Worst case QC-MDPC decoder for McEliece cryptosystem

Julia Chaulet, Nicolas Sendrier

Worst case QC-MDPC decoder for McEliece cryptosystem

Julia Chaulet
Inria, Paris, France
Thales TCS, Gennevilliers, France
julia.chaulet@inria.fr

Nicolas Sendrier
Inria, Paris, France
nicolas.sendrier@inria.fr

***Abstract*—QC-MDPC-McEliece is a recent variant of the McEliece encryption scheme which enjoys relatively small key sizes as well as a security reduction to hard problems of coding theory. Furthermore, it remains secure against a quantum adversary and is very well suited to low cost implementations on embedded devices.**

Decoding MDPC codes is achieved with the (iterative) bit flipping algorithm, as for LDPC codes. Variable time decoders might leak some information on the code structure (that is on the sparse parity check equations) and must be avoided. A constant time decoder is easy to emulate, but its running time depends on the worst case rather than on the average case. So far implementations were focused on minimizing the average cost. We show that the tuning of the algorithm is not the same to reduce the maximal number of iterations as for reducing the average cost. This provides some indications on how to engineer the QC-MDPC-McEliece scheme to resist a timing side-channel attack.

I. Introduction

With the advance of quantum computing, efficient algorithms against number theory based cryptosystems [1] have become a threat that cannot be neglected. There is a need to develop post-quantum cryptosystems, that is cryptosystems which remain secure against an adversary equipped with a quantum computer.

Among all possible techniques for post-quantum cryptosystems we found code-based cryptography and in particular the McEliece public-key encryption scheme [2]. The original McEliece scheme uses (randomly permuted) binary Goppa codes and so far has resisted all cryptanalytic attempts even from quantum adversaries. The security is twofold and relies on two assumptions, the hardness of generic decoding, the *message security*, and the pseudorandomness of Goppa codes, the *key security*. Generic decoding is NP-complete [3] and is also believed to be hard on average. Though the pseudorandomness of Goppa codes has not been studied as thoroughly as generic decoding, no efficient algorithm is known to distinguish a random matrix from a generator matrix of a Goppa code, except when the code rate is close to one [4], a case which is not a threat against the McEliece encryption scheme.

This work was supported in part by the Commission of the European Communities through the Horizon 2020 program under project number 645622 PQCRYPTO.

One of the drawback of code-based public-key schemes is that they require large public keys (a generator matrix of the code). Gaborit proposed to use quasi-cyclic codes [5] to solve that problem. As demonstrated in [6], the use of quasi-cyclic codes does not significantly change the security reduction. However, the key security, which essentially requires that the public key (a generator matrix) does not leak information on the algebraic structure, is more problematic, and quasi-cyclic codes with algebraic structure may sometimes have vulnerabilities [7].

In 2013, the use of quasi-cyclic MDPC (Moderate Density Parity Check) codes was suggested to instantiate the McEliece scheme in [8]. This version of McEliece enjoys relatively small key sizes (a few thousand bits) and its security provably reduces to the hardness of deciding whether a given quasi-cyclic code contains a word of small weight or not. Moreover, the decryption essentially consists in decoding and can be achieved with the same iterative algorithms as for LDPC codes. In particular, a low cost implementation, suitable for embedded systems, can be achieved using a hard decision bit flipping iterative algorithm, as demonstrated in [9].

Our contributions. In this paper we will further examine the decoding of MDPC codes. The bit flipping decoding algorithm involves a threshold which can be chosen in many different ways and which may change along the iterations of a given decoding instance. A series of paper [9], [10], [11] are focused on how to choose the threshold in order to reduce the average number of iterations to successfully decode a given number of errors. However, if an adversary is able to measure, through a *side-channel*, the number of iterations of the decoder, he might be able to deduce some information on the secret key (the sparse parity check matrix). That is to say, for instance, by submitting properly chosen noisy codewords to the decoder and observing the decoder behavior for each of them.

Such an attack can be countered by designing a constant time decoder. In the present case this means adding fake iterations until we reach a prescribed number of iterations. This number will be chosen such that stopping the decoding process at this point ensures a negligible probability of failure. In other words, we wish to optimize the decoder behavior in the worst case rather than in the average case in order to minimize the decoding cost.

The main observation of this paper is that minimizing

the average cost leads to significantly different algorithms than minimizing the worst case. Using a heuristic approach we have determined a variant of the bit flipping algorithm which favors the worst case for a particular set of parameters. Combined with intensive simulation, it allows us to provide some guidelines for the engineering of QC-MDPC-McEliece decryption aiming to resist to above mentioned timing attacks. It also allows us to understand how to choose the parameters of both the system and the decoding algorithm such that the probability of decoding failure remains negligible.

II. Preliminaries

$\mathbb{F}_2$ denotes the field with two elements, $\mathrm{wt}()$ denotes the Hamming weight.

A. Moderate Density Parity Check Codes

The Moderate Density Parity Check (MDPC) code construction is very similar to the LDPC code's one. They both are binary linear codes defined by a sparse parity check matrix H and only differ on the sparseness of this matrix. For MDPC codes, the parity check matrix row weight w is much larger than for LDPC codes, typically $w = O(\sqrt{n})$. We denote (n, r, w)-MDPC, a MDPC code of length n, codimension r whose sparse parity check matrix has a row weight w.

Definition 1. *A square matrix is said circulant if its rows are the successive cyclic shifts of its first one.*

A linear code is *quasi-cyclic* (QC) if its generator or parity check matrix is composed of circulant blocks.

Though the parameters are flexible, all instances of QC-MDPC-McEliece proposed in [8] use codes of rate $1/2$. For the sake of simplicity, we will do the same in the rest of this paper and consider QC-MDPC codes with two (circulant) blocks, $H = [H_0 \mid H_1] \in \mathbb{F}_2^{r \times n}$.

B. McEliece Cryptosystem Using (QC-)MDPC Codes

Let t denotes the number of errors which can be corrected by a bit flipping iterative decoder of an (n, r, w)-MDPC code with $n = 2r$. Typically, we expect that $tw = O(n)$ and thus as $w = O(\sqrt{n})$, we obtain $t = O(\sqrt{n})$. The McEliece cryptosystem instantiated with (QC-)MDPC codes works as follow.

1) *Key Generation.* Generate a (two) block circulant parity check matrix $H = [H_0 \mid H_1] \in \mathbb{F}_2^{r \times n}$ with rows of weight w and such that H_1 is invertible. Compute its associated generator matrix $G = [I \mid (H_1^{-1} H_0)^T] \in \mathbb{F}_2^{(n-r) \times n}$ in systematic form. The *public key* is G and the *private key* is H.
2) *Encryption.* Let $m \in \mathbb{F}_2^{n-r}$ be the plaintext. Generate a random vector $e \in \mathbb{F}_2^n$ such that $\mathrm{wt}(e) = t$. The *ciphertext* is $c = mG + e \in \mathbb{F}_2^n$.
3) *Decryption.* Decode the ciphertext c to get a codeword mG. The *plaintext* is obtained by truncating the first $n-r$ bits of that codeword.

C. Security Assessment

Note that using LDPC codes for the McEliece scheme is deemed to be unsafe [12]. However increasing the parity check matrix density appears to have a positive effect in that respect.

1) Reduction: One of the strong feature of QC-MDPC-McEliece is its security reduction. Following [6], the system remains secure as long as:

(i) Decoding t errors in a QC $[n, n-r]$ binary linear code is hard.
(ii) Deciding whether the code spanned by some block circulant $r \times n$ matrix (orthogonal to the public key) has minimum distance $\leq w$ is hard.

Both of these problems are NP-hard in the non cyclic case [3], [13]. Their exact status is unknown in the circulant case, however there is a consensus to say that cyclicity alone will not make the problem easy. Note that the situation is very similar to lattice-based cryptography: nobody believes that the cyclic or "ring" versions of the generic lattice problems are easy, even though there is no completeness results.

2) Practical Security: In practice, the best known attacks are all based on information set decoding and its variants [14], [15], [16], [17], [18], either for decoding t errors in the code spanned by the public key or for finding words of weight w in its dual. Details can be found in [8], but typical sizes are:

- $(n, r, w, t) = (9602, 4801, 90, 84)$ for 80 bits of security
- $(n, r, w, t) = (19714, 9857, 142, 134)$ for 128 bits of security

using a (n, r, w)-QC-MDPC code correcting t errors (S bits of security means that all known attacks cost $\geq 2^S$ elementary operations).

III. Decoding QC-MDPC Codes

A. Decoding Algorithm

It is proposed in [19] to use a hard decision decoder derived from the bit flipping algorithm introduced by Gallager [20] (originally used as a LDPC decoder) to decode QC-MDPC when used to instantiate the McEliece cryptosystem. In fact, any LDPC decoding algorithms from [20] can be used as MDPC decoder as long as the error correction capability matches with the chosen parameters. Note that the solution proposed in [19] has a slightly lower capability compared to the original algorithm from [20]. However, the error rates used in practice to instantiate the McEliece cryptosystem are lower than those required in information theory. For instance, to obtain 80-bits of security, the parameters from [8] ensures an error rate around $0.87\ 10^2$. This algorithm seems well dedicated when implementing the cryptosystem in a constraint environment because it uses simple and fast operations. Actually, it already exists several embedded implementations of the cryptosystem. In [10], von Maurich and Güneysu achieved a very lightweight implementation of the McEliece scheme using QC-MDPC codes and a very high speed implementation of several decoding algorithms for QC-MDPC codes are proposed in [9].

The bit flipping algorithm is sketched in Figure 1. When H is sparse enough and x is close enough to the code of parity check matrix H, the algorithm returns the closest codeword to x. The algorithm performance depends on the choice of b's value.

Input: $H \in \mathbb{F}_2^{r \times n}$, $x \in \mathbb{F}_2^n$
$s \leftarrow xH^T$ ▷ Compute the syndrome
while $s \neq 0$ **do**
  for $i \in \{0, \ldots, n-1\}$ **do**
    $\sigma_i \leftarrow \langle s, h_i \rangle \in \mathbb{Z}$ ▷ Compute the i-th counter
    if $\sigma_i \geq b$ **then**
      $x_i \leftarrow x_i \oplus 1$ ▷ Flip the corresponding bit
    end if
  end for
  $s \leftarrow xH^T$ ▷ Update the syndrome
end while
return x

---

h_i denotes the i-th column of H.
The i-th counter σ_i is the number of unsatisfied parity check equations involving the position i.
The threshold $b \in \mathbb{N}$ depends of the iteration level, it may be precomputed or it may be computed somehow for each particular instance.

Fig. 1: Bit flipping algorithm

In Gallager's original bit flipping algorithm, one threshold value per iteration is precomputed. The formulas used to compute these thresholds guarantee some probability of decoding failure. This solution can be used to decode QC-MDPC codes but there is no certainty regarding the probability of a decoding failure.

Another solution proposed in [19] is to flip only bits that violate the maximum number of parity check equations, up to a few units.

In [9] several other tunings were proposed in order to speed up the average running time of the algorithm. Instead of using the same syndrome s for all positions while an iteration, they proposed to update the syndrome after each bit flip.

B. Security Against Timing Attacks

This algorithm is iterative and probabilistic, so we have to consider two problems. First, the fact that the number of iterations of the algorithm depends on both the error pattern and the parity check matrix, may leak some information about the secret key and thus may lead to a successful timing attack. For the moment, such attack is unknown, but to avoid it we suggest to set the number of iterations of the algorithm regardless of the instance. This number has to be large enough to ensure a negligible decoding failure probability. These decoding failures are the second problem to investigate. Somehow, a non zero probability of decoding failure may also reveal information about the secret key. Obviously, considering a decoding failure as a decoding instance which require more iteration or an infinite number of iteration, goes back to the first problem. That is to say, protecting the cryptosystem against timing attack.

In practice, the average number of iterations to decode most of the errors is very low, typically around 3 for a $(9602, 4801, 90)$-MDPC code correcting 84 errors. However, a few error patterns require much more iterations to be fully corrected. We will refer to these patterns as *worst cases* for the decoding algorithm. By abuse of language, we will refer to the *maximum number of iterations* instead of the number of iterations observed in these worst cases.

Thus, our goal is to tune the algorithm to minimize the maximum number of iteration rather than the average number of iterations and to find the better trade-off between overall running time and security.

C. Tuning the Decoder

We wish to find the best rules to tune the decoder, that is to say the best rules for computing the threshold. Moreover, we want to keep a low cost decoding procedure. Therefore the threshold computation may only involve data which is easily obtained in the regular execution of the bit flipping algorithm: the counter (number of unsatisfied parity check equations) for each position, the syndrome (and also its weight), the number of flipped bits at each iteration, ...

In the next section, we describe our approach to optimize the threshold choice.

IV. Minimizing the Number of Iterations for the Worst Cases

A. Approach

As explained in the previous section, we want to minimize the maximum number of iterations of the decoding algorithm in order to set this value for each decoding instance. In section III-A, we have highlighted that this algorithm relies on the counter values for each position of the error vector, to decide whether the position is considered as correct or wrong. Therefore, the threshold used to make a decision is of main importance. Obviously, it has a direct impact on the number of iterations needed: if the threshold is too high, just a few errors will be corrected at each iterations, and if the threshold is too low, the risk is to flip more correct bits (but whose neighbors are false) than wrong bits.

In order to minimize the maximal number of iterations, we propose to benefit from information available while decoding in order to avoid extra operations. As shown above, only few values are computed during any of the iteration: the syndrome (and its weight), the number of unsatisfied parity check equations for each position and the number of flipped bits. The syndrome weight appears to be a more relevant information about how well the decoding is processed, because it only depends on the error vector and the parity check matrix. Furthermore, one can note that the sum of the number of unsatisfied parity check equations is proportional to the syndrome weight. Thus, our idea was to take advantage of this information to adjust the threshold value for each iteration and

each instance, and decrease the maximum number of iteration required.

Our approach was to investigate a large amount of decoding traces (for each iteration, we stored the value of: the error weight, the syndrome weight, the number of flipped bit, ...) and to analyze instances requiring the highest number of iterations. Then, we tried to decrease this number for these worst cases by adapting the value of thresholds.

Our task was to obtain a global optimization, which was in fact much more challenging than trying to minimize the error weight one iteration at a time. It appears that the best solution was not necessarily the obvious one: it can happen that decreasing the number of corrected bits in the first iterations, also decrease the error weight at the last iteration. Furthermore, it reveals that the behavior of the worst cases cannot be well predicted using the weight of the syndrome, the number of unsatisfied parity check equation for each position or the number of flipped bits. Still, these information helped us to analyze the decoding process.

The result of these investigations is that computing the thresholds as a function of the syndrome weight, regardless to the current iteration, seems to be the better solution to achieve the lowest maximum number of iterations.

B. Results

The Figure 2 shows the step function used to compute the threshold according to the syndrome weight that appears to be the best choice in order to decrease the maximum number of iterations.

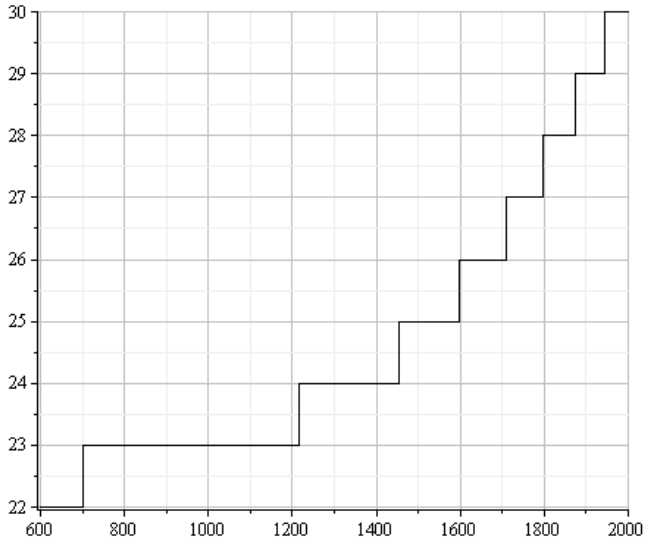

Fig. 2: Computation of the threshold values as a function of the syndrome weight for $(9602, 4801, 90)$-QC-MDPC codes in order to correct up to 84 errors.

As mentioned in [11], the minimal average number of iteration is achieved by updating the syndrome after each bit flip and by using precomputed thresholds. However, it is also noticed that the higher performance decoding algorithm can diverge for some particular instances. That is to say, some error patterns cannot be decoded at all, and a second decoding attempt has to be done, using different threshold values. This does not affect much the average running time of decoding but does not permit the design of a (reasonable) constant time decoder.

We compare in Table I the behavior of our worst case decoder and the most efficient (on average) algorithm from [9].

Iteration	This work	[9] algorithm
2	$0.187\ 10^{-5}$	0.536
3	0.768	0.462
4	0.231	$0.145\ 10^{-2}$
5	$0.408\ 10^{-3}$	$0.201\ 10^{-3}$
6	$0.321\ 10^{-5}$	$0.152\ 10^{-5}$
7	$0.150\ 10^{-6}$	$0.200\ 10^{-7}$
∞[1]	0	$0.948\ 10^{-5}$

TABLE I: Proportion of errors patterns which are decoded in a fixed number of iterations. We are comparing the result of our work and the new algorithm from [9] which used precomputed thresholds and abort iteration when the syndrome becomes zero. We used $(9602, 4801, 90)$-QC-MDPC codes and errors of weight 84. The simulations were made over 1000 codes with 10^5 error patterns per code for our work and 500 codes with 10^5 error patterns per code for the algorithm from [9]

Note that, we do not find cases of divergence using our decoding technique. These results suggest that the adjustment of the threshold improves the worst case decoding but decreases the efficiency of the algorithm for average cases. However, our goal was to minimize the maximum number of iterations, which is 7 for these parameters and not to speed up the average running time of the decoder. Although we did not find instances which are not decoded after the 7^{th} iterations, we cannot guarantee the decoding failure probability to be zero. Hence, it is probably safer to fix the number of iterations to 8 or 9 on practice. An other viable solution could be to slightly increase the length and the dimension of the code, keeping the same code rate and the same values for w and t. This would improve the overall efficiency of the decoder.

V. Conclusion

Obtaining a constant time decoder for QC-MDPC is a safeguard against timing attacks on the QC-MDPC-McEliece scheme. We show here that the best constant-time decoder requires a specific tuning of the bit flipping algorithm which relies on the syndrome weight. A limitation of our approach is that a significant effort is needed to find the optimal threshold rule for every set of code parameters. We think however our work gives a serious hint on how to find the thresholds for others code parameters.

Finding theoretical bounds or estimates of the decoding failure probability seems to be a challenging task, but for cryptographic purpose, it could be of interest to reduce that probability to a very small amount, say 2^{-128}, with some kind

[1]Proportion of errors which cannot be decoded.

of guarantee. We hope that our preliminary work could open the way to such an achievement.

REFERENCES

[1] P. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *FOCS*, S. Goldwasser, Ed., 1994, pp. 124–134.

[2] R. J. McEliece, *A Public-Key System Based on Algebraic Coding Theory*. Jet Propulsion Lab, 1978, pp. 114–116, dSN Progress Report 44.

[3] E. Berlekamp, R. McEliece, and H. van Tilborg, "On the inherent intractability of certain coding problems," *IEEE Trans. Inform. Theory*, vol. 24, no. 3, pp. 384–386, May 1978.

[4] J.-C. Faugère, V. Gauthier, A. Otmani, L. Perret, and J.-P. Tillich, "A distinguisher for high rate McEliece cryptosystems," in *Proc. IEEE Inf. Theory Workshop- ITW 2011*, Paraty, Brasil, Oct. 2011, pp. 282–286.

[5] P. Gaborit, "Shorter keys for code based cryptography," in *Proceedings of the 2005 International Workshop on Coding and Cryptography (WCC 2005)*, Bergen, Norway, Mar. 2005, pp. 81–91.

[6] N. Sendrier, "On the use of structured codes in code based cryptography," in *Coding Theory and Cryptography III*, L. S. S. Nikova, B. Preneel, Ed. The Royal Flemish Academy of Belgium for Science and the Arts, 2010, pp. 59–68.

[7] J.-C. Faugère, A. Otmani, L. Perret, and J.-P. Tillich, "Algebraic cryptanalysis of McEliece variants with compact keys," in *Advances in Cryptology - EUROCRYPT 2010*, ser. Lecture Notes in Comput. Sci., vol. 6110, 2010, pp. 279–298.

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

[9] S. Heyse, I. von Maurich, and T. Güneysu, "Smaller keys for code-based cryptography: QC-MDPC McEliece implementations on embedded devices," in *Cryptographic Hardware and Embedded Systems - CHES 2013*, ser. Lecture Notes in Comput. Sci., G. Bertoni and J. Coron, Eds., vol. 8086. Springer, 2013, pp. 273–292.

[10] I. von Maurich and T. Güneysu, "Lightweight code-based cryptography: QC-MDPC McEliece encryption on reconfigurable devices," in *Design, Automation & Test in Europe Conference & Exhibition, DATE 2014, Dresden, Germany, March 24-28, 2014*, 2014, pp. 1–6. [Online]. Available: http://dx.doi.org/10.7873/DATE.2014.051

[11] I. V. Maurich, T. Oder, and T. Güneysu, "Implementing QC-MDPC McEliece encryption," *ACM Trans. Embed. Comput. Syst.*, vol. 14, no. 3, pp. 44:1–44:27, Apr. 2015. [Online]. Available: http://doi.acm.org/10.1145/2700102

[12] C. Monico, J. Rosenthal, and A. A. Shokrollahi, "Using low density parity check codes in the McEliece cryptosystem," in *Proc. IEEE Int. Symposium Inf. Theory - ISIT*, Sorrento, Italy, 2000, p. 215.

[13] A. Vardy, "The intractability of computing the minimum distance of a code," *IEEE Trans. Inform. Theory*, vol. 43, no. 6, pp. 1757–1766, Nov. 1997.

[14] E. Prange, "The use of information sets in decoding cyclic codes," *IRE Transactions on Information Theory*, vol. 8, no. 5, pp. 5–9, 1962. [Online]. Available: http://dx.doi.org/10.1109/TIT.1962.1057777

[15] J. Stern, "A method for finding codewords of small weight," in *Coding Theory and Applications*, ser. Lecture Notes in Comput. Sci., G. D. Cohen and J. Wolfmann, Eds., vol. 388. Springer, 1988, pp. 106–113.

[16] A. May, A. Meurer, and E. Thomae, "Decoding random linear codes in $O(2^{0.054n})$," in *Advances in Cryptology - ASIACRYPT 2011*, ser. LNCS, D. H. Lee and X. Wang, Eds., vol. 7073. Springer, 2011, pp. 107–124.

[17] A. Becker, A. Joux, A. May, and A. Meurer, "Decoding random binary linear codes in $2^{n/20}$: How $1 + 1 = 0$ improves information set decoding," in *Advances in Cryptology - EUROCRYPT 2012*, ser. Lecture Notes in Comput. Sci. Springer, 2012.

[18] A. May and I. Ozerov, "On computing nearest neighbors with applications to decoding of binary linear codes," in *Advances in Cryptology - EUROCRYPT 2015*, ser. LNCS, E. Oswald and M. Fischlin, Eds., vol. 9056. Springer, 2015, pp. 203–228.

[19] R. Misoczki, J.-P. Tillich, N. Sendrier, and P. S. L. M. Barreto, "MDPC-McEliece: New McEliece variants from moderate density parity-check codes," *IACR Cryptology ePrint Archive, Report2012/409*, vol. 2012, 2012.

[20] R. G. Gallager, *Low Density Parity Check Codes*. Cambridge, Massachusetts: M.I.T. Press, 1963.