QC-MDPC (BIKE) Failure Analysis Survey

Ray Perlner

Overview

- BIKE is a code based KEM and a 3rd round candidate in the NIST PQC standardization process
- It uses the Niederreiter variant of the McEliece Construction, with a QC-MDPC code
  - Alternately, this could be viewed as the NTRU construction with Hamming metric
- Unlike Goppa McEliece, QC-MDPC McEliece has a decoder that sometimes fails
- In order to get IND-CCA security for up to 2^{64} queries, the failure rate must be very low
- The BIKE team's best estimates of the failure rate for BIKE's parameters is low enough
  - But are the estimates correct?
  - The BIKE team does not claim their estimates are correct, and therefore only claims IND-CPA security
  - Can we do more to confirm or disconfirm

Some Coding Theory

- Generator matrix (Systematic form)
  - $n \times k$

$$G = [I_k \mid C]$$

- Parity Check matrix (Systematic form)
  - $n \times (n-k)$

$$H = [-C^T \mid I_{n-k}]$$

- **Defining feature: $HG^T = 0$**
- Codewords x may either be defined as
  - n-bit vectors that can be expressed as $x = mG$ for k-bit m
  - Solutions to $Hx^T = 0$

- Syndrome:

$$s = H(mG + e)^T \overset{=}{} H(e^T)$$

  - Mapping s to minimal weight e is sometimes easy but NP hard in general.

- McEliece Encryption: $mG + e$ is ciphertext, m is plaintext.
- Niederreiter Encryption: s is ciphertext, e is plaintext.
  - Note: Both "McEliece" and Niederreiter KEMs for BIKE use Hash(e) as shared secret.

Quasi-Cyclic Structure

- Use $n = 2r$; $k = r$, where r is prime and $x^r - 1$ is $(x-1)$ times a primitive polynomial mod 2.
- Represent $\mathrm{r} \times r$ blocks as polynomials in the ring $\mathrm{GF2}[x]/x^r - 1$.
  - Now block multiplication commutes.
  - And blocks only require k bit representation.
  - They look like this:

$$\begin{pmatrix} a & b & c & d & e & f \\ f & a & b & c & d & e \\ e & f & a & b & c & d \\ d & e & f & a & b & c \\ c & d & e & f & a & b \\ b & c & d & e & f & a \end{pmatrix}$$

BIKE Construction (Niederreiter)

- Public key: Blockwise cyclic *parity check* matrix $\mathrm{H} = \begin{pmatrix} I & h \end{pmatrix}$
  - $h = x^{-1}y$; (x, y) is "short" (weight w) in *Hamming metric*
- Ciphertext: $\mathrm{c} = He^T = e_0 + he_1$
  - (e_0, e_1) is "short" (weight t) in *Hamming metric*
- Decoding
  - Private key allows decoding of $xc = xe_0 + ye_1 = \begin{pmatrix} x & y \end{pmatrix} \begin{pmatrix} e_0 \\ e_1 \end{pmatrix}$ using
    - Bit flipping algorithm
  - If decoding succeeds, use (e_0, e_1) to derive a shared secret
  - Combine with Fujisaki-Okamoto transform for CCA security, if failure rate is low enough

BIKE Parameters

Security	r	w	t	DFR
Level 1	12,323	142	134	2^{-128}
Level 3	24,659	206	199	2^{-192}

- Note:
  - w and t are approximately equal to the security parameter
  - $r \approx \frac{wt}{2}$
  - To lower the DFR, increase r, while fixing w, t

Bit-Flipping Decoder

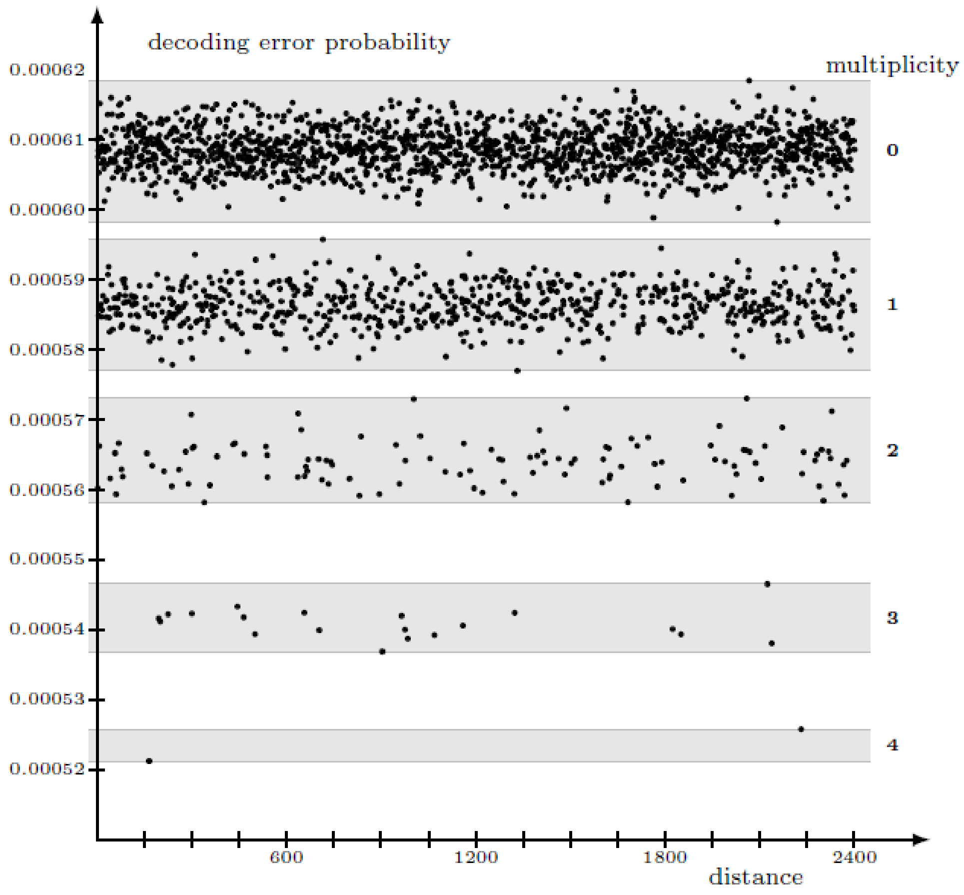
- We want to solve $(x \quad y)\begin{pmatrix} e_0 \\ e_1 \end{pmatrix} = s$ for $\begin{pmatrix} e_0 \\ e_1 \end{pmatrix}$
  - Think of x, y as matrices
- Since $\begin{pmatrix} e_0 \\ e_1 \end{pmatrix}$ is t-sparse, s is the sum of t columns of $(x \quad y)$
  - Each column has weight $\frac{w}{2}$
  - Since $\frac{wt}{2} < r$, not many bits cancel
  - So the columns in the sum (with same index as nonzero bits of $\begin{pmatrix} e_0 \\ e_1 \end{pmatrix}$) share a lot of nonzero bits with s
- Iterated algorithm to decode
  - Guess that the columns with a lot of 1s in common with s are nonzero bits of $\begin{pmatrix} e_0 \\ e_1 \end{pmatrix}$
  - Subtract off the syndrome corresponding to the guess from both sides of $(x \quad y)\begin{pmatrix} e_0 \\ e_1 \end{pmatrix} = s$
  - Resulting in $(x \quad y)\begin{pmatrix} {e_0}' \\ {e_1}' \end{pmatrix} = s'$
  - If $s' = 0$, you're done. Otherwise, try to decode s' same way as s

Other Decoders

- Many variants of the basic bit flipping decoder have been proposed
  - Backflip decoder (2nd round) [Sendrier, Vasseur 2019]
  - Black-Grey-Flip (BGF) decoder (pre-3rd round) [Drucker, Gueron, Kostic 2019]
- Usually the motivation is a lower decryption failure rate
- However, all decoders work on a similar principle to bit flipping
- No proposed parameter set claims a zero decryption failure
  - As we'll see later, the DFR cannot be 0
- Why are decryption failures bad?

GJS attack
[Guo, Johansson, Stankovski 2016]

- The bit flipping decoder doesn't always work
- Ciphertexts/error vectors that induce failures give statistical info about private key
  - Error vectors where there are bits the same distance apart as two bits of the private key are LESS likely to induce a decoding failure
  - Lists of distances between pairs of nonzero coefficients in each of x, y, e_0, e_1 are called distance spectra
- Can recover a key with ~100,000 known *decryption failures*
- Interesting tidbit: If ALL of the nonzero coefficients of $\binom{e_0}{e_1}$ are in e_0 or e_1 the DFR is higher. Too rare to use?



Error Amplification [Nilsson, Johansson, Wagner 2018]

- Builds upon distance spectrum idea from GJS attack
- Can use a known decryption failure to build other ciphertexts more likely to produce a decryption failure
  - This means that the majority of the cost of the attack involves finding the first decryption failure
- If the number of iterations to used to decode is variable, that side channel information can be used to speed up finding the first decryption failure
  - We think BIKE’s current implementation is constant time, and if we discover it isn’t, we will expect them to fix it

Bounding the Error Rate

- When the DFR is high, e.g. 2^{-35} or more, it can be directly measured
- But, to protect against known attacks, need DFR$< 2^{-64}$, and to prove security need DFR$< 2^{-128}$ (Cat 1) or DFR$< 2^{-192}$ (Cat 3)
- How do we know when we reach these targets if we cannot directly measure the DFR?
  - LEDAcrypt's approach was to use really conservative parameters (~50% larger key sizes than BIKE) and get a loose upper bound on the DFR
    - But they had other problems (attacked and patched in 2^{nd} round, due to extra structure removed in patch)
  - BIKE derives a curve to fit the relation between the parameter r and the DFR from a Markov model with simplified asssumptions, and extrapolates

Markov Model and Extrapolation
[Sendrier, Vasseur 2018], [Sendrier, Vasseur 2019], [Drucker, Gueron, Kostic 2019]

- Simulate a simplified decoding algorithm using a Markov model
  - Drawbacks:
    - Doesn’t analyze actual BIKE decoder, but one which is simpler and empirically less efficient
    - Treats bits of syndrome as independent random variables
    - Treats any weight t decoding as a success
- Uses the derived form (exponential in r^2 up to a critical value, then exponential in r, no kink) to extrapolate DFR assuming critical point is as soon as it can be
  - That this extrapolation works “long enough” is called concavity assumption in [Sendrier, Vasseur 2019]
  - Error floors suggest it can’t work forever, but might work long enough

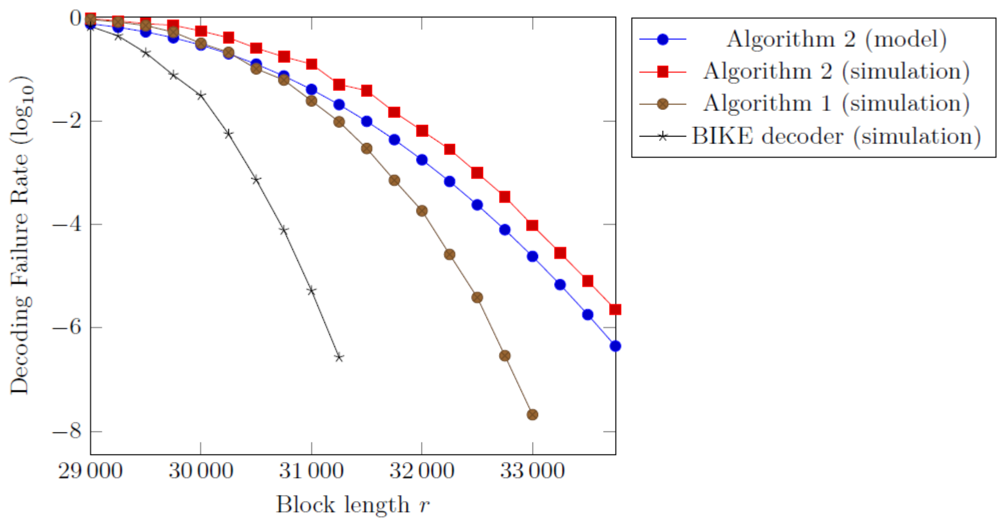


Figure 1. DFR of the step-by-step algorithm in the models and from simulations (infinite number of iterations)

Error Floors

Image on right from [Richardson] illustrating similar phenomenon in related code to BIKE

- Note that $(x \quad y)\begin{pmatrix} y \\ x \end{pmatrix} = 0$
- This means $\begin{pmatrix} y \\ x \end{pmatrix}^T$ is a valid generator matrix for the QC-MDPC code underlying BIKE, and its rows are codewords of weight w
- If the error pattern shares at least w/2 one bits with a short codeword, there is another codeword with the same syndrome, and decryption must fail with at least 50% probability
- For category 1 BIKE parameters this means the DFR is at least 2^{-346}

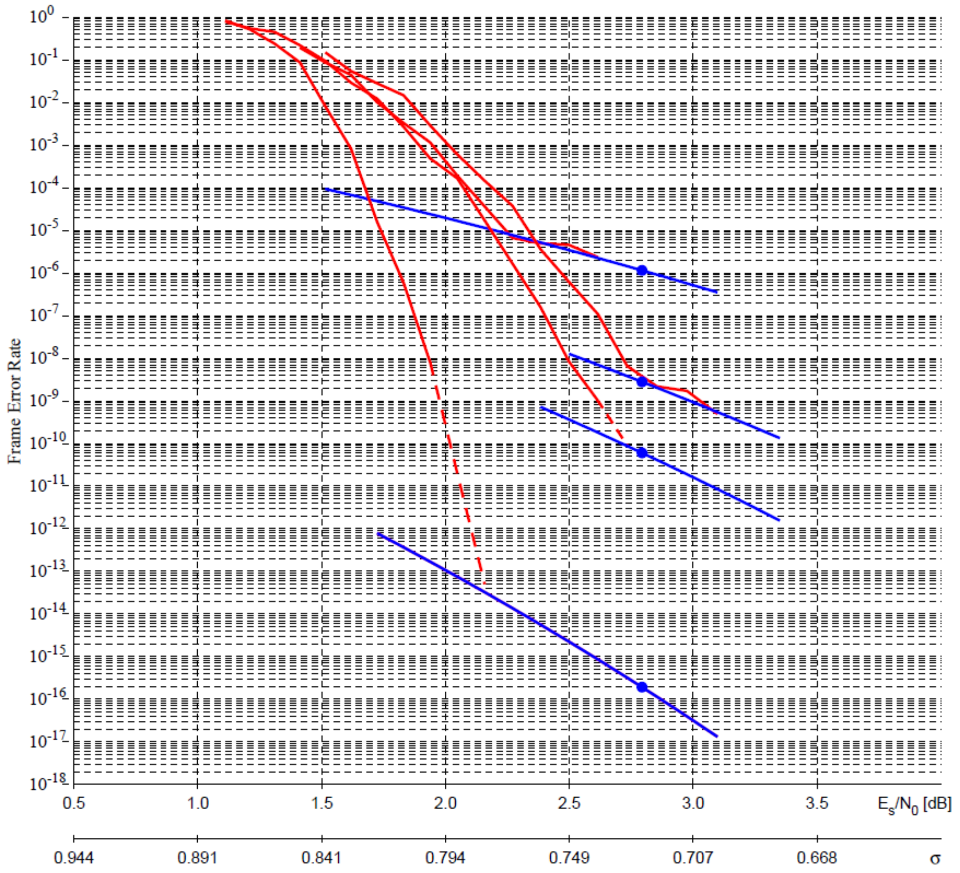


Figure 1: Simulation and error floor predictions for some regular (3,6) LDPC codes using a 5-bit decoder. The codes in order from highest to lowest error floor are the Margulis graph (n=2640), an n=2048 code, an n=2640 code (same as the Margulis graph), and an n=8096 code. The dashed curves are extrapolations. Except for the Margulis code, code simulations were performed on an FPGA platform. Error floor predictions are computed on a PC.

Directions for Future Work

- Can we get a tighter proven bound on DFR , preferably with a more efficient decoder, than what the LEDAcrypt team did
  - Maybe some kind of computer aided proof?
  - This is the dream, but I have no idea how to do it
- Can we test the convexity assumption with the real decoder
  - Use parameters targeting a low security parameter, so we can directly measure error floor area
  - Use normal, or intermediate parameters, but amplify error floor phenomenon by choosing error vectors near (but not within $t - w/2$ of) known, short codewords

References

- Latest BIKE spec. https://bikesuite.org/files/v4.0/BIKE_Spec.2020.05.03.1.pdf
- Reaction attack
  - Guo, Johansson, Stankovski 2016: https://eprint.iacr.org/2016/858.pdf
- Error Amplification
  - Nilsson, Johansson, Wagner 2018: https://eprint.iacr.org/2018/1223.pdf
- Estimating/bounding failure rate
  - Markovian analysis (Simple decoder, infinite iterations)
    - [Sendrier, Vasseur 2018]: https://eprint.iacr.org/2018/1207
  - Extrapolation (Backflip decoder)
    - [Sendrier, Vasseur 2019]: https://eprint.iacr.org/2019/1434.pdf
  - Extrapolation (Black and Gray Decoder)
    - [Drucker, Gueron, Kostic 2019]: https://eprint.iacr.org/2019/1423
  - Explicit bounds for 1 (tight) or 2 (loose) iterations (IR BF Decoder)
    - [Baldi et al. 2020] https://re.public.polimi.it/retrieve/handle/11311/1144467/513367/SECRYPT_2020_118_CR.pdf
- Describing error floors (I don't think this paper originated the idea):
    - [Richardson]: https://web.stanford.edu/class/ee388/papers/ErrorFloors.pdf