A Weighted Bit Flipping Decoder for QC-MDPC-based Cryptosystems

Alexander Nilsson*†, Irina E. Bocharova‡, Boris D. Kudryashov‡ and Thomas Johansson*,

*Dept. of Electrical and Information Technology,
Lund University
Email:{alexander.nilsson, thomas.johansson}
@eit.lth.se

†Advenica AB,
Malmö, Sweden

‡Dept. of Information Systems
St. Petersburg Univ. of Information Technologies,
Mechanics and Optics
and Univ. of Tartu, Estonia
Email: {irinaboc, boriskud}@ut.ee

***Abstract*—A new "Weighted Bit-flipping" (WBF) iterative decoder is presented and analyzed with respect to its Decoding Failure Rate (DFR). We show that the DFR is indeed lower than that of the BGF decoder as suggested by the BIKE third round submission to the NIST PQC standardization process. The WBF decoder requires more iterations to complete than BGF, but by creating a hybrid decoder we show that a lower DFR compared to that of the BGF decoder can still be achieved while keeping the computational tradeoff to a minimum.**

I. Introduction

It is well-known that quantum computers (QC) can break the ubiquitous public key cryptosystems (PKC) RSA and ECC as well as the key-agreement protocols DH and ECDH. This is due to QCs ability to solve the integer factorization and the discrete logarithm problems in polynomial time. To prepare against the development of QCs of sufficient scale to threaten online communications, many potential replacement algorithms have been suggested [1]–[4] in the field of post quantum cryptography (PQC). The code-based PKC scheme McEliece [1] is one of these proposed replacements. The security of this PKC is based on hardness of decoding a random linear code (from the field of coding theory). However, relatively recently, new versions this code-based PKC were proposed.

One of the more promising PQC candidates is based on the use of quasi-cyclic moderate density parity-check (QC-MDPC) codes [5], instead of Goppa codes. These code-based schemes improves McEliece in several aspects, one being the simplified description of a code-based PKC without a permutation matrix; another one being the reduced size of the public key. Another key factor in favor of MDPC codes is that they allow for relatively efficient decoding for a legitimate user while still ensuring a required level of security for the corresponding public key. Based on this new idea of using iterative decoding of MDPC codes, a number of different schemes have since then been published. In particular, the NIST PQC standardization process' third round submission schemes BIKE [2] and HQC [4] are relevant this paper.

On the flip side of the coin, much effort is required in both lowering the DFR[1] of the iterative decoding process and in increasing the number of correctable errors. Failing to do so affects the legitimate user because it raises the requirement of the block-size (and therefore communication bandwidth) but more importantly, a poor DFR performance enables reaction attacks [6]–[8] based on the observation of uncorrectable error patterns. One decoder is the original iterative Bit-Flipping (BF) decoder [9] and it, when applied to QC-MDPC codes, suffers from relatively poor DFR performance due to it originally being developed for the low-density parity-check (LDPC) codes that are intensively employed in communication systems. The inefficiency of the BF decoder stems from the higher probability of MDPC codes to include short cycles in the corresponding Tanner graph [5].

A number of different attempts to improve BF decoding of QC-MDPC codes have been developed in the last couple of years. In particular, the Black-Grey Flip (BGF) decoder [10] was selected by BIKE due to its quick convergence (i.e., good DFR despite a small number of iterations).

Similarities between LDPC codes and MDPC codes allow for a transfer of ideas which lie behind some improved BF algorithms to the cryptography area. However, specific features of MDPC codes require significant modifications of known iterative decoding techniques. Examples of improved BF algorithms of this type can be found in [11] and [12], where BF decoding followed by post-processing was studied.

Our main contribution in this paper is a new decoder design following an analysis of belief propagation (BP) decoding. Due to the increased computational complexity of our decoder, we additionally present a hybrid of our design and that of the BGF decoder. Both new decoders are evaluated with simulations to confirm their respective DFRs.

II. QC-MDPC based McEliece cryptosystem

Key Generation, Encryption and Decryption can be performed in a number of different ways (e.g., see BIKE specification [13]), but for the sake of simplicity and generality

[1]Decoding Failure Rate, also known as the Frame Error Rate (FER) in the field of coding theory.

we present a summary of the original proposal by Misoczki, Tillich and Sendrier [5]. We assume that differences between different variants of QC-MDPC-based schemes do not affect the relative DFR and/or computational efficiency of decoding algorithms, although the performance in absolute terms is expected to depend on the instantiated scheme in question.

We consider the (n,r,w)-QC-MDPC code, where $n=n_0r$. It is defined through a parity check matrix

$$H=\begin{pmatrix}H_0 & H_1 & \cdots & H_{n_0-1}\end{pmatrix}, \quad (1)$$

where the $r\times r$ submatrices H_i, $i=0,1,\dots,n_0-1$ are circulant matrices with row weight w_i. Notice that for MDPC codes $\sum_{i=0}^{n_0-1} w_i$ is chosen to be of order $\sqrt{n\log n}$. In the sequel, for simplicity, we analyze only the case of $n_0=2$ and equal weights $w_i=w$, for $i=\{0,1\}$.

(a) *Key Generation:* Generate two random sequences $\boldsymbol{h}_0,\boldsymbol{h}_1$ of length $r=n/2$ and Hamming weight w. Form circulant matrices H_0,H_1 as cyclic shifts of $\boldsymbol{h}_0,\boldsymbol{h}_1$, respectively. Construct a generator matrix G as

$$G=\left(\begin{array}{c|c} I & Q\end{array}\right), \text{where } Q=\left(H_1^{-1}H_0\right)^{\mathrm{T}}, \quad (2)$$

where T denotes matrix transposition and I is the identity matrix. The public key is G (compactly represented by the first row of Q) and the private key is H (compactly represented by the sequence $\boldsymbol{h}_0,\boldsymbol{h}_1$).

(b) *Encryption:* Generate a random sequence $\boldsymbol{e}$ of length n and Hamming weight t. The ciphertext $\boldsymbol{x}$ for the plaintext $\boldsymbol{m}$ is $\boldsymbol{x}=\boldsymbol{u}+\boldsymbol{e}$, where $\boldsymbol{u}=(\boldsymbol{m},\boldsymbol{m}Q)$.

(c) *Decryption:* Perform iterative decoding of $\boldsymbol{x}$ by using the known matrix H from Equation (1). Obtain the plaintext $\boldsymbol{m}$ as the first k bits of the decoded sequence.

III. Decoding of MDPC Codes

In this section, we briefly recall the iterative decoding algorithms applied to decoding LDPC/MDPC codes. There are many implementations of the BF decoding [14], which differ both in complexity and error correcting performance. For the detailed overview of MDPC-code-based versions see [10], [15]. All of these versions are used in the same manner as for LDPC codes, determined by sparse parity-check matrices. Our motivation is to modify the BF algorithm in order to take into account that parity-check matrices of MDPC codes are much denser than those of LDPC codes.

A. Belief propagation decoding

We start with describing the soft-decision BP algorithm, which shows close to optimal performance when used with LDPC codes. In decryption, we interpret vectors $\boldsymbol{u}$ and $\boldsymbol{x}=(x_1,\dots,x_n)=\boldsymbol{u}+\boldsymbol{e}$ as the input and output of the binary symmetric channel (BSC) with crossover probability $\epsilon=t/n$. In order to apply BP decoding, we transform the hard-decision BSC output to a soft-decision sequence of log-likelihoods ratios (LLRs) $\boldsymbol{y}=(y_1,\dots,y_n)$, where

$$y_i=\mu(1-2x_i),\ \mu=\log\frac{1-\epsilon}{\epsilon}. \quad (3)$$

BP decoding (alternatively called message passing algorithm) has many different implementations. Two of them, namely, sum-product (SP) algorithm [14] and its approximation, called min-sum (MS) algorithm [16] are well known in coding theory.

Let $V=\{1,\dots,n\}$ and $C=\{1,\dots,r\}$ be the sets of variable and check nodes (code symbols and parity-checks) of an MDPC code's Tanner graph and denote by $Z=\{z_{cv}\}$ messages exchanged between nodes $c\in C$, $v\in V$. Notice that number of elements in Z is determined by the number of non-zero elements in the parity-check matrix H.

1) SP and MS decoding:

(a) Initialization: let $z_{cv}=y_v$, $v\in V$, $c\in C$.

(b) At each iteration of BP decoding

- For each $c\in C$ and all v connected to c compute either

$$z_{cv}=\prod_{v'\neq v}\operatorname{sign}(z_{cv'})\phi\left(\sum_{v'\neq v}\phi(|z_{cv'}|)\right), \quad (4)$$

where $\phi(\lambda)=\ln\frac{e^\lambda+1}{e^\lambda-1}$, $\lambda>0$. or

$$z_{cv}=\prod_{v'\neq v}\operatorname{sign}(z_{cv'})\alpha\left(\min_{v'\neq v}|z_{v'}|-\beta\right), \quad (5)$$

for SP and MS versions, respectively. The constants $\alpha>0$ and $\beta\geq 0$ represent a normalization factor and an offset, respectively.

- For each $v\in V$ and all c connected to v update

$$z_{cv}=y_v+\sum_{c'\neq c} z_{c'v}. \quad (6)$$

(c) Compute hard decisions as

$$u_v=\operatorname{sign}\left(y_v+\sum_c z_{cv}\right). \quad (7)$$

Early termination of the decoding procedure can be done if at each iteration the sequence of hard decisions is computed according to Equation (7) and the corresponding syndrome is evaluated. All-zero syndrome vector indicates that a valid codeword is recovered. The complexity of the SP version of BP decoding is determined by the computational complexity of Equation (4).

2) Bit-Flipping (BF) decoding: BF decoding is a simple hard-decision version of BP decoding algorithm. The two originally suggested in [14]. BF algorithms differ in the way the decoding threshold is chosen.

(a) Initialization: compute syndrome $\boldsymbol{s}=\boldsymbol{x}H^{\mathrm{T}}$.

(b) At each iteration

- For each $c\in C$ and all v connected to c set $z_{cv}=s_c$, where s_c is the c-th component of $\boldsymbol{s}$.
- For each $v\in V$ and all c connected to v compute a number of unsatisfied checks (NUC): $\sigma_v=\sum_c z_{cv}$.
- Flip symbols of $\boldsymbol{x}$ at positions v such that $\sigma_v>\max_v\sigma_v-\delta$.

(c) Return $\boldsymbol{x}$.

Early termination can be performed if none of positions were flipped at some iteration. The parameter $\delta > 0$ determines a tradeoff between DFR and complexity.

B. Black-Grey-Flip Decoder (BGF)

Building on the basic BF (Bit-Flipping) decoder we first briefly describe the "Black-Grey" (BG) decoder[2] described in [17]. On top of the BF algorithm this decoder utilizes two additional lists (called black and grey, respectively) to keep track of bit flips which are considered "uncertain". The bits specified in these lists are reevaluated against conditions. Bits that meet these conditions are flipped again, thereby undoing the first flip. Each iteration of the BG implementation is divided into three steps.

Step I Perform a single round of BF decoding, (i.e. flipping bits according to their NUC value σ_v and a threshold τ) producing two lists:

- A black list containing all flips performed.
- A grey list containing all flips with a NUC value close to (but not above) the threshold.

Recompute the syndrome.

Step II Another round of BF decoding but this time only consider bits in the black list. Recompute the syndrome.

Step III Another round of BF decoding but this time only consider bits in the grey list. Recompute the syndrome.

BGF introduces an optimization due to recognizing (see [17]) that it is only during the first iteration there are uncertain bits to reevaluate. Steps II and III are therefore only performed for the first iteration. Leaving only Step I makes BGF equivalent to BF decoding during the remaining number of iterations. One can view the relative time-complexity of the BGF decoder as $1 \times \text{BG}_{\text{rounds}} + 4 \times \text{BF}_{\text{rounds}} \approx 7 \times \text{BF}_{\text{rounds}}$ according to [17].

IV. Analysis of BF decoding of MDPC codes

In BP decoding (SP, MS or BF) we associate with nonzero elements of parity-check matrix H an array of random variables z_{cv}. At decoding iterations, z_{cv} values are updated depending on syndrome values. In case of BF decoding the Hamming weight of parity checks determines row-processing errors to next iterations. In what follows, we try to construct an approximate model of error propagation in BP decoding.

For a code of rate R defined by a parity-check matrix H with column weight w, a single error influences w rows with $d = w/(1-R) = wn_0$ nonzero symbols in each row. In total, $N_1 = wd$ variables z_{cv} associated with nonzero elements of H in w parity checks are affected by a single error. Among them only w elements belong to the column corresponding to the "true" erroneous variable node. The other $(wd - w)$ nonzeros can be interpreted as an additional binary noise caused by correlation between parity checks.

If the number of introduced errors $t \gg 1$, then the nonzero symbols of the binary noise are assumed to be independent and identically distributed. Assume that a single variable z_{cv} is equal to 1 due to a single channel error with probability

$$p = \frac{\text{\#of affected positions}}{\text{total \# of nonzero elements}} = \frac{wd - w}{rd} = \frac{w - 1 + R}{r}.$$

If t errors occur then z_{cv} will be equal to 1 if the number of its flips will be odd and equal to 0, otherwise. The probability of "false" ones caused by odd number of flips is equal to (see [14])

$$\varepsilon = \frac{1 - (1 - 2p)^t}{2} = \frac{1 - \left(\frac{2(w-1+R)}{r}\right)^t}{2}. \tag{8}$$

Below we interpret ε in Equation (8) as the probability of errors in computing NUCs induced by the binary noise caused by high correlation of parity checks.

Example 1: Let $r = 4801$, $w = 45$, $t = 100$. It is easy to check that $\varepsilon = 0.423$ and that the expected value of NUC is equal to $\epsilon w = 18.9$ for error-free positions and is equal to $(1 - \epsilon)w = 26.1$ for positions in error. These estimates were confirmed by long simulations. □

What follows from this example is that the information about errors obtained from NUC values σ_v is "extremely noisy". We can expect $\sigma_v < w/2$ for many error positions and $\sigma_v > w/2$ for many error-free positions In other words, syndrome values used in BF decoding for making decisions about bits to be flipped are typically unreliable. These considerations lead us to the conclusion that syndrome values used in BF decoding should be classified and weighted according to their 'reliabilities'. Similar ideas are behind one improvement of BF decoding in [18].

V. The new versions of BF decoder

For a given syndrome vector $\boldsymbol{s} = (s_0, \ldots, s_{r-1})$, particular values $s_c = 0$ or 1 mean that the number of erroneous (to be flipped) positions θ_c among positions involved into the c-th check is even or odd, respectively. Our goal is to find a statistical criterion for distinguishing between cases $\theta_c = 0$ or $\theta_c \in \{2, 4, \cdots\}$ if $s_c = 0$ and between cases $\theta_c = 1$ or $\theta_c \in \{3, 5, \cdots\}$ if $s_c = 1$. In other words, we aim at estimating "reliability" of the syndrome components s_c.

Positions v with the number of NUCs $\sigma_v > w/2$ are considered as candidates for flipping. For a check c, $c = 0, \ldots, r-1$ let $\boldsymbol{\sigma}^{[c]} = (\sigma_0^{[c]}, \ldots, \sigma_{d-1}^{[c]})$ be a sequence of $\sigma_i = \sigma_{v_i}$, where v_i, $i = 0, \ldots, d-1$ are nonzero symbols of the check c. We are going to measure reliability of s_c as a function of the NUC sequence $\boldsymbol{\sigma}^{[c]}$.

Let $\chi(x)$ be an indicator function which is equal to 1 if x is true and 0, otherwise. The number of votes for flipping in the c-th parity check can be expressed as

$$\theta_c = \sum_{i=0}^{d-1} \chi(\sigma_i^{[c]} > w/2). \tag{9}$$

To obtain a quantitative measure of reliability we evaluated, empirically for the MDPC code from Example 1, the following

[2]"Black-Grey-Flip" (BGF) is an optimized variant of the BG decoder first found in the BIKE pre-Round-1 submission CAKE by Sendrier and Misoczki.

logarithmic likelihood values

$$A_0(\theta_c) = \log \frac{\Pr\left(e_c = 0 | s_c = 0, \theta_c\right)}{\Pr\left(e_c > 0 | s_c = 0, \theta_c\right)}, \tag{10a}$$

$$A_1(\theta_c) = \log \frac{\Pr\left(e_c = 1 | s_c = 1, \theta_c\right)}{\Pr\left(e_c \neq 1 | s_c = 1, \theta_c\right)}, \tag{10b}$$

where e_c is the number of errors (out of t introduced errors) in the symbols of the c-th check. Functions $A_s(\theta_c)$, $s = 0, 1$ are identical for all checks $c = 0, \ldots, r-1$ due to the code regularity. This allows to use $A_s(\theta)$ as a reliability measure for the syndrome value s given that the number of votes for flipping in a check is equal to θ. Functions $A_s(\theta)$ empirically evaluated for the code of Example 1 are presented in Figure 1. We can see that to the syndrome components s corresponding to the check with a small number of votes θ should be assigned 3–4 times larger weights than to the components corresponding to checks with large θ.

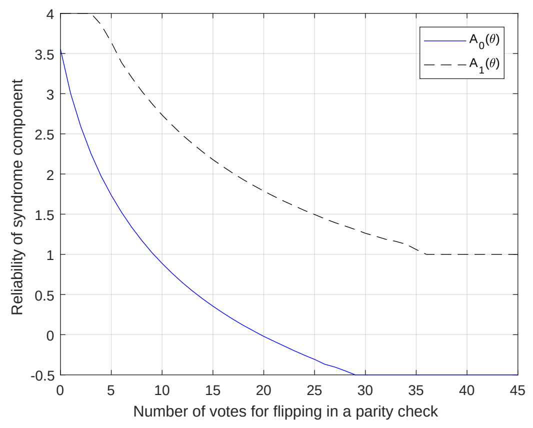

Fig. 1. Logarithmic likelihood ratios defined as in Equation (10) for syndrome components as functions of the number of votes for flipping (NUCs above $w/2$)

It follows that after computing syndrome and NUCs, one can compute weighting factors for syndrome component either by A_0 or by A_1 depending on the value of the syndrome component. For all variable nodes, the weighted NUC values can be obtained by adding the reliability info with NUC weighted by 1/2 (chosen empirically). The rest of the algorithm works as in the original BF algorithm with the threshold equal to the maximal shifted and weighted NUC value. A variable node with the maximal value of combined reliability and NUC is flipped.

The algorithm shown in Figure 2 is an approximated version of the theoretical WBF algorithm described above. Instead of using fixed arrays of weighting coefficients A_0, A_1, the algorithm searches for minimum numbers of votes separately for satisfied and unsatisfied checks (lines 6, 7). The contribution of syndrome components with the smallest number of votes is amplified (lines 10 and 12) by multiplication with the fixed coefficient 3.

Simulations show that performance of the theoretical and simplified weighted algorithms are hardly distinguishable.

```
 1  function [y,it]= WBF(y,V,C,w,maxit)
 2  s=mod(sum(y(V),2),2)'; % syndrome
 3  for it=1:maxit
 4    nuc=sum(s(C)); % NUC
 5    SM=sum(nuc(V)>w/2,2)'; % votes
 6    minSM1=min(SM(s==1));
 7    minSM0=min(SM(s==0));
 8    REL=2*s-1;
 9    mask=(s==0) & (SM<=minSM0+3);
10    REL(mask==1)=REL(mask==1)*3;
11    mask=(s==1) & (SM<=minSM1+3);
12    REL(mask==1)=REL(mask==1)*3;
13    metric=sum(REL(C));
14    [mm,~]=max(metric);
15    f=find(metric==mm);
16    [~,J]=min(sum(SM(C(:,f))));
17    s(C(:,f(J)))=~s(C(:,f(J)));
18    y(f(J))=~y(f(J)); % flip
19    if sum(s)==0, return; end
20  end
```

Fig. 2. Matlab program for simplified WBF decoding.

In Figure 2, only one position is flipped per iteration. Similarly to the BF decoding, a certain threshold can be selected in such a way that at step 16 in Figure 2 more than one flip can be allowed. This modification will make the convergency faster at the cost of slightly increasing DFR. The simulated WBF decoder in Section VI-B use this optimization.

A. Complexity and constant-time considerations

A single round of normal BF decoding comprises of the following parts:

- (Re)compute the syndrome
- Check the syndrome Hamming weight.
- Calculate NUCs.
- Check NUCs and flip based on the threshold.

It has been shown to be possible to implement these operations efficiently and in constant time [19]. For WBF decoding we must *additionally* consider the following operations:

- Calculate the number of votes to flip, for each syndrome position.
- Find the minimum number of votes. [3]
- Determine the reliability of each syndrome value (based on number of votes).
- Determine a new "metric" for each variable node (based on the weighted syndrome values).
- Find and flip the variables with the highest metric.

It is possible to implement these operations in a constant time manner, since none of them requires branching or indexing based on secret data. However, to implement them *efficiently* is left as out of scope for this paper.

[3]It might be possible to empirically determine a static threshold to use instead of dynamically calculate the minimum.

B. A hybrid decoder

Accounting for the decrease in interference noise due to a reduction in the number of error positions, we know that making the right bit-flipping decisions is more important early in the decoding process. Due to this, one can argue for an optimization of using an adaptable decoding algorithm, which progressively uses less and less expensive computations (this is exactly the optimization introduced for the BG decoder to make the BGF decoder) [17].

To realize such an algorithm while keeping the implementation complexity low and constant time we look into a hybrid solution where we pair two different decoding algorithms based on DFR and execution speed criteria.

We consider the pairing of our WBF algorithm together with Black Grey Flip (BGF) decoder, which has a very quick convergence. We limit WBF to run only 2 iterations, after which we run the (BGF) decoder for 3 iterations. As previously explained, this means that the BGF decoder will run a single round of BG decoding followed by 2 rounds of normal BF decoding. The complexity of the combined algorithm can informally be thought of as $2 \times \text{WBF}_{\text{rounds}} + 1 \times \text{BG}_{\text{rounds}} + 2 \times \text{BF}_{\text{rounds}} \approx 2 \times \text{WBF}_{\text{rounds}} + 5 \times \text{BF}_{\text{rounds}}$. Appreciating the fact that one round of WBF requires more computations than one round of BF we recognize that even this hybrid will be slower than that of the BGF decoder. Our hybrid decoder can still remain a competitive alternative if properly implemented (e.g., by using techniques from [19]).

VI. Simulation

In this section, we present a comparative analysis between our proposed decoders and the BGF decoder.

A. Method

We apply a similar methodology as the one given in [17], [20]–[22]. By using the same parameters as those used by BIKE (security level 1) [13], we facilitate easier comparisons of different DFR rates. One must note that we use a different way of encoding messages, which might affect the DFR rates in absolute terms. However, the relative differences between decoders are assumed to be unaffected, as previously stated.

B. Results

In this section, we present the simulation results of the decoding algorithms studied in this paper. The data is presented in Figure 3. Using the same methodology as in [17], [20]–[22], we attempt in Table I to estimate a value of r that achieves a DFR $= 2^{-128}$. The estimation was made using a simple linear extrapolation from the last two data points of each graph. This is the most straightforward and conservative choice [20].

In Table I the extrapolation does not yield quite the expected values. For instance, the Hybrid decoder performs better than WBF, although the opposite is clearly visible in Figure 3. We assume this to be a limitation of the experiment; even lower DFRs are needed if we want to make certain that we are indeed extrapolating only the linear part of the curves. The WBF and hybrid algorithms are also fine-tuned for lower r-values with a high DFR, due to practical reasons. It's possible that the WBF algorithm can be further tuned for r-values corresponding to a low DFR.

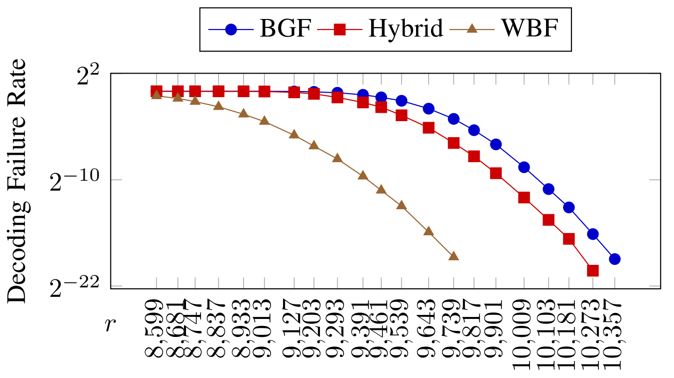

Fig. 3. Graph of the DFR as a function of r, for different decoding algorithms. Minimum 1,000 decryption failures for each data point with each key used for maximum 10,000 ciphertexts.

TABLE I
A summary of the different decoding algorithms. See text for extrapolation method.

Algorithm	Max # Iterations	Average # Iterations	Extrapolated r (DFR $= 2^{-128}$)
WBF	92	34	13379
Hybrid	5	5	13018
BGF	5	5	13660

VII. Conclusion

In this paper, we presented a new Weighted Bitflipping iterative decoder (WBF). We showed that it significantly lowers the DFR for all simulated parameters compared to other decoders. Specifically, we compared it to the Black Grey Flip decoder as found in the NIST PQC 3rd round submission BIKE [13].

Due to the increased computational complexity of the WBF decoder, we proposed a hybrid decoder which combines the quick convergence of the BGF decoder with the better DFR of the WBF decoder. This results in a much smaller computational complexity by lowering the required number of iterations.

Acknowledgments

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation, by the grant PRG49 from the Estonian Research Council and by the ERDF via CoE project EXCITE. The computations were enabled by resources provided by the Swedish National Infrastructure for Computing (SNIC) at LUNARC, partially funded by the Swedish Research Council through grant agreement no. 2018–05973.

REFERENCES

[1] R. J. Mceliece, "A public-key cryptosystem based on algebraic coding theory," *Coding Thv*, vol. 4244, pp. 114–116, 1978.

[2] N. Aragon, P. Barreto, S. Bettaieb, L. Bidoux, O. Blazy, J.-C. Deneuville, P. Gaborit, S. Gueron, T. Guneysu, C. A. Melchor *et al.*, "Bike: bit flipping key encapsulation," 2017.

[3] J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehlé, "Crystals – kyber: a cca-secure module-lattice-based kem," Cryptology ePrint Archive, Report 2017/634, 2017, https://eprint.iacr.org/2017/634.

[4] C. A. Melchor, N. Aragon, S. Bettaieb, L. Bidoux, O. Blazy, J.-C. Deneuville, P. Gaborit, E. Persichetti, G. Zémor, and I. Bourges, "Hamming quasi-cyclic (hqc)," *NIST PQC Round*, vol. 2, pp. 4–13, 2018.

[5] R. Misoczki, J.-P. Tillich, N. Sendrier, and P. S. Barreto, "MDPC-McEliece: New McEliece variants from moderate density parity-check codes," in *IEEE Trans. Inf. Theory*, 2013, pp. 2069–2073.

[6] Q. Guo, T. Johansson, and P. Stankovski, "A key recovery attack on MDPC with CCA security using decoding errors," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2016, pp. 789–815.

[7] Q. Guo, T. Johansson, and P. S. Wagner, "A Key Recovery Reaction Attack on QC-MDPC," *IEEE Trans. on Inf. Theory*, 2018.

[8] A. Nilsson, T. Johansson, and P. Wagner Stankovski, "Error Amplification in Code-based Cryptography," *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, pp. 238–258, 2019.

[9] R. G. Gallager, *Information theory and reliable communication*. Wiley, 1968.

[10] N. Drucker, S. Gueron, and D. Kostic, "On constant-time QC-MDPC decoding with negligible failure rate," Cryptology ePrint Archive, Report 2019/1289, Tech. Rep., 2019.

[11] I. E. Bocharova, B. D. Kudryashov, and V. Skachek, "AVN-based elimination of short cycles in Tanner graphs of QC LDPC Codes," in *2019 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2019, pp. 56–60.

[12] H. Bartz and G. Liva, "On decoding schemes for the MDPC-McEliece cryptosystem," *arXiv preprint arXiv:1801.05659*, 2018.

[13] N. Aragon, P. S. L M Barreto, S. Bettaieb, O. Blazy, P. Gaborit, S. Gueron, C. Aguilar Melchor, R. Misoczki, E. Persichetti, and G. Zémor, "BIKE: Bit flipping key encapsulation submission for round 3 consideration," Tech. Rep., 2020.

[14] R. G. Gallager, *Low-density parity-check codes*. M.I.T. Press: Cambridge, MA, 1963.

[15] I. V. Maurich, T. Oder, and T. Güneysu, "Implementing QC-MDPC McEliece encryption," *ACM Trans. on Embedded Computing Systems (TECS)*, vol. 14, no. 3, p. 44, 2015.

[16] M. P. Fossorier, M. Mihaljevic, and H. Imai, "Reduced complexity iterative decoding of low-density parity check codes based on belief propagation," *IEEE Trans. on Commun.*, vol. 47, no. 5, pp. 673–680, 1999.

[17] N. Drucker, S. Gueron, and D. Kostic, "QC-MDPC decoders with several shades of gray," Tech. Rep. Report 2019/1423, 2020. [Online]. Available: https://eprint.iacr.org/2019/1423.pdfhttps://eprint.iacr.org/2019/1423

[18] H. Kamabe and S. Kobota, "Simple improvements of bit-flipping decoding," in *2010 The 12th International Conference on Advanced Communication Technology (ICACT)*, vol. 1. IEEE, 2010, pp. 113–118.

[19] N. Drucker and S. Gueron, "A toolbox for software optimization of QC-MDPC code-based cryptosystems," *J. Cryptogr. Eng.*, vol. 9, no. 4, pp. 341–357, nov 2019. [Online]. Available: http://link.springer.com/10.1007/s13389-018-00200-4

[20] N. Sendrier and V. Vasseur, "About low DFR for QC-MDPC decoding," in *International Conference on Post-Quantum Cryptography*. Springer, 2020, pp. 20–34.

[21] ——, "On the decoding failure rate of QC-MDPC bit-flipping decoders," vol. 11505 LNCS, pp. 404–416, 2019. [Online]. Available: http://link.springer.com/10.1007/978-3-030-25510-7{_}22

[22] N. Drucker, S. Gueron, and D. Kostic, "On constant-time QC-MDPC decoders with negligible failure rate," in *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, 2020, vol. 12087 LNCS, pp. 50–79. [Online]. Available: http://link.springer.com/10.1007/978-3-030-54074-6{_}4