Bit-flipping Decoder Failure Rate Estimation for (v,w)-regular Codes

Alessandro Annechini, Alessandro Barenghi, Gerardo Pelosi
Fifth PQC Standardization Conference - 12th April 2024
Context: Code-based KEMs with iterative decoding

- Current 4th round candidate BIKE is built on sparse QC random codes (QC-MDPC)
  - QC-MDPCs are decoded with an iterative, fixed point procedure
  - Achieved DFR depends on both the code and the decoder choice
- Decoding failures reveal information on the private key, breaking IND-CCA2
  - Estimating DFR in closed-form has proven to be challenging
  - [WWW23]: estimates of DFR for BIKE w/ BGF decoder were optimistic

Contributions

1. Closed form estimate of avg. DFR for \((v, w)\)-regular codes w/ 2-iteration BF decoder
2. Analyze the code parameters for a IND-CCA2 QC-MDPC scheme

- Accepted at IEEE International Symposium on Information Theory (ISIT 2024)
(v, w)-regular and QC-MDPC binary codes

(v, w)-regular codes

- Binary block codes with length n, dimension k and redundancy n – k = r
- Each column h_{i,j} of the parity check matrix H has Hamming weight \(\text{wt}(h_{i,j}) = v\)
- Each row of h_{i,:} the parity check matrix H has Hamming weight \(\text{wt}(h_{i,:}) = w = \frac{n}{r}v\)

QC-MDPC codes

- Subset of (v, w)-regular codes with H defined tiling \(p \times p\) circulant matrices, \(v \approx \sqrt{n}\)
- Both BIKE and LEDAcrypt use \(n = n_0p, r = p\) codes, \(w = n_0v, p\) prime, \(\text{ord}_p(2) = p – 1\)
  - BIKE uses \(n_0 = 2\)
  - LEDAcrypt uses \(n_0 \in \{2, 3, 4\}\) codes
Iterative syndrome decoding: find $e$, given $H$ and $s = He^T$

Toy example: $n = 10$, $r = 5$, $v = 2$, $w = 4$, $wt(e) = 2$

<table>
<thead>
<tr>
<th>1 0 1 0 0 0 1 0 0 1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 0 1 0 1 0 1 0 0</td>
<td>1</td>
</tr>
<tr>
<td>0 0 1 0 1 0 1 0 1 0</td>
<td>1</td>
</tr>
<tr>
<td>1 0 0 1 0 0 0 1 0 1</td>
<td>1</td>
</tr>
<tr>
<td>0 1 0 0 1 1 0 0 1 0</td>
<td>0</td>
</tr>
</tbody>
</table>

p.c. matrix $H$

| 0 0 1 0 0 0 0 1 0 0 |

error $e$
Iterative BF decoder: initialization

\[ \text{iter} = 0 \text{ completed iter.s; invariant } s_{(\text{iter})} = H(e \oplus \bar{e}_{(\text{iter})})^T \]

### p.c. matrix H

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

### s(\text{iter})

|   |   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 |   |   |   |   |   |   |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |   |
| 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |   |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |   |

### error e


### error est. \( \bar{e}_{(\text{iter})} \)

0 0 0 0 0 0 0 0 0 0 0
UPC computation

\[ \text{iter} = 0 \] completed iter.s; invariant \( s_{(\text{iter})} = H(e \oplus \bar{e}_{(\text{iter})})^T \)

p.c. matrix \( H \)

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

error est. \( \bar{e}_{(\text{iter})} \)

|   | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

upc

|   | 2 | 1 | 2 | 2 | 1 | 1 | 2 | 1 | 1 | 2 |

\( s_{(\text{iter})} \)

|   | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |

\( e \)

|   | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |

|   | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
Flip $\bar{e}_{(\text{iter})j}$ if $\text{upc}_j \geq \text{th}$

$\text{iter} = 0$ completed iter.s; invariant $s_{(\text{iter})} = H(e \oplus \bar{e}_{(\text{iter})})^T$

<table>
<thead>
<tr>
<th>1 0 1 0 0 0 1 0 0 1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 0 1 0 1 0 1 0 0</td>
<td>1</td>
</tr>
<tr>
<td>0 0 1 0 1 0 1 0 1 0</td>
<td>1</td>
</tr>
<tr>
<td>1 0 0 1 0 0 0 1 0 1</td>
<td>1</td>
</tr>
<tr>
<td>0 1 0 0 1 1 0 0 1 0</td>
<td>0</td>
</tr>
</tbody>
</table>

p.c. matrix $H$

error est. $\bar{e}_{(\text{iter})}$

| 1 0 1 1 0 0 1 0 0 1 |

upc

| 2 1 2 2 1 1 2 1 1 2 |
Update $s$ as $s \oplus h_j$ if $\bar{e}_{(\text{iter})j}$ was flipped

$\text{iter} = 0$ completed iter.s; invariant $s_{(\text{iter})} = H(e \oplus \bar{e}_{(\text{iter})})^T$

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>P.c. matrix $H$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

|      | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |

Error est. $\bar{e}_{(\text{iter})}$
Increment \( \text{iter} \), if \( s_{(\text{iter})} = 0 \) \( \Rightarrow e \oplus \bar{e}_{(\text{iter})} = 0 \) return \( \bar{e}_{(\text{iter})} = 0 \)

\[
\text{iter} = 0 \text{ completed iter.s}; \text{ invariant } s_{(\text{iter})} = H(e \oplus \bar{e}_{(\text{iter})})^T
\]

p.c. matrix \( H \):

\[
\begin{array}{cccccccccc}
1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\
\end{array}
\]

error est. \( \bar{e}_{(\text{iter})} \):

\[
\begin{array}{cccccccccc}
1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\
\end{array}
\]

\[
\begin{array}{cccccccccc}
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\
\end{array}
\]

\[
\begin{array}{cccccccccc}
1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
\end{array}
\]

\[
\begin{array}{cccccccccc}
1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
\end{array}
\]

\[
\begin{array}{cccccccccc}
1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
\end{array}
\]
Average DFR estimation technique

Outline of the method

1. Derive the distribution of the syndrome weight $wt(s)$, $Pr(W_t = y)$
2. Derive the probability distribution of number of discrepancies between the error $e$ and its estimate $\bar{e}_{(1)}$ added ($d_+$) and removed ($d_-$) by the first iteration
3. Partition error estimate $\bar{e}_{(1)}$ bits after first iteration in classes, derive $Pr(E_{(2)} = d)$, and the DFR as $1 - Pr(E_{(2)} = 0)$

Bonus from code-specific knowledge (if available)

- [Til18, BBC+23]: Given a specific $H$ compute $\tau(H)$ s.t. for all $0 \leq x \leq \tau(H)$ $Pr(E_{(2)} = 0|E_{(1)} = x) = 1$, i.e., if $wt(e \oplus \bar{e}_{(1)}) \leq \tau(H)$ the 2nd iteration converges to $s = 0$
Syndrome weight distribution estimation

Method - Step 1

- Compute distribution of the r.v. \( W_t \) modeling \( \text{wt}(s) = \text{wt}(\text{He}^T) \), i.e., the syndrome weight of a weight-\( t \) error \( e \) through a \((v, w)\)-regular p.c. matrix \( H \)

Working assumption

- Rows of \( H \) are independently and uniformly random drawn from the set of binary vectors with length \( n \) and \( w \) asserted bits

Strategy

- \( W_t \) derived as the result of \( t \) steps on a non-homogeneous Markov Chain (MC):
  - MC steps model the effect of adding an asserted bit to \( e \Rightarrow \) column of \( H \) to \( s \)
  - MC transition probabilities derived counting the number of flips taking place in \( s \)
  - Initial distribution, i.e., \( W_0 \) is simply \( \text{Pr}(W_0 = 0) = 1 \)
Numerical validation of the distribution of $\mathcal{W}_t$

$(v, n_0 v)$ regular codes with $n = n_0 r$, $\text{wt}(e) = t$, $10^9$ samples per pt. (sim +, model ×)

$n_0 = 2, r = 2200, v = 11, t = 18$

$n_0 = 4, r = 13397, v = 83, t = 66$
First iteration discrepancy distribution estimation

Method - Step 2

- Model #discrepancies between $e$ and $\tilde{e}_{(1)}$, split into added ($d_+$) and removed ($d_-$), as random variables: $\delta_+(d_+) = \Pr(d_+ \text{ discrepancies added})$ and $\delta_-(d_-)$

Strategy

- Knowing $\mathcal{W}_t$, compute $\Pr(\text{ discrepancies added}|\mathcal{W}_t = w)$ for all $w \in \{0, \ldots, n - k\}$ through counting arguments
- Compute probability $p_{\text{unsat}|b}$ that a p.c. equation is unsatisfied, given that a bit involved in it $e_j$ is equal to $b \in \{0, 1\}$
- Compute probability distribution of $u_{\text{pc}_j}$ given that $e_j$ is equal to $b \in \{0, 1\}$
- For any 1st iteration threshold $\mathcal{E}_{(1)}$ of choice, compute $\delta_+(d_+)$ and $\delta_-(d_-)$
  - Note: The number of discrepancies after the 1st it. is: $\mathcal{E}_{(1)} = t - d_- + d_+$
Numerical validation of $\delta_+(d_+)$ and $\delta_-(d_-)$

$n_0 = 4$, $p = 13397$, $n = n_0p$, $k = (n_0 - 1)p$, $v = 83$, $t = 66$, $10^5$ samples per point

$(\text{sim } +, \text{ model } \times, \text{ technique from } [\text{BBC}^+23] \times)$
Second iteration failure rate estimation

Method - Step 3

- Obtain the second iteration DFR as $1 - \Pr(\mathcal{E}_{(2)} = 0)$

Strategy

- Partition positions of $\bar{e}_{(1)}$ into $J_{a,b}$, $a, b \in \{0, 1\}$, $a = e_j, b = e_j \oplus \bar{e}_{(1),j}$; for each $J_{a,b}$:
  - Derive the probability that a p.c. equation involving $\bar{e}_{(1),j}, j \in J_{a,b}$ becomes/stays unsat after the first iteration
  - Derive the UPC value distribution in the second iteration for $\bar{e}_{(1),j}, j \in J_{a,b}$
  - Combine the above with the distributions of $|J_{a,b}|$ (obtained from the ones of $d_+$ and $d_-$) to obtain $\Pr(\mathcal{E}_{(2)} = d)$
(v, 2v)-regular LDPC codes, \( v \in \{9, 11, 13, 15, 17\} \), \( \frac{k}{n} = \frac{1}{2} \), \( t = 18 \), parallel decoder w/ thresholds, \( \text{th1} = \text{th2} = \lceil \frac{v+1}{2} \rceil \). \( 10^8 \) decodes or 100 decoding failures per point.
(v, 2v)-regular LDPC codes, $t \in \{10, \ldots, 39\}$, $\frac{k}{n} = \frac{1}{2}$, $v = 11$, parallel decoder w/ thresholds, $\text{th}_1 = \text{th}_2 = \left\lceil \frac{v+1}{2} \right\rceil$. $10^8$ decodes or 100 decoding failures per point.
Impact on code based-cryptosystem design

Comparison with previous non-extrapolation estimates on 2 iterations decoder

<table>
<thead>
<tr>
<th>$n_0$</th>
<th>$p$</th>
<th>$v$</th>
<th>$t$</th>
<th>$\min \tau(H)$</th>
<th>LEDAcrypt</th>
<th>This work</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>23371</td>
<td>71</td>
<td>130</td>
<td>10</td>
<td>$2^{-64}$</td>
<td>$2^{-147}$</td>
</tr>
<tr>
<td>3</td>
<td>16067</td>
<td>79</td>
<td>83</td>
<td>9</td>
<td>$2^{-64}$</td>
<td>$2^{-139}$</td>
</tr>
<tr>
<td>4</td>
<td>13397</td>
<td>83</td>
<td>66</td>
<td>8</td>
<td>$2^{-64}$</td>
<td>$2^{-134}$</td>
</tr>
<tr>
<td>2</td>
<td>28277</td>
<td>69</td>
<td>129</td>
<td>11</td>
<td>$2^{-128}$</td>
<td>$2^{-203}$</td>
</tr>
<tr>
<td>3</td>
<td>19709</td>
<td>79</td>
<td>82</td>
<td>10</td>
<td>$2^{-128}$</td>
<td>$2^{-198}$</td>
</tr>
<tr>
<td>4</td>
<td>16229</td>
<td>83</td>
<td>65</td>
<td>9</td>
<td>$2^{-128}$</td>
<td>$2^{-189}$</td>
</tr>
</tbody>
</table>

- Computations above consider that for all $0 \leq x \leq \tau(H)$ $\Pr(\mathcal{E}(2) = 0 | \mathcal{E}(1) = x) = 1$
- Computations above done with syndrome independent thresholds
  - Syndrome weight dependent thresholds can also be modeled
  - Employing them yields a more effective decoder, lowering DFR further
Considerations on weak keys

**Effects of weak keys**

- Weak keys [DGK20, Vas21, ABH+22, WWW23] are p.c. matrices defining codes with poor correction capabilities; they are detrimental to the average DFR
- This work provides a technique to estimate the average DFR over all the possible codes (keypairs), employing a 2-iteration BF decoder
  - This matches the IND-CCA2 requirement [HHK17]

**Filtering**

- Weak keys from [DGK20, Vas21] can be filtered via pattern-matching
- [BBC+20, BBC+23]: Weak keys are characterized by $\tau(H)$ values definitely below average and can be removed discarding codes with $\tau(H)$ below a chosen threshold $\bar{\tau}$
  - Bonus point: the improvement of the average DFR is automatically quantified in our approach
Concluding remarks

Take-away points

- We provide a closed-form method to estimate the average DFR of a random \((v, w)\)-regular code decoded via 2-iterations parallel BF iterative decoding
- Adopting our approach and tuning BIKE parameters accordingly would yield an IND-CCA2 version of BIKE
- The effect of weak keys is taken into account in our estimates, considering both the case in which they are discarded and the one in which they’re not

Ongoing future directions

- Extend the technique to a higher number of parallel BF decoder iterations
- Complete a performance-security optimized design for LEDAcrypt parameters, with syndrome-weight dependent thresholds
Thank you for the attention!


Jean-Pierre Tillich.  
The Decoding Failure Probability of MDPC Codes.  

Valentin Vasseur.  

Tianrui Wang, Anyu Wang, and Xiaoyun Wang.  
Exploring decryption failures of BIKE: new class of weak keys and key recovery attacks.  