Error Amplification in Code-based Cryptography

Alexander Nilsson[1,2*], Thomas Johansson[1*†] and Paul Stankovski Wagner[1*†]

[1] Dept. of Electrical and Information Technology, Lund University, Sweden
[2] Advenica AB, Malmö, Sweden
{alexander.nilsson,thomas.johansson,paul.stankovski}@eit.lth.se

Abstract. Code-based cryptography is one of the main techniques enabling cryptographic primitives in a post-quantum scenario. In particular, the MDPC scheme is a basic scheme from which many other schemes have been derived. These schemes rely on iterative decoding in the decryption process and thus have a certain small probability p of having a decryption (decoding) error.

In this paper we show a very fundamental and important property of code-based encryption schemes. Given one initial error pattern that fails to decode, the time needed to generate another message that fails to decode is strictly *much* less than $1/p$. We show this by developing a method for fast generation of undecodable error patterns (error pattern chaining), which additionally proves that a measure of closeness in ciphertext space can be exploited through its strong linkage to the difficulty of decoding these messages. Furthermore, if side-channel information is also available (time to decode), then the initial error pattern no longer needs to be given since one can be easily generated in this case.

These observations are fundamentally important because they show that a, say, 128-bit encryption scheme is not inherently safe from reaction attacks even if it employs a decoder with a failure rate of 2^{-128}. In fact, unless explicit protective measures are taken, having a failure rate at all – of any magnitude – can pose a security problem because of the error amplification effect of our method.

A key-recovery reaction attack was recently shown on the MDPC scheme as well as similar schemes, taking advantage of decoding errors in order to recover the secret key. It was also shown that knowing the number of iterations in the iterative decoding step, which could be received in a timing attack, would also enable and enhance such an attack. In this paper we apply our error pattern chaining method to show how to improve the performance of such reaction attacks in the CPA case. We show that after identifying a single decoding error (or a decoding step taking more time than expected in a timing attack), we can adaptively create new error patterns that have a much higher decoding error probability than for a random error. This leads to a significant improvement of the attack based on decoding errors in the CPA case and it also gives the strongest known attack on MDPC-like schemes, both with and without using side-channel information.

Keywords: post-quantum cryptography · MDPC · timing attack · side-channel attack · iterative decoding · error amplification · error pattern chaining

*This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation

†Sponsored in part by the Swedish Foundation for Strategic Research (SSF) strategic mobility grant SM17-0062.

IACR Transactions on Cryptographic Hardware and Embedded Systems ISSN 2569-2925, Vol. 2019, No. 1, pp. 238–258
DOI:10.13154/tches.v2019.i1.238-258

1 Introduction

Future quantum computers will be able to break cryptography based on integer factorization and discrete log in polynomial time. This fact pushed cryptographic research to focus on post-quantum solutions, i.e., finding new primitives based on more well suited mathematical problems that may still be difficult to solve for a quantum computer. That is, the new primitives must be efficiently computable on classical computers, but an adversary must not be able to break the scheme even if she has access to a powerful quantum computer. This area of research is called post-quantum cryptography [Ber09].

Code-based cryptography leverages difficult problems in coding theory and is one of the main techniques enabling cryptographic primitives in this post-quantum scenario. The classic idea in code-based cryptography is the McEliece scheme from 1978 [McE78]. However, recently much more attractive schemes have been proposed, centered around the quasi-cyclic medium density parity check (QC-MDPC) scheme [MTSB13]. It has a significantly smaller public key than the McEliece scheme. The QC-MDPC scheme is, like many other similar variants of code-based cryptographic encryption schemes, based on iterative decoding; it has a certain probability of decryption failure (for proposed parameters in the range of 10^{-4} to 10^{-8} [HvMG13], depending on decoding algorithm, implementation and chosen parameters). Also, as the decryption step is iterative, it will require a varying number of rounds before finishing, which leads to varying decoding time in a standard implementation. This fact opens the door for possible timing attacks based on the number of required decoding rounds. It was shown by Guo, Johansson and Stankovski [GJS16], that decoding errors can be used to reconstruct the secret key. The attack breaks the chosen-ciphertext attack (CCA) security of the scheme and provides attackers with a key recovery attack that requires submitting 200-350 million ciphertexts for decryption. This was based on proposed parameters for 80-bit security and the decryption device using an iterative decoding algorithm with a decoding error probability around 10^{-4}. Better decoding algorithms (with higher complexity) could lower the decoding error and this would lead to an increase in the complexity of the attack.

Let us briefly recall the idea behind the Guo et al. attack. The authors identified a dependency between the secret key and decoding failures. They found that if there were two ones in the key at (cyclic) distance d and the error pattern also contained two ones at a distance d, then the probability for decoding error is smaller than in the opposite case. This observation was used to build a so called distance spectrum of the secret key. A distance spectrum can be viewed as the set of all distances between any two non-zero bit positions in the key. In order to build this distance spectrum, the authors simply aggregated the distance spectrum of each bit pattern in messages which led to a decoding failure. This distance spectrum can, in a reconstruction step, be used to directly determine the secret key.

In the recent NIST post-quantum standardization project [CJL+16], a number of code-based schemes have been submitted. Looking through these submissions, one can see that the above described attack has impact on the security analysis of such schemes. The attack has also been generalized, analyzed and improved in different directions. In [FHS+17] the attack was extended to break the QC-LDPC McEliece scheme presented in [BBC08]. A similar attack on LEDApkc appears in [FHZ].

In the Guo et al. paper, it was pointed out that the attack would also extend to a timing attack. This was fully examined by Eaton et al. in [ELPS18]. Not only did they provide the framework for a timing attack, but they also gave an extended theoretical treatment of the attack and showed the dependence on the syndrome weight in decoding.

1.1 Contributions

In this paper we show a method for generating large quantities of error patterns that fail to decode for any given iterative decoder. Given one initial error pattern that fails to decode, the time needed to generate another message that fails to decode is almost negligible. Our method for fast generation of undecodable error patterns (error pattern chaining) additionally proves that there is a measure of closeness in ciphertext space such that similar ciphertext messages are roughly equally difficult to decode. In addition, when side-channel information such as decoding time or number of iteration used during decoding is also available, then the initial error pattern no longer needs to be given since one can easily be generated instead.

These observations are fundamentally important because they show that a, say, 128-bit encryption scheme is not inherently safe from reaction attacks even if it employs a decoder with a failure rate of 2^{-128}. Extrapolating from the general ideas of this paper and those of Guo et al., it would seem that failure rates, regardless of their magnitude, appear to convey a potential for security problems. We further argue that using very low failure rates is, on its own, not enough to discount such weaknesses, in any cryptographic system.

We also apply our findings towards improving the attacks of Guo et al. and Eaton et al. We use error pattern chaining to explore the possibility of artificially and adaptively increasing the error probability and improve on previous works by extracting much more information from *all* decoding attempts. We do this in a chosen plaintext attack (CPA) setting. The attacks are improved in two ways: 1) it increases the possibility of finding another non-decodable error pattern and 2) it enables us to extract more information from patterns that *can* be decoded. If we additionally consider a timing attack we can improve the attack even further, both in finding initial error patterns and in performing the main attack.

Simulations show that a distance spectrum can be built from this chain of errors in a similar manner as in the original attack. This method enables us to use more than an order of magnitude fewer decoding trials to recover the secret key in the CPA setting, compared to the original attack of Guo et al. in [GJS16].

Comparing our work with the recent work of Eaton et al. [ELPS18], we improve upon their original attack even *without* using private information such as the syndrome weight. When Eaton et al. convert their attack to a timing-based side-channel attack, they require $2^{25} \approx 33.5$M ciphertexts to fully recover the key for standard parameters of 80-bit security. Using our idea of chains of related error patterns, our new attack requires less than 12M ciphertexts without using any side-channel information, and less than 8M ciphertexts with side-channel information.

Taking into consideration the fact that [GJSW] has shown that a fully correct distance spectrum is not necessary to recover the secret key, we show that as few as 310 000 ciphertexts are necessary to perform a successful attack, using a distance spectrum with 900 errors, for 80-bit security.

It is also clear that this new adaptive approach will be even more beneficial when the decoding error is very small, which could be expected in a scheme proposed for actual use.

1.2 Paper Organization

In Section 2 we give some background of code-based cryptography, QC-MDPC and the original reaction attack by Guo et al. [GJS16]. In Section 3 we provide the theory behind our new method of performing error rate amplification. We also present how to use this method in an attack against QC-MDPC. Then, in Section 4, we describe the results we obtained by implementing the new method and testing it against a few different decoder implementations. Finally, we conclude the paper in Section 5.

2 Background

In this section we briefly give some of the basic background information necessary to follow this paper.

2.1 Coding Theory and Public-Key Cryptography

We review some basics from coding theory and show its application to public-key cryptography.

Definition 1 (Linear codes). *An $[n, k]$ linear code $\mathcal{C}$ over a finite field $\mathbb{F}_q$ is a linear subspace of $\mathbb{F}_q^n$ of dimension k.*

Definition 2 (Generator matrix). *A $k \times n$ matrix G with entries from $\mathbb{F}_q$ having rowspan $\mathcal{C}$ is a generator matrix for the $[n, k]$ linear code $\mathcal{C}$.*

The code $\mathcal{C}$ is the kernel of an $(n-k) \times n$ matrix H called a *parity-check matrix* of $\mathcal{C}$. We have $\mathbf{c}H^{\mathrm{T}} = \mathbf{0}$, if and only if $\mathbf{c} \in \mathcal{C}$, where H^{T} is the transpose of H.

The code $\mathcal{C}$ can be represented by different generator matrices. An important one is the systematic form, i.e., when each input symbol is directly represented in a position in the codeword. One can find a $k \times k$ submatrix of G forming the identity matrix and after a permutation one can consider G of the form $G = \begin{pmatrix} I & P \end{pmatrix}$. If G has such a systematic form then $H = \begin{pmatrix} -P^{\mathrm{T}} & I \end{pmatrix}$.

We now only consider binary codes, i.e. $q = 2$. The Hamming weight $w_H(\mathbf{x})$ of a binary vector in $\mathbf{x} \in \mathbb{F}_2^n$ is the number of nonzero entries in the vector. The minimum (Hamming) distance of the code $\mathcal{C}$ is defined as $d = \min_{\mathbf{x},\mathbf{y} \in \mathcal{C}} w_H(\mathbf{x} - \mathbf{y})$, where $\mathbf{x} \neq \mathbf{y}$.

Definition 3 (Quasi-cyclic codes). *An $[n, k]$ quasi-cylic (QC) code $\mathcal{C}$ is a linear code such that for some integer n_0, every cyclic shift of any codeword by n_0 steps is again a codeword.*

In particular, if $n = n_0 k$ for a QC code, then a generator matrix of the form

$$G = \begin{pmatrix} I & P_0 & P_1 & \cdots & P_{n_0-1} \end{pmatrix}$$

is a possible representation of a QC code, where P_i is a $k \times k$ cyclic matrix, i.e. the rows (or columns) of P are obtained by cyclic rotations of the first row. Also, the algebra of $k \times k$ binary circulant matrices is isomorphic to the algebra of polynomials modulo $x^k + 1$ over $\mathbb{F}_2$, allowing an alternative description.

Another useful class of codes is the class of Low-Density Parity-Check codes (LDPC codes), defined as linear codes that admit a sparse parity-check matrix H, where sparsity means that each row of H has at most w ones, for some small w. This sparse matrix can be represented in the form of a bipartite graph, that consists of $n-k$ upper nodes (named "check nodes") representing the $n-k$ parity equations and n lower nodes (named "variable nodes") representing the n codeword positions. A variable node is connected to a check node if the variable is present in that parity check. Each check node is then connected to w variable nodes. We call this graph representation a "Tanner" graph, which is a frequently used term in work on iterative decoding algorithms.

2.2 McEliece Cryptosystem

In 1978 McEliece showed how a public key cryptosystem (PKC) could be constructed using tools from coding theory. We briefly describe the original McEliece PKC here. This scheme uses three matrices G, S, P, where G is a $k \times n$ generator matrix of a binary $[n, k, 2t+1]$ linear code. The original and still secure proposal in [McE78] is to use Goppa codes (see [Gop70, MS77]). Then S is a $k \times k$ random binary non-singular matrix (called the

scrambling matrix), and P is an $n \times n$ random permutation matrix (called the permutation matrix). As designers we compute the new $k \times n$ matrix $G' = SGP$. The scheme works as follows. The private key is (G, S, P) and the public key is (G', t). In encryption, a message $\mathbf{m}$ is mapped to a ciphertext $\mathbf{c}$ by $\mathbf{c} = \mathbf{m}G' + \mathbf{e}$, where $\mathbf{c}$ is the n-bit ciphertext, $\mathbf{m}$ is the k-bit plaintext and $\mathbf{e}$ an n-bit error vector with (Hamming) weight t. In decryption, one uses an efficient decoding algorithm for Goppa codes to decode $\mathbf{c}$ to find the error $\mathbf{e}P^{-1}$, and recover $\mathbf{m}S$ and thus $\mathbf{m}$.

Knowing the description of the selected Goppa code G allows efficient decoding, as there are many efficient decoding algorithms for this problem running in polynomial time. But knowing only the public key G', the attacker is facing a decoding problem for a code that looks like a random code, which is a difficult problem. The attacker can either try to decode an intercepted ciphertext (message recovery attack) or try to recover the secret matrix G from the public matrix G' (key recovery attack).

2.3 The QC-MDPC Public Key Encryption System

In [MTSB13], a new powerful version of the McEliece PKC was proposed. It has a simpler description as it does not use permutation and scrambling matrices as in the original McEliece construction or in other proposed generalizations [BCGM07, LJ12]. The idea is to use codes that allow iterative decoding. In coding theory, this typically involves LDPC codes, but for a cryptographic scheme this is not secure. LDPC codes have parity-checks with very small Hamming weight and such parity-checks in a given LDPC code correspond to codewords in the dual code. Since the dual code can be computed, it is also easy to find low-weight codewords in the dual code and thus the low-weight parity checks. The solution proposed in [MTSB13] is to increase the weight of the parity checks to a larger value, but still small in comparison with the dimension of the code. This makes the task of finding low-weight codewords in the dual code much more costly. In this way, key-recovery attacks by searching for low weight codewords are avoided.

Such codes with increased parity-check weight are called *Moderate-Density Parity-Check* codes (MDPC codes), and they can be decoded with the same decoding algorithms used to decode LDPC codes. The quasi-cyclic variant of MDPC codes are called QC-MDPC codes. These are of special interest, since the quasi-cyclic property allows us to represent the code by a single row of the generator matrix. Since the public key is a generator matrix, this gives us very compact keys. We will now describe the different steps of the QC-MDPC public key cryptosystem as proposed in [MTSB13]. We will restrict ourselves to $n_0 = 2$, corresponding to parameters $r = k = n/2$.

2.3.1 Key Generation

1. Choose an $[n, n/2]$ code in the QC-MDPC family, described by the parity-check matrix $H \in \mathbb{F}_2^{r \times n}$, such that
$$H = \begin{pmatrix} H_0 & H_1 \end{pmatrix},$$
where each H_i is a circulant $r \times r$ matrix with weight w_i in each row and with $w = \sum w_i$.

2. Generate the public key $G \in \mathbb{F}_2^{(n-r) \times n}$ from H as,
$$G = \begin{pmatrix} I & P \end{pmatrix},$$
where
$$P = \left(\left(H_1^{-1} H_0 \right)^{\mathrm{T}} \right).$$

Recall, the QC-MDPC scheme has no need for permutation or scrambling matrices.

2.3.2 Encryption

Let $\mathbf{m} \in \mathbb{F}_2^{(n-r)}$ be the plaintext. Multiply $\mathbf{m}$ with the public key G and add noise within the correction radius t of the code, i.e., $\mathbf{c} = \mathbf{m}G + \mathbf{e}$, where $w_H(\mathbf{e}) \leq t$. The parameter t is obtained from the error correcting capability of the decoding algorithm for the MDPC code [MTSB13]. The error vector is uniformly chosen among all binary n-tuples with $w_H(\mathbf{e}) \leq t$.

2.4 Decryption

Let $\mathbf{c} \in \mathbb{F}_2^n$ be a received ciphertext. Given the secret low-weight parity check matrix H, a low-complexity decoding procedure is used to obtain the plaintext $\mathbf{m}$.

The authors of [MTSB13] propose the use of Gallager's bit-flipping algorithm [Gal62] for the decoding of MDPC codes. This bit-flipping procedure is vital to the proposed key recovery attack under consideration. The decoding step works as follows:

1. Compute the syndrome $\mathbf{s} = \mathbf{c}H^T$. Since $\mathbf{m}H^T = \mathbf{0}$, this is equivalently expressed as $\mathbf{s} = \mathbf{e}H^T$. Consider the Tanner graph corresponding to H. Create a counter with an initial value 0 for each variable node.

2. Run through all parity-check equations (rows of H and/or check nodes in the graph) and for every variable node connected to an unsatisfied check node, increase its corresponding counter by one.

3. Run through all variable nodes and flip its value if its counter is larger than a predetermined threshold δ_j.

4. If all the equations are satisfied, or iteration counter j reached a maximum value, then stop; otherwise, set all counters to 0, increase j by one, and go to Step 2.

The decoding procedure will stop if all the parity-checks are satisfied or if the limit on the maximum number of iterations is reached.

It is well known that this iterative decoding algorithm used with LDPC codes has an error-correction capability that increases linearly with the length of the code. MDPC codes have slightly higher parity-check weight than LDPC codes and this slightly influences the error-correction capability in a negative way. So the actual performance of this algorithm on MDPC codes is relatively poor compared with that on LDPC codes. More details on the decoding performance can be found in [MTSB13]. Other variants of this decoding algorithm have been proposed [HvMG13, MOG15], to reduce the decoding error probability, involving changing the flipping and the thresholds, introducing more rounds, or other techniques. The error probability for proposed parameters is still large, for any decoding algorithm, compared to the corresponding security level. For 80-bit security, it is typically in the range $10^{-4} - 10^{-8}$, whereas a value of 2^{-80} would be required to be more sure that decoding errors would not be a tool in cryptanalysis.

2.5 Proposed Parameters

In [MTSB13], the following parameters were proposed for a QC-MDPC scheme with 80-bit, 128-bit and 256-bit security level.

Implementations of the QC-MDPC scheme [HvMG13, MOG15] and a variant [vMHG16] demonstrate excellent efficiency in terms of computational complexity and key sizes for encryption and decryption on constrained platforms such as embedded micro-controllers and FPGAs.

Table 1: Proposed QC-MDPC instances with key size and security level.

Parameters						
n	r	w	t	n_0	Key size	Security
9602	4801	90	84	2	4801	80
19714	9857	142	134	2	9857	128
65542	32771	274	264	2	32771	256

2.5.1 Key Recovery Attack

A major attack on the QC-MDPC scheme was recently described in [GJS16]. In the underlying setting, it uses the assumption that Mallory, taking the role of Alice, can observe the reaction of Bob, i.e. whether the decryption step was successful or if there was a decoding error. This is a natural assumption, as if Bob received a decoding error he would typically need to ask for a retransmission of the message in some way.

The attack in [GJS16] recovers the secret key. The objective of the key recovery attack is to recover the parity check matrix H (knowing only G). Obviously, H can be easily be derived from any H_i, thereby reducing the problem to finding H_0. Since H_0 is a circular matrix, it is enough to recover the first row h_0.

The key idea is to examine the decoding result for different error patterns. In particular, Mallory will pick errors from special subsets. Let Ψ_d be the set of all binary vectors of length $n = 2r$ having exactly t ones, where all ones are placed with distance d in the first half of the vector. The second half of the vector is all zero. The set Ψ_d guarantees repeated ones at distance d at least $t/2$ times, where

$$\begin{aligned}\Psi_d \quad = \quad & \{\mathbf{v} = (\mathbf{e}, \mathbf{f}) \mid w_H(\mathbf{f}) = 0, \text{ and } \exists \text{ distinct } s_1, s_2, \ldots, s_t, \text{ s.t. } \mathbf{e}_{s_i} = 1, \text{ and} \\ & s_{2i} = (s_{2i-1} + d) \bmod r \text{ for } i = 1, \ldots, \frac{t}{2}, \text{ and } w_H(\mathbf{e}) = t\}.\end{aligned}$$

Mallory will now send M messages to Bob, using QC-MDPC with the error selected from the subset Ψ_d. When there is a decoding error with Bob, she will record this and after M messages she will be able to compute an empirical decoding error probability for the subset Ψ_d. Furthermore she will do this for $d = 1, 2, \ldots, U$ for some suitable upper bound U.

Algorithm 1 Computing the distance spectrum

Input: parameters n, r, w and t of the underlying QC-MDPC code, number of decoding trials M per distance.

Output: distance spectrum $D(\mathbf{h}_0)$ (multiplicity vector).

1: **for** all distances d **do**
2: Try M decoding trials using the designed error pattern
3: Perform statistical test to decide multiplicity $\mu(d)$
4: Set position d in $D(\mathbf{h}_0)$ to the multiplicity $\mu(d)$
5: **end for**

The main observation is that there is a strong correlation between the decoding error probability for error vectors from Ψ_d and the existence of a distance d between two ones in the secret vector $\mathbf{h}_0$. If there exists two ones in $\mathbf{h}_0$ at distance d, the decoding error probability is much smaller than if distance d does not exist between two ones.

After sending $M \times U$ messages, we look at the decoding error probability in each Ψ_d and classify each d, $d = 1, 2, \ldots, U$ according to its multiplicity, since each distance can appear many times. We denote the multiplicity as $\mu(d)$. This provides a distance spectrum

for $\mathbf{h}_0$, which we denote as $D(\mathbf{h}_0)$. In this paper the distance spectrum is presented as a *multiplicity vector* of length U (as opposed to a set, which was used in [GJS16]) and defined as

$$D(\mathbf{h}_0) = (\mu(1), \mu(2), \ldots, \mu(U)).$$

As an example from [GJS16], for the bit pattern $\mathbf{c} = 0011001$ we have $U = 3$ and

$$D(\mathbf{c}) = (1, 0, 2)\,,$$

with distance multiplicities $\mu(1) = 1, \mu(2) = 0$ and $\mu(3) = 2$.

The procedure for computing the distance spectrum is specified in Algorithm 1.

The final step is to do derive $\mathbf{h}_0$ from knowing the distance spectrum $D(\mathbf{h}_0)$. This is rather straightforward. Start by assigning the first two ones in a length i_0 vector in position 0 and i_0, where i_0 is the index of the smallest non-zero value in the empirical $D(\mathbf{h}_0)$. Then put the third one in a position and test if the two distances between this third one and the previous two ones both appear in the distance spectrum. If they do not, we test the next position for the third bit. If they do, we move to test the fourth bit and its distances to the previous three ones, etc. After reconstruction, we have restored $\mathbf{h}_0$. Reconstruction is possible even if some smaller fraction of entries in the empirical $D(\mathbf{h}_0)$ are wrong and/or are missing. We refer to [GJS16] for more details on the reconstruction part.

The attack was implemented and tested on a proposed instance of QC-MDPC for 80-bit security. It used about 240M ciphertexts (in the CPA-attack scenario) and successfully recovered the secret key with low computational complexity.

3 A New Improved Attack through a Chaining Method for Error Vectors

The main contribution of this paper is to propose a new adaptive way of selecting error vectors (patterns) which provides an improved attack for the CPA scenario. We introduce this new chaining method as a way to artificially increase the decoding error rate for codes that use iterative decoders. The chaining method works by leveraging the knowledge of a single error pattern into finding a new pattern that is similar to the first pattern. By repeating the process a chain of similar error patterns is created.

This section will provide a detailed description of both the chaining method and how it might be used as an attack on the QC-MDPC scheme. We will also discuss how side channel information might be used to improve the efficiency of the attack. But first the attack models used in this paper are presented.

3.1 Attack Models

The following attack scenario is assumed. Bob continuously receives messages, encrypted with his public key, from Mallory. Mallory is able to detect each time Bob fails to decrypt any of the messages sent by her. By crafting the messages sent to Bob in a certain way and doing it a number of times it is possible for Mallory to recover Bob's secret key. This is a so called *reaction attack*, it is similar in nature to an *adaptive chosen ciphertext attack* (CCA2), but requires only to know decryption success or failure, not the result of the decryption. Thus, this attack model requires weaker assumptions than CCA2, but stronger than CPA (*chosen plaintext attack*). This scenario is very similar to the one described by Guo et al. in [GJS16], although the attacks described here only targets the case when the error can be freely chosen by Mallory. For definitions on CPA, CCA and CC2 we refer the reader to [S+03].

A version of the new attack, where side-channel information is used is also included. This attack version requires an attack model which additionally allows timing information to leak about the decoding such that the number of iterations can be determined. This would be provided by a decoder implementation if it is not running in constant time.

3.2 Description of the Basic Attack

The chaining method for generation of error patterns works by first finding an initial error pattern $\mathbf{e}_0$, that fails to be decoded. We do this by random selection and trial until one is found. It will be shown later that the finding of $\mathbf{e}_0$ can be improved upon by utilizing side-channel information (see Section 3.5). Let an error pattern $\mathbf{e}_i$ be given, causing a

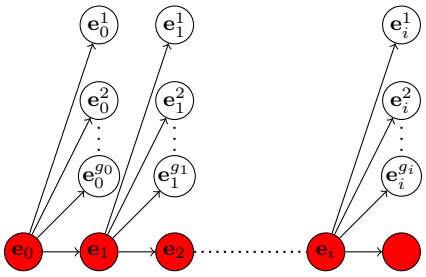


Figure 1: Visualization of error chains. White and red nodes indicate error patterns that can and cannot be decoded, respectively. Starting from an undecodable error pattern $\mathbf{e}_0$, we successively generate several new error patterns $\mathbf{e}_0^g$ by swapping a randomly chosen 0 and 1 in $\mathbf{e}_0$. Once a new undecodable error pattern $\mathbf{e}_1$ is found, it is used as the new base for generating new error patterns. By iterating this procedure, we end up with a (red) chain of undecodable error patterns.

decoding error. It is known that if a particular distance d in the distance spectrum $D(\mathbf{e}_i)$ does *not* exist in the key represented by the secret vector $D(\mathbf{h}_0)$ (e.g. the value of position d in $D(\mathbf{h}_0)$ is zero), then the probability of decoding failure is increased [GJS16]. This means that if we, somehow, can find another similar error pattern, denoted $\mathbf{e}_{i+1}$, that also fails to be decoded, then the *differences* between the distance spectra of these two error patterns should also contribute to error patters which are harder to decode than the average. We can detect this by measuring an increased decoding failure rate for such patterns.

We use the following notation. A distance spectrum of a binary vector $\mathbf{e}$ of length $2U$ is written as $D(\mathbf{e}) = (D_1(\mathbf{e}), D_2(\mathbf{e}), \ldots, D_U(\mathbf{e}))$. The difference between two spectra for vectors $\mathbf{e}_i$, $\mathbf{e}_{i+1}$ is defined as the element-wise subtraction of the two vectors, denoted $D(\mathbf{e}_i) - D(\mathbf{e}_{i+1})$. We simplify by writing for each distance d, where $1 \leq d \leq U$,

$$\Delta D_d = D_d(\mathbf{e}_i) - D_d(\mathbf{e}_{i+1})$$

In short ΔD_d indicates the change in the multiplicity for the specific distance d, when comparing $D(\mathbf{e}_{i+1})$ to $D(\mathbf{e}_i)$.

It is possible to find an error pattern similar to $\mathbf{e}_i$ by randomly[1] bit-permuting a copy of $\mathbf{e}_i$. In this copy, denoted $\mathbf{e}'$, both a single bit position containing the value 1 and a single bit position with the value 0 are randomly selected. These bits are then flipped (effectively moving a single bit in the pattern). Afterwards, the new error pattern is used

[1] We wish to remark that there may be other methods of finding modifications that might be even more effective at amplifying the error rate. Search heuristics such as evolutionary algorithms or other methods are applicable here.

in the encryption and the corresponding ciphertext is subsequently sent to Bob. If $\mathbf{e}'$ can be successfully decoded (CASE-0) then it is discarded and we create a new $\mathbf{e}'$ by again modifying a copy of $\mathbf{e}_i$. If the decoding fails (CASE-1) then $\mathbf{e}_{i+1} \leftarrow \mathbf{e}'$ and the bit-permutation procedure is repeated with $\mathbf{e}_{i+1}$ instead being the new base error pattern (and we increment i). See Algorithm 2 and Figure 1,

For CASE-1, a distance d is less likely to be added (e.g. $D_d(\mathbf{e}_{i+1}) > D_d(\mathbf{e}_i)$) if d also exists as a distance in h_0. We introduce the vector F to represent the aggregated distance spectrum of all additions to the spectrum (i.e. $\Delta D_d > 0$). Similarly the vector G represents the combined distance spectrum of all distance removals from the patterns in the chain. Running through all differences between consecutive error patterns $\mathbf{e}_{i+1}, \mathbf{e}_i$ in the created chain, for $i = 1, 2, \ldots$, we update F and G as

$$\begin{cases} F_d \leftarrow F_d + \Delta D_d & \textbf{if } \Delta D_d > 0, \\ G_d \leftarrow G_d - \Delta D_d & \textbf{if } \Delta D_d < 0, \\ \text{no change} & \textbf{if } \Delta D_d = 0, \end{cases} \qquad \forall d \in \{1, \ldots, U\}, \tag{1}$$

By combining the results of M decoding attempts (ciphertexts) the vectors F and G accumulate all changes to the distance spectra, which in turn gives us combined distance spectra of statistical significance (provided M is large enough).

For CASE-0 our simulations show that it is not a simple matter of reversing the logic above; Considering that $\mathbf{e}'$ now leads to successful decoding, additions (in respect to $\mathbf{e}_i$) to the distance spectrum could either be a large contributor towards the decoding success or a small one. A large contribution would indicate a higher probability of being represented in the distance spectrum of $\mathbf{h}_0$ and a small contribution would indicate a higher probability of *not* being represented. It will be shown in later sections that the number of iterations required for decoding plays a large role of how to interpret distance additions and deletions. The basic attack, as described in this section, ignore this and simply perform the same operations described above for CASE-1, although with different output vectors and different input. The outputs are stored into vectors A and B and for input we use those error pattern which resulted in decoding successes, instead of failures. So for all $\mathbf{e}'$ with decoding success we consider $\Delta D = D(\mathbf{e}_i) - D(\mathbf{e}')$ and update vectors A and B as

$$\begin{cases} A_d \leftarrow A_d + \Delta D_d & \textbf{if } \Delta D_d > 0, \\ B_d \leftarrow A_d - \Delta D_d & \textbf{if } \Delta D_d < 0, \\ \text{no change} & \textbf{if } \Delta D_d = 0, \end{cases} \qquad \forall d \in \{1, \ldots, U\}. \tag{2}$$

3.3 The attack when Side-Channel Information is available

Let us consider the scenario where Bob is not using a constant time implementation. We assume that this will give us information on how many iterations that were used in the decoding step. In such a scenario it is possible to more efficiently utilize CASE-0 information when mounting an attack, increasing the amplification effect of the chaining method even further.

In practice, that attack is performed in a very similar manner to the one without side-channel information. We simply modify it so that we additionally save the difference of the distance spectra for each successful decoding (CASE-0) into *different* vectors depending on the number of iterations required for the decoding. Introducing vectors $A_j = (A_{1,j}, A_{2,j}, \ldots, A_{U,j})$ and similarly vectors B_j, where j runs through the possible

Algorithm 2 Algorithm for constructing error pattern chains and gathering distance information, without side channel information.

Input: $\mathbf{e}_0, T, m, G$ ▷ $\mathbf{e}_0$: initial error pattern, T: chain length
Output: A, B, G, F

$A \leftarrow$ zero-vector of length $r/2$
$B \leftarrow$ zero-vector of length $r/2$
$F \leftarrow$ zero-vector of length $r/2$
$G \leftarrow$ zero-vector of length $r/2$
$\mathbf{e} \leftarrow \mathbf{e}_0$
$i \leftarrow 0$
while $i < T$ **do**
  $s \leftarrow$ distance spectrum of $\mathbf{e}$
  repeat
    $x \leftarrow$ position of a random 1 in $\mathbf{e}$
    $y \leftarrow$ position of a random 0 in $\mathbf{e}$
    $\mathbf{e}' \leftarrow$ copy $\mathbf{e}$
    flip bits x and y in $\mathbf{e}'$ ▷ Random 1-bit permutation of $\mathbf{e}$
    $s' \leftarrow$ distance spectrum of $\mathbf{e}'$
    $\mathbf{c} \leftarrow mG + \mathbf{e}'$ ▷ Encrypt with $\mathbf{e}'$
    $l \leftarrow$ is $\mathbf{c}$ decodable? ▷ Attempt to decrypt $\mathbf{c}$
    for all indexes d of s **do** ▷ Loop through distance spectrum (DS)
      $j \leftarrow s'[d] - s[d]$ ▷ Value of ΔD_d
      if $j > 0$ **then**
        if l **then**
          $F[d] \leftarrow F[d] + |j|$ ▷ Save distance additions for decoding successes
        else
          $A[d] \leftarrow A[d] + |j|$ ▷ Save distance additions for decoding failures
        end if
      else if $j < 0$ **then**
        if l **then**
          $G[d] \leftarrow G[d] + |j|$ ▷ Save distance deletions for decoding successes
        else
          $B[d] \leftarrow B[d] + |j|$ ▷ Save distance deletions for decoding failures
        end if
      end if
    end for
  until not l
  $e \leftarrow \mathbf{e}'$
  $i \leftarrow i + 1$
end while

values for the number of iterations, we have an update of the form

$$\begin{cases} A_{d,j} \leftarrow A_{d,j} + \Delta D_d & \textbf{if } \Delta D_d > 0, \\ B_{d,j} \leftarrow B_{d,j} - \Delta D_d & \textbf{if } \Delta D_d < 0, \\ \text{no change} & \textbf{if } \Delta D_d = 0, \end{cases} \qquad \forall d \in \{1, \ldots, U\}. \tag{3}$$

As was explained in the previous section, the number of iterations influences which distances are more probable to appear or disappear, when comparing $D(\mathbf{e}_i)$ and $D(\mathbf{e}')$. Now j is the number of iterations required for decoding error pattern $\mathbf{e}'$. For a specific implementation of the decoder, there exists a value I where the probabilities flip. By this we mean that if $j < I$ our simulations show that distances d that exist in $D(\mathbf{h}_0)$ are more probable to be represented in $A_{d,j}$. Conversely, if $j \geq I$, the same distances are *less* likely to be represented in $A_{d,j}$.

The simulations show that for $B_{d,j}$ there is no such property. All distances d, regardless of j, are less likely to disappear from $D(\mathbf{e}')$ if they exist in h_0 (although the statistical significance varies with both the implementation of the decoder and with j).

3.4 Performing an attack

Using vectors F, G, A and B, or optionally $A_j, B_j, \forall j$ in place for plain A and B, one may reconstruct the distance spectrum of h_0 with relative ease, provided the number of samples M is large enough.

An intuitive explanation of the statistical nature of our experiment can be given by using an alternative representation of the vectors: Divide each element of the vectors F and G with the number of unsuccessful decoding attempts. The same is done for A_j and B_j (for each value of j), dividing with the number of successful decoding attempts which correspond to the value of j. If this is done then the results would be the probability for each d causing the underlying event (for example, for the F vector, this is the event that a distance d is added to the new error pattern and it causes a new decoding error).

A straightforward attack would be to simply use, for example, vector F and directly perform a key recovery using the same algorithm as published by Guo et al. However, to extract as much information as possible from each decoding attempt the following formula may be used to calculate a new aggregated distance vector (considering the case where side-channel information is available);

$$F + G + \left(\sum_{j=1}^{I-1} A'_j + B_j \right) + \left(\sum_{j=I}^{J} A_j + B_j \right), \tag{4}$$

where

$$A'_j = \max(A_j) - A_j + \min(A_j). \tag{5}$$

Here J is the maximum number of iterations allowed by the decoder implementation. The reason we calculate A'_j for $j < I$ instead of using A_j directly stems from the fact that the probability relationship for each distance d to occur in A_j is flipped when comparing with $A_j, j \geq I$. This follows directly from the discussion given in the previous section. We calculate A'_j in this manner to preserve the total sum of the vector so that the weighting of each vector is proportional to the number of samples used to build each vector.

If side-channel information is *not* available, then the above formula might be simplified as

$$F + G + A' + B, \tag{6}$$

where

$$A' = \max(A) - A + \min(A). \tag{7}$$

We use A' here for the same reasons as when we are calculating with (4), except here we are assuming that the number of ciphertexts resulting in a j below I is greater than the number of ciphertexts resulting in $j \geq I$. This has always been the case in our simulations and it can be checked by the reader by comparing the ratio of each iteration in Figure 2b with the I derived from Figure 5a. As long as the assumption holds true the vector A will result in an aggregated distance spectrum with inverse probabilities compared to, for example, the F vector. The results added to A vector where $j \geq I$ will add noise, which is why using side-channel information provides for a more effective attack.

Equations (4) and (6) are simple formulas, and it is conceivable that they might be further optimized, for example by introducing some weight factor for each vector. This is not done in this paper and may be regarded as future work.

3.5 Speeding up $\mathbf{e}_0$ generation

The algorithm described thus far requires $\mathbf{e}_0$ (a first undecodable error pattern) as input. Depending on the decoding failure rate of the decoder being used this might not be trivial in practice. As it happens, one might actually utilize the side-channel attack described above to find the first undecodable error pattern.

This is done by selecting any random error pattern, attempt to decode it, store the number of iterations required for decoding and modify the pattern similarly to how we do it in chaining method described above. Then we try to decode the new pattern, if the number of iterations are lower we discard the modifications and try again, otherwise we use the modified error pattern as the new point-of-origin and base the modifications on this new pattern. We keep this up until a pattern has been found that cannot be decoded. Our simulations show this method offers significant speedups compared to the standard method of random trial and error.

4 Implementations and Numerical Results

In this section we describe how our simulations[2] were made and what results were produced. First we describe how the simulations were designed. Next, we describe the different decoder implementations used in this paper and how they affect the decoding failure rate. We follow up on this by showing the amplification effect on these different implementations, according to our simulation results. Then we show, using our simulations on the QC-MDPC scheme, that the amount of necessary ciphertexts are indeed significantly reduced compared to the original attack by Guo et al. Finally, the effect of using side-channel information is analyzed.

4.1 Simulation set-up

All results shown in this paper are based on simulations using the same key and using parameters for 80-bit security. We have confirmed our findings by rerunning our simulations using 9 other keys, although those results are not shown here, for sake of brevity.

During these simulations we make use of the chaining method to find $\mathbf{e}_0$, as described in Section 3.5. We confirm with our simulations that using side-channel information gives significant speedup. In fact, even though we are using decoder $\mathcal{Q}$ (see below) we can find $\mathbf{e}_0$ in a matter of seconds or a few minutes at most.

[2]Source code available upon request.

4.2 Decoder implementations

The decoder implementations used in this paper are based on the descriptions of $\mathcal{B}$ and $\mathcal{F}$, given in [HvMG13]. We have also implemented a decoder $\mathcal{Q}$ based on the descriptions provided in [Cho16].

4.2.1 Decoder $\mathcal{B}$

This is the standard Gallager probabilistic iterative bit-flipping decoder implemented as originally described in [Gal62] and later in [HvMG13]. Once per iteration the syndrome is computed to determine how many parity checks are violated. This decoding algorithm is using precomputed threshold values T_j, where j is the current iteration number. These are used to determine what bits violate at least T_j parity checks and those that do are flipped. Before starting the next iteration the syndrome is compared to zero. If it is zero then the algorithm stops. For this algorithm the maximum number of iterations J is set to 10, and once $j > J$ the decoder stops and returns a decoding failure. The thresholds, for $\mathcal{B}$, are (as given in [HvMG13]):

$$28, 26, 24, 22, 20, 18, 16, 14, 12, 10$$

This algorithm is not constant-time, and extracting side-channel information would be trivial. In our experiment we have modified it to return the actual number of iterations and thus removing the noise that would have been introduced by measuring the amount of time required for decoding (as would be required in a real attack scenario).

As we can see in Figure 2a this decoder has a relatively high decoding failure rate and there are other implementations that perform better in this regard, as is shown in the next two sections. This is the decoder that was used in [GJS16].

4.2.2 Decoder $\mathcal{F}$

This decoder is a variant of decoder $\mathcal{B}$ and is also implemented as described in [HvMG13]. Instead of computing the syndrome once every iteration however, it *directly* updates the syndrome as soon as a bit is flipped (this is called an *in-place* decoder). It also compares the syndrome to zero after each update and exits before the current iteration has run its course. It uses the same T_j and J as decoder $\mathcal{B}$. As can be seen in Figure 2a this strategy improves the decoding failure rate, compared to decoder $\mathcal{B}$.

4.2.3 Decoder $\mathcal{Q}$

This decoder is our implementation of *QcBits* (pronounced "quick bits"). We have implemented it according to the description provided in [Cho16]. Our implementation does not implement any of the performance enhancements nor does it attempt to be constant time. Algorithmically, *QcBits* appears to be equivalent with decoder $\mathcal{F}$ but with different threshold values T_j and $J = 6$. It should be noted however that our implementation $\mathcal{Q}$ uses $J = 10$, since keeping J constant across all implementations makes comparisons of the decoder characteristics easier. The thresholds used are, as given in [Cho16] and the published source code of *QcBits*:

$$29, 27, 25, 24, 23, 23, 23, 23, 23, 23$$

It is worth noting that the last 4 values are *not* given in [Cho16], since they stop at 6 iterations, but were instead found in the published source code. As can be seen in Figure 2a these 4 last values are indeed not optimal and should probably instead follow a decreasing formula in the same manner as for decoders $\mathcal{B}$ and $\mathcal{F}$.

The author of *QcBits* claims no decoding failures when decoding 10^8 ciphertexts. As can be seen in Figure 2a we were unable to reproduce this very low decoding failure rate. Our implementation might be flawed, but regardless, decoder $\mathcal{Q}$ shows value to us in our simulations due to its decoding failure rate still beating the other decoders. This enables us to get a good indication of the amplification effect of the chaining method, as we will see in the next section.

4.3 Amplification Effect

To test the amplification effect of the chaining method we implemented a small test routine that generates ciphertexts with error patterns according to our chaining method. We collected 2^{28} number of samples by running the algorithm in 20 threads and thereby creating 20 separate error chains. The entire test was run 10 times (with different keys) for each algorithm. As can be seen in Figure 2b the amplification effect is significant regardless of the decoding failure rate of the decoder in question. In fact, based on the results of these simulations, the amplification effect appears to increase for decoders with lower decoding failure rate. In these simulations we have only concerned ourselves with H_0. In practice, for 80-bit security, this means that we are only looking at the first half of the 9602-bit ciphertext.

4.4 Chaining Attack

Using the Algorithm 2 as presented in Section 3.2 we generated the results shown in Figure 3. It is worth noting here that plots of vectors F and G look almost identical (not shown here), and the underlying data is extremely similar (although not identical). These results show that the amount of extra information that can be extracted from using both vectors, instead of only one of them, is extremely limited.

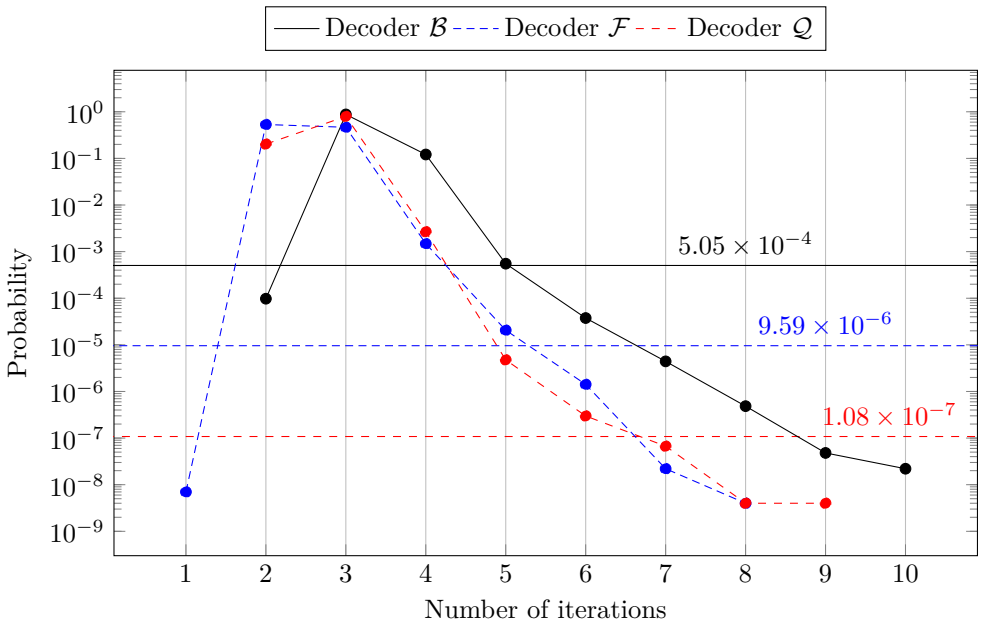
A simple explanation of why this would be the case is given if one first considers the fact that each pattern is part of a chain. As such, a distance in the pattern cannot reasonably be removed from the distance spectrum of the current pattern $\mathbf{e}_i$ *unless* it has first been added to a pattern $\mathbf{e}_l$, where $l < i$.

From vector F in Figure 3 we can easily see that the probability for each of the distances are stratifying based on the multiplicity of the distance in $\mathbf{h}_0$, just as described in Section 3.2 and in [GJS16]. It is interesting to note that the multiplicities are are not separating into layers by an equal amount, for the different decoder implementations. In Figure 3 we see as expected that decoder $\mathcal{B}$ requires the least amount of ciphertexts in order to separate into different layers. However it appears that decoder $\mathcal{F}$ unexpectedly requires *more* ciphertexts than $\mathcal{Q}$. This is an interesting discovery that we have not investigated further.

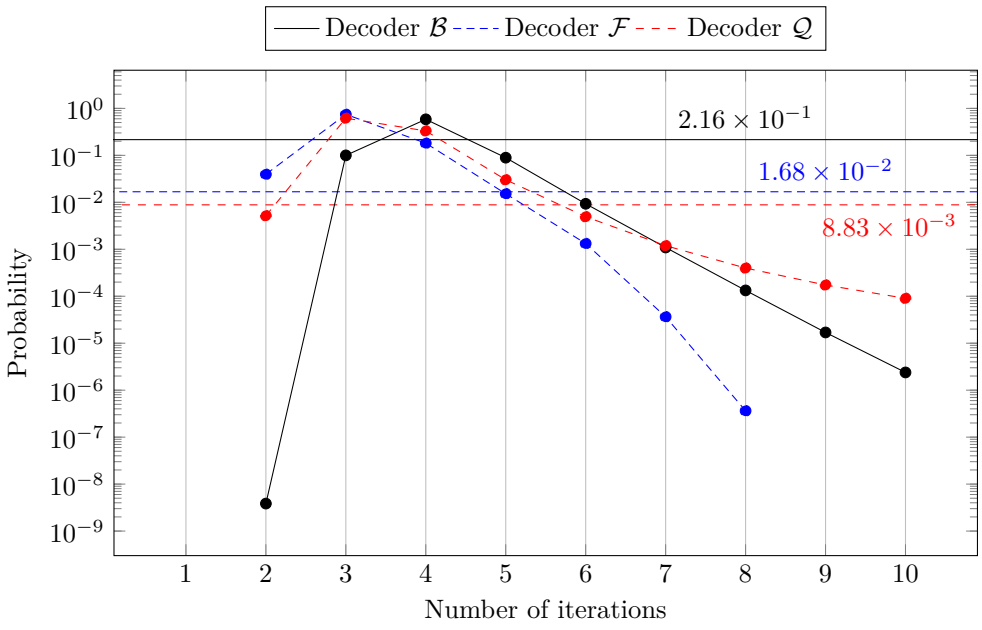
Also noteworthy in Figure 3 is the asymmetry of the results for the *in-place* decoders. It was, to our knowledge, first discovered and discussed by Eaton et al. in [ELPS18]. In order to utilize the results of these decoders in a real attack one would first be required to normalize the vectors according to a regression fitting, before categorizing each distance by its correct multiplicity.

In order to facilitate the comparison of our results with those of [GJS16], we will henceforth only consider results from simulations with decoder $\mathcal{B}$. In Figure 4 we plot the number of errors on the accumulated distance spectrum as a function of the number of ciphertexts.

To calculate the number of errors we use knowledge of the key to partition each distance according to its multiplicity in the real key and to calculate the average of all multiplicities (see Figure 3). For each distance d in the aggregated distance spectrum we find the multiplicity that is the closest and use this as our guess. Again we use knowledge of the key to detect if our guess is correct or wrong.

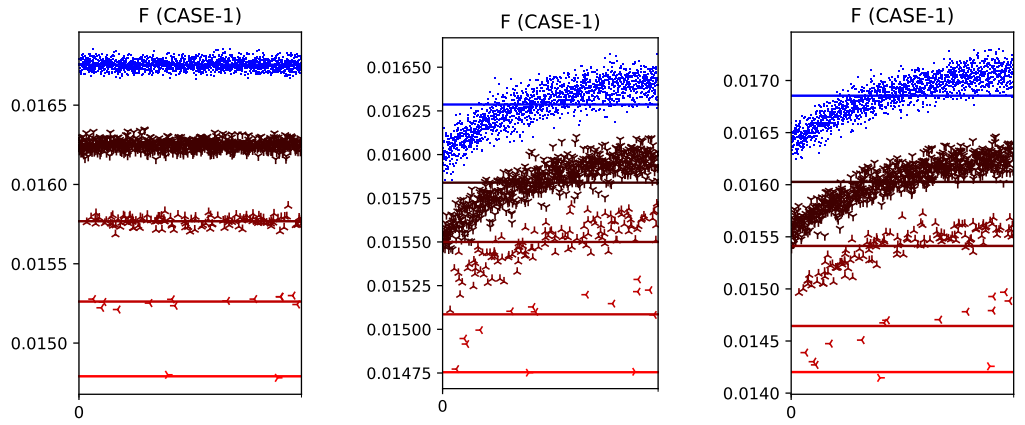


(a) Without error amplification (uniformly random error patterns). This data is based on 2^{28} number of decoding trials each, for decoders $\mathcal{B}, \mathcal{F}$ and $\mathcal{Q}$.



(b) With error amplification (chaining method). This data is based on 2^{28} number of decoding trials each, for decoders $\mathcal{B}, \mathcal{F}$ and $\mathcal{Q}$.

Figure 2: In these logarithmic plots the markers indicate by what probability (y-axis) each of the implementations can decode a message using (exactly) a certain number of iterations (x-axis). The horizontal lines indicate the decoder failure rate for each decoder implementation.



(a) Decoder $\mathcal{B}$ (b) Decoder $\mathcal{F}$ (c) Decoder $\mathcal{Q}$

Figure 3: The plot for vector F for decoders $\mathcal{B}$, $\mathcal{F}$ and $\mathcal{Q}$. Multiplicity levels for each distance $d \in D(h_0)$ are easily distinguished. This data is based on 2^{28} decoding attempts, for each decoder. Blue dots indicate distances that do *not* appear in $\mathbf{h}_0$ (multiplicity 0). Dark markers indicate those distances that appear in $\mathbf{h}_0$ once (multiplicity 1). Higher multiplicities are marked analogously with less dark shades of red and with different markers. The horizontal lines represent the average probability, for each multiplicity. In [ELPS18] Eaton et al. discovered and discussed the asymmetric nature of *in-place* decoders. As we can see this asymmetric shape is also found in Figure 3b and Figure 3c.

In Figure 4 we see that the required number of ciphertexts is in the order of 12M as compared to the 240M in [GJS16] for CPA security, a speedup of a factor of 20. Since it was noted in Section 4.3 that the amplification effect appears to be increasing with the use of "better" decoder implementations we expect the speedup factor to increase in proportion. However, [GJS16] does not provide any such numbers for us to compare with.

In [GJSW] it was shown that it is in fact not necessary to have a 100% correct distance spectrum in order to mount an attack. Guo et al. showed that 900 errors in the distance spectrum can be reliably tolerated (for 80-bit security) and still being able to successfully recover H in a reasonable time. For comparison, Guo et al. required 4M ciphertexts to recover a distance spectrum with 900 errors. Our simulations show this requirement to have dropped to approximately 310 000 ciphertexts, for decoder $\mathcal{B}$, using our amplification attack with the chaining method.

4.5 Chaining Attack, With Side-Channel Information

As was explained in Section 3.3, for vector A the probabilities for each distance d flips depending on if the number of iterations required to decode the ciphertext is below a implementation dependent threshold value I, or not. By close examination of Figure 5a one can see that $I_{\mathcal{B}}$ is 5. Although not shown here, we have similar results for the other decoders: $I_{\mathcal{F}} = 4$ and $I_{\mathcal{Q}} = 4$. Figure 5b confirms that there is no such analogous I for vector B, i.e. distances that are removed from the distance spectrum of $\mathbf{e}_i$.

Figure 4 show that the attack can indeed be further improved by utilizing side-channel information. A simple comparison with [GJS16] gives us a factor of $240/8 = 30$, (compared to 20 without side-channel information). We acknowledge that these calculations are based on ideal data which does not depend on timing measurements and as such provides no measurement errors nor any noise introduced by, for example, CPU scheduling and cache timings.

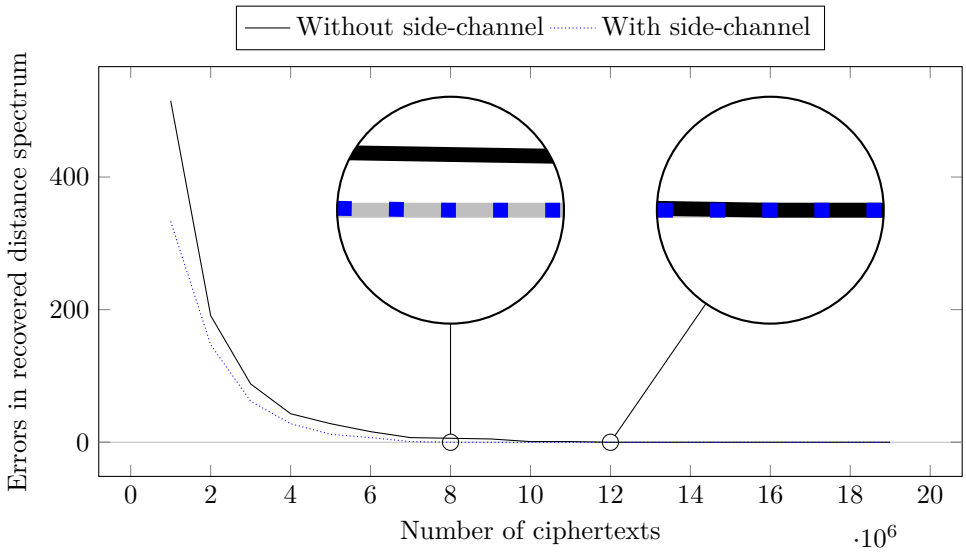
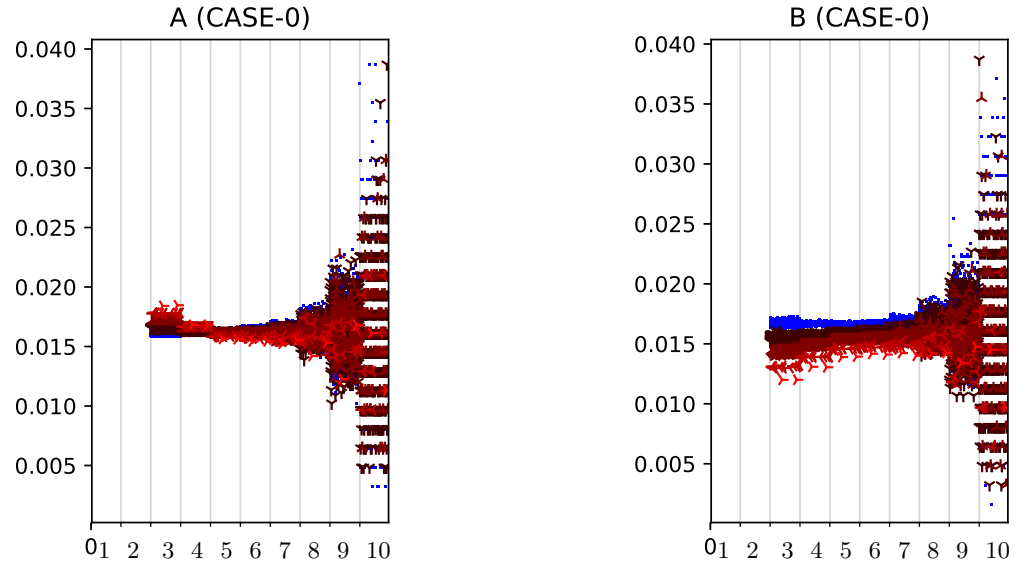


Figure 4: Number of errors in recovered distance spectrum for decoder $\mathcal{B}$. The number of errors is calculated according to the description given in Section 4.4. The results plotted are both with and without side-channel information. The circled values are the lowest number of ciphertexts needed where the number of errors reach zero (and stays there), with and without side-channel information respectively. For easier viewing $y = 0$ is shown as a grid line.



(a) Plot of vector A_j where $0 < j \leq 10$.

(b) Plot of vector B_j where $0 < j \leq 10$.

Figure 5: Decoder $\mathcal{B}$: The plot for vectors A and B. The vectors have been converted to show probabilities instead of raw values, for easier understanding of y-values. This data is based on 2^{28} ciphertexts. The plots show the probability for decoding success using j (x-axis) number of iterations for each distance in the distance spectrum. j in this plot is discrete and ranges from 1 to 10 and is indicated by the grid lines. The wider range of values for certain iterations (e.g. $j = 10$) are a by-product of the smaller sample size. In the Figure 5a we can see that $I = 5$ since the probabilities flip for the results where $j \geq 5$.

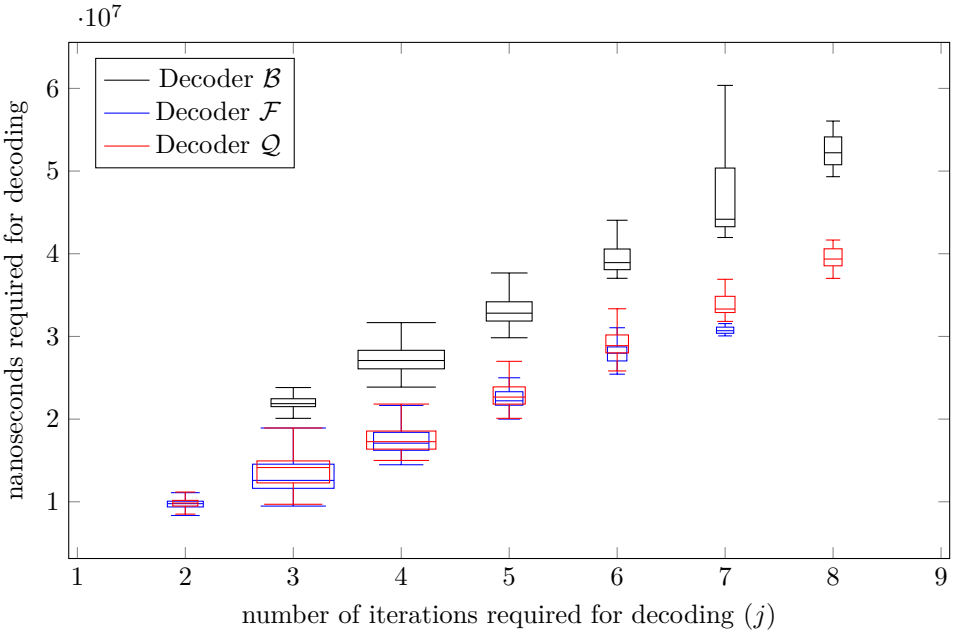


Figure 6: Box plots showing the measurement distribution of the number of nanoseconds (y-axis) for decoder $\mathcal{B}$, $\mathcal{F}$ and $\mathcal{Q}$ to decode a single message using j number of iterations (x-axis). The middle line shows the median and the upper and lower edges of the boxes show the upper and lower quartiles, respectively. The upper and lower whiskers in turn indicate the lowest and highest measurement still within 1.5 IQR (InterQuartile Range) of the upper and lower quartiles, respectively. The width indicate the relative number of collected measurements (i.e. the relative reliability) of each box plot. It should be noted that the underlying data has been cleaned from very obvious outliers before generating the plots. Roughly 2^{16} measurements, for each decoder, was used to generate the data which is summarized in this figure.

4.5.1 Non-constant time decoders

Figure 6 presents the results of a simple experiment that measures the decoding time of 2^{16} ciphertext decodings, for each decoder. We argue that these results would indicate the practicality of using side channel information on non-constant time decoder implementations, in a close to real world setting. By observing the distributions of the measurements one can quickly see that the number of iterations are indeed easily distinguishable. It should be noted however that these measurements are done in-process and do therefore not introduce any additional noise, from for example, network latency.

5 Conclusions

In the general case, we have shown that the advertised decoding failure rate of a decoder implementation might not always tell the whole truth about the security of the particular implementation. We have shown that knowledge of a single error pattern might be used as leverage for an attacker to generate other difficult-to-decode error patterns. We get this amplification effect using the chaining method described in this paper.

Specifically for QC-MDPC we have shown that using the technique described in this paper one can mount a successful attack against the scheme using more than an order of magnitude fewer ciphertexts than previous state-of-the-art attacks, for CPA security.

This is valid without using side-channels or private information such as syndrome weight. This is the result of both the error rate amplification and the fact that we can extract information also from decoding *successes* as well as failures.

Additionally we have shown that side-channel information can be used to improve the efficiency of the attack, if such information is available. It can also be used to speed up the discovery of the initial error pattern which we use to bootstrap the attack.

6 Further Work

In this paper we have only presented results related to parameters corresponding to 80 bits of security. It would be interesting to see how the results scale to 128- and 256-bit security parameters.

Another point of research would be to continue the simulations using other decoders based on different techniques. In [CS16], Chaulet et al. present a decoder which uses dynamically calculated thresholds, and which is able to achieve failure rates comparable to our decoder $\mathcal{Q}$.

Although Eaton et al. do give a tentative explanation in [ELPS18], the asymmetric nature of *in-place* decoders have not been fully explained to our satisfaction and we would like to have a more complete understanding of the effects of this asymmetry.

References

[BBC08] Marco Baldi, Marco Bodrato, and Franco Chiaraluce. A New Analysis of the McEliece Cryptosystem Based on QC-LDPC Codes. In *Security and Cryptography for Networks, 6th International Conference, SCN 2008, Amalfi, Italy, September 10-12, 2008. Proceedings*, pages 246–262, 2008.

[BCGM07] Marco Baldi, Franco Chiaraluce, Roberto Garello, and Francesco Mininni. Quasi-Cyclic Low-Density Parity-Check Codes in the McEliece Cryptosystem. In *Proceedings of IEEE International Conference on Communications, ICC 2007, Glasgow, Scotland, 24-28 June 2007*, pages 951–956, 2007.

[Ber09] Daniel J. Bernstein. Introduction to post-quantum cryptography. In *Post-quantum cryptography*, pages 1–14. Springer, 2009.

[Cho16] Tung Chou. QcBits: Constant-Time Small-Key Code-Based Cryptography. *Cryptographic Hardware and Embedded Systems – CHES 2016*, January 2016.

[CJL+16] Lily Chen, Stephen Jordan, Yi-Kai Liu, Dustin Moody, Rene Peralta, Ray Perlner, and Daniel Smith-Tone. Report on post-quantum cryptography. *National Institute of Standards and Technology Internal Report*, 8105, 2016.

[CS16] Julia Chaulet and Nicolas Sendrier. Worst case QC-MDPC decoder for McEliece cryptosystem. *CoRR*, abs/1608.06080, 2016.

[ELPS18] Edward Eaton, Matthieu Lequesne, Alex Parent, and Nicolas Sendrier. QC-MDPC: A Timing Attack and a CCA2 KEM. In *International Conference on Post-Quantum Cryptography*, pages 47–76. Springer, 2018.

[FHS+17] Tomáš Fabšič, Viliam Hromada, Paul Stankovski, Pavol Zajac, Qian Guo, and Thomas Johansson. A Reaction Attack on the QC-LDPC McEliece Cryptosystem. In *International Workshop on Post-Quantum Cryptography*, pages 51–68. Springer, 2017.

[FHZ] Tomáš Fabšic, Viliam Hromada, and Pavol Zajac. A Reaction Attack on LEDApkc. Cryptology ePrint Archive, Report 2018/140 (2018). http://eprint.iacr.org/.

[Gal62] Robert Gallager. Low-density parity-check codes. *IRE Transactions on information theory*, 8(1):21–28, 1962.

[GJS16] Qian Guo, Thomas Johansson, and Paul Stankovski. A Key Recovery Attack on MDPC with CCA Security Using Decoding Errors. In *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Proceedings*, volume 10031 LNCS, pages 789–815. Springer Verlag, 2016.

[GJSW] Qian Guo, Thomas Johansson, and Paul Stankovski Wagner. A Key Recovery Reaction Attack on QC-MDPC. Accepted for publication in IEEE Transactions on Information Theory, final manuscript in preparation. Full version of [GJS16].

[Gop70] Valerii Denisovich Goppa. A new class of linear correcting codes. *Problemy Peredachi Informatsii*, 6(3):24–30, 1970.

[HvMG13] Stefan Heyse, Ingo von Maurich, and Tim Güneysu. *Smaller Keys for Code-Based Cryptography: QC-MDPC McEliece Implementations on Embedded Devices*, pages 273–292. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[LJ12] Carl Löndahl and Thomas Johansson. A New Version of McEliece PKC Based on Convolutional Codes. In *Information and Communications Security - 14th International Conference, ICICS 2012, Hong Kong, China, October 29-31, 2012. Proceedings*, pages 461–470, 2012.

[McE78] Robert J. McEliece. A Public-Key Cryptosystem Based On Algebraic Coding Theory. *DSN Progress Report 42–44*, pages 114–116, 1978.

[MOG15] Ingo von Maurich, Tobias Oder, and Tim Güneysu. Implementing QC-MDPC McEliece Encryption. *ACM Transactions on Embedded Computing Systems (TECS)*, 14(3):44, 2015.

[MS77] Florence J. MacWilliams and Neil J. Alexander Sloane. *The Theory of Error Correcting Codes*, volume 16. Elsevier, 1977.

[MTSB13] Rafael Misoczki, Jean-Pierre Tillich, Nicolas Sendrier, and Paulo SLM Barreto. MDPC-McEliece: New McEliece Variants from Moderate Density Parity-Check Codes. In *Information Theory Proceedings (ISIT), 2013 IEEE International Symposium on*, pages 2069–2073. IEEE, 2013.

[S+03] Nigel P. Smart et al. *Cryptography: an introduction*, volume 3. McGraw-Hill New York, 2003.

[vMHG16] Ingo von Maurich, Lukas Heberle, and Tim Güneysu. IND-CCA Secure Hybrid Encryption from QC-MDPC Niederreiter. In *Post-Quantum Cryptography*, pages 1–17. Springer, 2016.