding. In Ueli Maurer, editor, *Advances in Cryptology - EUROCRYPT'96*, volume 1070 of *LNCS*, pages 245–255. Springer, 1996.
10. Robert G. Gallager. *Low Density Parity Check Codes*. M.I.T. Press, Cambridge, Massachusetts, 1963.
11. Qian Guo, Thomas Johansson, and Paul Stankovski. A key recovery attack on MDPC with CCA security using decoding errors. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology - ASIACRYPT 2016*, volume 10031 of *LNCS*, pages 789–815, 2016.
12. Yann Hamdaoui and Nicolas Sendrier. A non asymptotic analysis of information set decoding. IACR Cryptology ePrint Archive, Report2013/162, 2013. http://eprint.iacr.org/2013/162.
13. Stefan Heyse, Ingo von Maurich, and Tim Güneysu. Smaller keys for code-based cryptography: QC-MDPC McEliece implementations on embedded devices. In Guido Bertoni and Jean-Sébastien Coron, editors, *Cryptographic Hardware and Embedded Systems - CHES 2013*, volume 8086 of *LNCS*, pages 273–292. Springer, 2013.
14. Gianluigi Liva and Hannes Bartz. Protograph-based quasi-cyclic MDPC codes for mceliece cryptosystems. In *ISTC*, pages 1–5, Hong Kong, China, December 2018. IEEE.

15. David J. C. MacKay and Michael S. Postol. Weaknesses of margulis and ramanujan-margulis low-density parity-check ccodes. *Electr. Notes Theor. Comput. Sci.*, 74:97–104, 2002.
16. Robert J. McEliece. *A Public-Key System Based on Algebraic Coding Theory*, pages 114–116. Jet Propulsion Lab, 1978. DSN Progress Report 44.
17. Rafael Misoczki, Jean-Pierre Tillich, Nicolas Sendrier, and Paulo S. L. M. Barreto. MDPC-McEliece: New McEliece variants from moderate density parity-check codes. In *Proc. IEEE Int. Symposium Inf. Theory - ISIT*, pages 2069–2073, 2013.
18. Eugene Prange. The use of information sets in decoding cyclic codes. *IRE Transactions on Information Theory*, 8(5):5–9, 1962.
19. Tom Richardson. Error floors of LDPC codes. In *Proc. of the 41th Annual Allerton Conf. on Communication, Control, and Computing*, 2003.
20. Nicolas Sendrier. Decoding one out of many. In *Post-Quantum Cryptography 2011*, volume 7071 of *LNCS*, pages 51–67, 2011.
21. Nicolas Sendrier and Valentin Vasseur. On the decoding failure rate of QC-MDPC bit-flipping decoders. In Jintai Ding and Rainer Steinwandt, editors, *Post-Quantum Cryptography 2019*, volume 11505 of *LNCS*, pages 404–416, Chongquing, China, May 2019. Springer.
22. Jean-Pierre Tillich. The decoding failure probability of MDPC codes. In *2018 IEEE International Symposium on Information Theory, ISIT 2018, Vail, CO, USA, June 17-22, 2018*, pages 941–945, 2018.

A Error Floors For QC-MDPC

The DFR study we are making here differs from what is done for communication systems where the code is fixed and the signal to noise ratio increases (*i.e.* the bit error probability decreases). We expect to observe the same kind of DFR behavior here for QC-MDPC when we fix (w, t) and let r grow. Some classes of error correcting codes, namely turbo-codes and LDPC codes to which MDPC codes are akin, suffer from a phenomenon known as error floor. The log(DFR) curve is first concave and quickly decreasing (the *waterfall*). Then at some point the concavity changes and the DFR decreases much more slowly, this is known as the *error floor* [15, 19]. This could contradict the Assumption 3, but fortunately error floors usually occur very low in DFR curves. The error floors are due to the existence of low weight codewords, in the case of turbo codes, or, for LDPC codes, to the existence of specific error configurations known as *near-codewords*. An (u, v)-near-codeword is an error pattern of relatively small weight u with a syndrome of small weight v (the syndrome is computed with the sparse parity check matrix). Intuitively, it can be seen as a cluster of errors which are less visible because, together, they only invalidate a few parity equations. If the initial error pattern contains a near-codeword the decoder is more prone to fail. If many near-codewords exist it may cause an error floor.

Error Floors From Near-Codewords. To affect decoding in a $(2r, r, w, t)$-QC-MDPC-McEliece scheme, an (u, v)-near-codewords (see definition above) must be such that u is smaller than t, and v significantly smaller than the typical

syndrome weight. The probability that such a near-codeword exists when the QC-MDPC is chosen at random is extremely small. A very small number of QC-MDPC codes may admit such words, but if they do there will be few of them. Moreover, the decoding of the few error patterns containing near-codewords will not automatically fail, the DFR will just increase a bit, with little impact on the average DFR. Unless there is an algebraic structure which is not immediately apparent, we do not expect near-codewords to have an impact on QC-MDPC DFR.

Error Floors From Low Weight Codewords. Regardless of the algorithm, the decoding of a noisy codeword will almost certainly fail if the noisy codeword comes closer to a codeword c_1 different from the original one c_0. For a given error e of weight t, and two codewords c_0 and c_1 at distance w from one another, the decoding will fail if $|\mathrm{c}_0 + \mathrm{e} - \mathrm{c}_1| \leq |e|$, which happens with probability

$$P_w = \sum_{i=w/2}^{w} \frac{\binom{w}{i}\binom{n-w}{t-i}}{\binom{n}{t}}. \tag{1}$$

An index 2 QC-MDPC code with block size r and parity check matrix row weight w will generally have exactly r codewords of weight w. If $\mathbf{H} = (\mathbf{H}_0 \mid \mathbf{H}_1)$ is the sparse parity check matrix, with two circulant blocks $\mathbf{H}_0, \mathbf{H}_1$, then $\mathbf{G} = (\mathbf{H}_1^\intercal \mid \mathbf{H}_0^\intercal)$ is a generator matrix of the code. With overwhelming probability, the r rows of that generator matrix are the only minimal weight codewords. Let us denote $P_\lambda(r) \approx rP_w$ the failure probability due to those codewords. A simple analysis shows that $\log_2 P_\lambda(r) \sim_{r\to\infty} C_\lambda - (w/2-1)\log_2 r$ where C_λ only depends of w and t. We have $\mathrm{DFR}_{\mathcal{D},\lambda}(r) \geq P_\lambda(r)$ for any decoder, this term will dominate when r grows and thus the logarithm of the DFR is *not concave* in the whole range $r \in [0, \infty[$. However the change of slope only happens for very large values of r. We have

$$\begin{aligned}
&\lambda = 128, \log_2 P_\lambda(r_\lambda^{\mathrm{CPA}}) = -396.8, \text{ and } \log_2 P_\lambda(r) \approx 535.0 - 70\log_2 r\\
&\lambda = 192, \log_2 P_\lambda(r_\lambda^{\mathrm{CPA}}) = -618.5, \text{ and } \log_2 P_\lambda(r) \approx 837.8 - 102\log_2 r\\
&\lambda = 256, \log_2 P_\lambda(r_\lambda^{\mathrm{CPA}}) = -868.7, \text{ and } \log_2 P_\lambda(r) \approx 1171.2 - 136\log_2 r
\end{aligned}$$

and this will not affect the DFR for values of r relevant for Assumption 3. Finally note that the sum of two (or more) rows of $\mathbf{G}$ may also contribute to the DFR. However, it is easily observed that the contribution of those codewords is even smaller.