Non-structural attacks on code-based cryptosystems

Authors: Anghel Florin-Mihai,Asandoaiei David, Iftinca Corina Elena, Tabacaru Robert

January 9, 2022

Abstract

Code-based cryptography looks to be one of the most promising prospects for the future of asymmetric cryptography. With an efficiency that is considerably better than RSA, and the advantage of offering security in the post-quantum scenario, their biggest problem is key size. In this paper, we are going to explore attacks against the Niederreiter cryptosytem, one of the main candidates for NIST standardisation, more specifically, Information Set Decoding techniques.

1 Introduction

In the coming years, large-scale quantum computers are looking more and more likely to become a reality, and this would bring about more than just a big technological advancement.
The work of Peter W. Shor[1] shows that, using a quantum computer, the factorization and discrete logarithm problems can be solved in polynomial time. This is a worrying discovery as most modern cryptographic systems rely on the intractability of these 2 problems. (RSA, DSA, ECC, Diffie-Hellman, etc.)
This led to NIST's Post-Quantum Cryptography Standardization Call, and the main areas of research for post-quantum cryptography are:

- Lattice-based cryptography
- Hash-based cryptography
- Code-based cryptography
- Multivariate cryptography
- Isogeny-based cryptography

The birth of coding theory was inspired by work of Golay, Hamming and Shannon in late 1940's[2]. Coding theory is a field of study concerned with the transmission of data across noisy channels and the recovery of corrupted messages. Equivalently, coding theory deals with attaining reliable and efficient information transmission over a noisy channel. The core of this subject lies in finding new error-correcting codes with improved parameters, along with developing their encoding and decoding algorithms. The algebraic codes which possess interesting attributes are highly demanded in specific areas. Codes which have some sort of randomness in their structure are admired by cryptographers. The goal of coding theory is then to encode information in such a way that even if the channel (or storage medium) acquaint errors, the receiver can correct the errors and retrieve the original transmitted information. The term error-correcting code is used for both the detection and the correction mechanisms. In other words, to be accurate, we have error-detecting code and

error-correcting code. The earlier one allows the detection of errors, whereas, later provides correction of errors discovered.
Code-based cryptography is a very promising candidate for post-quantum security as the concept has been around for quite some time. (The McEliece and Niederreiter Cryptosystems) The main problem of the base versions of these systems is the key size, so research is guided towards finding variants with smaller key sizes, while maintaining the level of security. Out of the many researchers that are tackling this topic, the works of Marco Baldi and Paolo Santini are of significant importance, outlining the state of the art, open challenges, and existing attacks.[3]
In this paper we will focus on Niederreiter cryptosystem and on Lee-Brickel and Prange attacks. The novelty of our work is given by the practical side of our research: the chosen parameters for the experiments, testing on different sizes and generalizing to GF(3). Works such as [4] and [5] have provided a great basis to build our implementation on.

2 Code based cryptography and Niederreiter

2.1 Introduction in code theory

This section introduces the notation and background on error correction codes, and the state for the problems which are mentioned and for which best solvers are ISD algorithms.
In the following, we consider the case of binary linear block codes, denoting as $C(n, d, k)$ a code of length n, dimension k and minimum distance d. This code is a subspace of $\{0, 1\}^n$ containing 2^k distinct vectors and it can be represented by a $k \times n$ binary matrix G known as the code generator matrix. In general, it is indicated that the amount of redundant bits in an element of the code is $r = n - k$. The minimum distance of a linear block code corresponds to the minimum weight of its codewords (apart from the null one which has null weights). An alternative representation is provided by the parity check matrix H, which is obtained through the algebraic constraint $HG^T = 0_{r\times k}$. The parity check matrix is a $r \times n$ binary matrix for which the product of a codeword c by H is a null r-bit long vector. The quantity obtained multiplying a generic n-bit vector $\tilde{c}$ by H is called the syndrome $s = H\tilde{c}^T$ of $\tilde{c}$ through H. Note that, if the binary vector $\tilde{c}$ is not a codeword, the syndrome of $\tilde{c}$ through H is not null, which means we can write $\tilde{c} = c + e$, with c being a codeword and $s = He^T$.
Decoding of $\tilde{c}$ through $C(n, k, d)$ consists in finding the codeword c whose distance from $\tilde{c}$ is minimum. The procedure of finding a length-n binary vector e, with Hamming weight smaller than or equal to some integer t, given a non-null syndrome s and a parity matrix H is known as syndrome decoding.
We can now recall the two decisional problems in coding theory which were proven to be NP-Complete by Berlekamp in [6], and from which the main building blocks of the cryptographic trapdoors of code-based cryptosystems are derived.

Coset Weight problem.

Given a random $r \times n$ binary matrix H, an r-bit vector s and a positive integer t, determine if an n-bit vector e with $wt(e) \leq t$, where $wt(\cdot)$ is the Hamming weight function, such that $He^T = s$ exists.
The Coset weight problem is also known as the Decisional Syndrome Decoding Problem (DSDP).

Subspace weight problem.

Given a random $r \times n$ binary matrix H, and a positive integer w, determine if an n-bit vector c with $wt(c) = t$, where $wt(\cdot)$ is the Hamming weight function, such that $He^T = 0_{r\times 1}$ exists.

The Subspace weights problem is also known as the Decisional Codeword Finding Problem (DCFP).

The typical hard problems which are employed to build code-based cryptographic trapdoors are the search variants of the aforementioned two problems. Specifically, the Syndrome Decoding Problem (SDP) asks to find an error vector of weight $\leq t$, while the Codeword Finding Problem (CFP) asks to find a codeword with a given weight w. A well known result in computational complexity theory, also mentioned in [7], states that any decision problem belonging to the NP-Complete class has a search-to-decision reduction. In other words, it is possible to solve an instance of the search problem with a polynomial amount of calls to an oracle for the corresponding decision problem.

2.2 Niederreiter cryptosystem

The class of NP-Complete problems is of particular interest to design cryptosystems, as it is widely believed that problems contained in such a class cannot be solved in polynomial time by a quantum computer. Indeed, the best known approaches to solve both the Codeword Finding Problem (CFP) and the Syndrome Decoding Problem (SDP) have a computational complexity which is exponential in the weight of the codeword or error vector to be found. A notable example of code-based cryptosystem relying on the hardness of the Syndrome Decoding Problem (SDP) is the one proposed by *Niederreiter* in [8].

The Niederreiter cryptosystem was proposed by Harald Niederreiter in 1986. It is a deterministic cryptosystem, with a high speed of encryption, and it can also be used to create a digital signature. At first, the cryptosystem used generalized Reed–Solomon (GRS) codes but due to the fact that it was broken by Sidel'nikovand Shestakov, it was subsequently adapted to use Goppa codes (Baldi, Barenghi, et al.).

In cryptography, the Niederreiter cryptosystem is a variation of the McEliece cryptosystem developed in 1986 by Harald Niederreiter. It applies the same idea to the parity check matrix, H, of a linear code. Niederreiter is equivalent to McEliece from a security point of view. It uses a syndrome as ciphertext and the message is an error pattern. The encryption of Niederreiter is about ten times faster than the encryption of McEliece. Niederreiter can be used to construct a digital signature scheme.

The Niederreiter cryptosystem works as follows:

Key Generation

1. A binary linear (n, k)-code C is randomly chosen, together with a parity check matrix H of size $(n - k) \times n$. The code should have an efficient decoding algorithm and should be capable of correcting at least t errors.

2. Compute the $(n - k) \times n$ parity check matrix H of the code.

3. Choose a random, dense $(n-k) \times (n-k)$ non-singular matrix S and a random $n \times n$ permutation matrix P and compute $H_{PUB} = SHP$.
4. Publish public key (H_{PUB}, t) and keep secret the private key (S, H, P).

Message Encryption

1. The message m you want to be send, is encoded as a binary string $e^{m'}$ of length n and weight at most t.
2. The ciphertext is computed as $c = H_{PUB}e^T$.

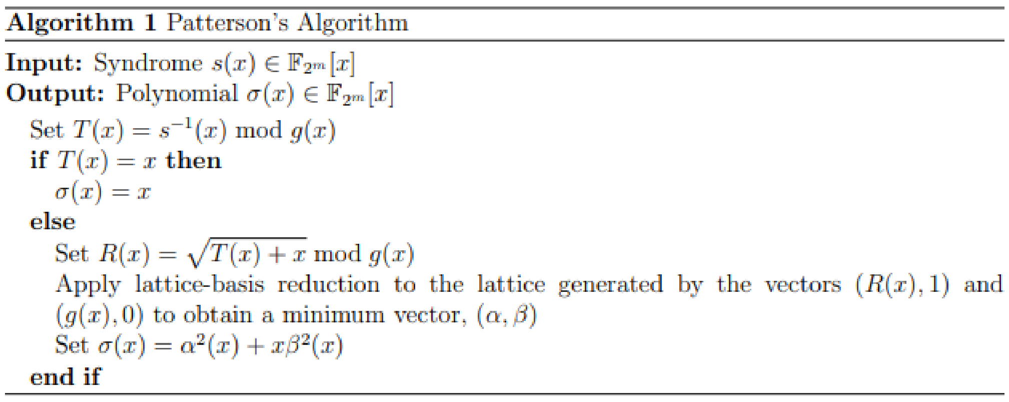
Message Decryption

1. Computation of $S^{-1}c = HPm^T$.
2. Applying of syndrome decoding algorithm for G to recover Pm^T.
3. Computation of the original message m, via $m^T = P^{-1}Pm^T$.

The Niederreiter scheme, based on the binary Goppa codes, is among the most promising candidates for inclusion into the post-quantum standards of asymmetric cryptosystems. Its main advantage besides the quantum resistance is high performance and cryptographic strength. The biggest drawback of this cipher is large key sizes which is the reason for the less prevalence of the given cryptosystem in comparison with RSA and ElGamal schemes.

2.3 Syndrome Decoding

The Niederreiter cryptosystem is constructed using the NP-hard problem of finding a codeword with minimal Hamming distance to a given word. Since the recipient of an encrypted message has knowledge of the Goppa polynomial used to construct the code, an efficient decoding method is possible. With this knowledge, the sender can efficiently locate the codeword that is within a Hamming distance of t from the encrypted message. One method of performing this efficient decoding is Patterson's Algorithm.



Algorithm 1 Patterson's Algorithm

```
Input: Syndrome $s(x) \in \mathbb{F}_{2^m}[x]$
Output: Polynomial $\sigma(x) \in \mathbb{F}_{2^m}[x]$
  Set $T(x) = s^{-1}(x) \bmod g(x)$
  if $T(x) = x$ then
    $\sigma(x) = x$
  else
    Set $R(x) = \sqrt{T(x) + x} \bmod g(x)$
    Apply lattice-basis reduction to the lattice generated by the vectors $(R(x), 1)$ and
    $(g(x), 0)$ to obtain a minimum vector, $(\alpha, \beta)$
    Set $\sigma(x) = \alpha^2(x) + x\beta^2(x)$
  end if
```

3 Attacking code based cryptosystems

For a code based cryptosystem, an attacker has two different options to retrieve an original plain text m when having the encrypted message y (according to [3]).

- Retrieve the secret code G having the public code $\hat{G}$.
- Decode y having no knowledge of a efficient decoding algorithm for $\hat{G}$.

The first type of attacks are called structural attacks. If a decodable representation of a code can be obtained in subexponential time, then this code should not be used in a Niederreiter cryptosystem. The best codes are the codes for which the best attacks are decoding random codes, the second type of attacks, called non-structural attacks. The best attack techniques should not be based on the structure of the code, thus being applicable to all the variants, also known as Information Set Decoding (ISD). ISD tries to find enough error-free locations in a codeword to make decoding possible regardless of the errors which affect the codeword itself.

3.1 Prange

The first known variant of ISD is Prange's algorithm (as presented in [3]), which is based on the idea of guessing a set I of k error-free positions in the error vector e to be found in the Syndrome Decoding Problem (SDP). The columns of H are permuted so that those indexed by I are packed to the left (operation equivalent to the multiplication of H by a properly sized permutation matrix P. Thus, we obtain the column-reordered matrix $\hat{H} = HP$, which can be put in Reduced Row Echelon Form (RREF), with the identity matrix I_r placed to the right ($[VI_r] = U\hat{H}$). If it is impossible to turn $\hat{H}$ in Reduced Row Echelon Form (RREF) because the rightmost submatrix $r \times r$ is not full-rank, a different permutation is picked. To bring H in Reduced Row Echelon Form (RREF), the same transformation U is then applied to the single-bit rows of the column syndrome vector s, obtaining $\bar{s} = Us$. If the weight of the permuted error vector $\hat{e}$ obtained as $\hat{e} = eP = [0_{1\times k}\bar{s}^T]$, where $0_{1\times k}$ is the all-zero error vector of length k, matches the expected error weight t, then the algorithm succeeds and the non-permuted error vector $\hat{e}P^T$ is returned. A pseudo-code for Prange's ISD algorithm is as follows:

Algorithm 1 Syndrome decoding formulation of Prange's ISD

```
Input: $s$: an $r$-bit long syndrome (column vector)
       $H$: an $r \times n$ binary parity-check matrix
       $t$: the weight of the error vector to be recovered
Output: $e$: an $n$-bit binary row error vector $He^T = s$, with WEIGHT$(e) = t$
Data: $P$: an $n \times n$ permutation matrix
      $\bar{s}$: an $r$-bit long binary column vector
      $V$: an $r \times k$ binary matrix $V = [v_0, \dots, v_{k-1}]$

1: repeat
2:     repeat
3:         $P \leftarrow$ RANDOMPERMUTATIONGEN$(n)$
4:         $\hat{H} \leftarrow HP$
5:         $\langle U, [V|W] \rangle \leftarrow$ REDROWECHELONFORM$(\hat{H})$
6:     until $W \neq I_r$
7:     $\bar{s} \leftarrow Us$
8:     $\hat{e} \leftarrow [0_{1\times k} \bar{s}^T]$
9: until WEIGHT$(\hat{e}) = t$
10: return $\hat{e}P^T$
```

3.2 Lee-Brickell

Another ISD algorithm (presented in [3]), introduced by Lee and Brickell, begins with the same operations as Prange: the computation of the Reduced Row Echelon Form (RREF) of $\hat{H}$ and the derivation of the corresponding syndrome $\bar{s}$. The improvement in the Lee Brickell compared to Prange comes from allowing p positions in the k selected in the error vector to be error-affected. The p remaining error positions are guessed. To verify that the guess is valid, the algorithm exploits the identity $[VI_r]\hat{e}^T = \bar{s}$, where $\hat{e}$ is split in two parts, $\hat{e} = [\hat{e}_1\hat{e}_2]$, with $\hat{e}_1$ being k bit long and with weight p, and $\hat{e}_2$ being r bits long and with weight $t - p$. The identity is rewritten as $V\hat{e}_1^T + I_r\hat{e}_2^T = V\hat{e}_1^T + \hat{e}_2^T = \bar{s}$, from which follows that $V\hat{e}_1^T + \bar{s} = \hat{e}_2^T$ must have weight $t - p$. Indeed, this condition is employed by the algorithm to check if the guess of p positions is correct. The algorithm is as follows:

Algorithm 2 Syndrome decoding formulation of Lee and Brickell's ISD

```
Input: $s$: an $r$-bit long syndrome (column vector)
       $H$: an $r \times n$ binary parity-check matrix
       $t$: the weight of the error vector to be recovered
Output: $e$: an $n$-bit binary row error vector $He^T = s$, with WEIGHT$(e) = t$
Data: $P$: an $n \times n$ permutation matrix
      $\hat{e} = eP$: the error vector permuted by $P$
      $p$: the weight of the first $k$ bits of $\hat{e}, 0 \leq p \leq t, p = 2$, proven optimal
      $\bar{s}$: an $r$-bit long binary column vector, equal to the syndrome of $e$ through $[VI_r]$
      $V$: an $r \times k$ binary matrix $V = [v_0, \dots, v_{k-1}]$

repeat
    $\langle P, [VI_r, \bar{s}\rangle \leftarrow$ ISEXTRACT$(H, s)$
    for $j \leftarrow 1$ to $\binom{k}{p}$ do
        $I \leftarrow$ INTEGERTOCOMBINATION$(j)$
        if WEIGHT$(\hat{s} + \sum_{i \in I} v_i) = t - p$ then
            $\hat{e} \leftarrow [0_{1 \times k}(\bar{s} + \sum_{i \in I} v_i)^T]$
            foreach $i \in I$ do
                $\hat{e} \leftarrow \hat{e} + [0_{1 \times i}10_{1 \times (r+k-1-i)}]$
            end for
            break
        end if
    end for
until WEIGHT$(\hat{e}) = t$
return $\hat{e}P^T$
```

3.3 Generalizing Lee-Brickell

While the above presented algorithms were originally designed for binary codes, we will present a generalization of the Lee-Brickell algorithm to be stated for finite fields F_q, as presented in [9]. (one can also see [10] for more information regarding information set decoding over F_q)

We will consider the following: Let C be an $[n, k]$ code over a finite field F_q. Let G be a generator matrix for C. Let I be a non-empty subset of $1, \dots, n$. Denote by G_I the restriction of G to the columns indexed by I. For any vector y in F_q^n denote by y_I the restriction of y to the coordinates indexed by I.

Let y be a vector in F_q^n at a distance w from the code C. The goal of this section is to determine a vector $e \in y + F_q^k G$ of weight w given G, y and w. Note that $y + F_q^k G$ is the coset of C containing e.

Lee-Brickell's generalized algorithm. Let p be an integer with $0 \leq p \leq w$.

1. Choose an information set I.

2. Replace y by $y - y_I G_I^{-1} G$.

3. For each size-p subset $A = \{a_1, \dots, a_p\} \subset I$ and for each $m = (m_1, \dots, m_p)$ in $(F_q^*)^p$ compute $e = y - \sum_{i=1}^{p} m_i g_{a_i}$. If e has weight w print e. Else go back to Step 1.

Step 1 can be performed by choosing k indices in $\{1, \ldots, n\}$ uniformly at random and then performing Gaussian elimination on G in order to see if its I-indexed columns form an invertible submatrix G_I. A better way of determining an information set I is to choose k columns one by one: check for each newly selected column if it does not linearly depend on the already selected columns.

If $p = 0$ Step 3 only consists of checking whether $y - y_I G_I^{-1} G$ has weight w. If $p > 0$ Step 3 requires going through all possible weighted sums of p rows of G which need to be subtracted from $y - y_I G_I^{-1} G$ in order to make up for the p errors permitted in I.

Steps 1-3 form one iteration of the generalized Lee-Brickell algorithm. If the set I chosen in Step 1 does not lead to a weight-w word in Step 3 another iteration has to be performed.

The parameter p is chosen to be a small number to keep the number of size-p subsets small in Step 3. In the binary case $p = 2$ is optimal.

4 Testing Environment and Results

In order to implement and test the mentioned algorithms we chose to use SageMath. It is an open source mathematical system built on top of python and many popular open source packages. We chose to use this system because it provides some very useful methods for performing mathematical operations such as Gaussian elimination, random polynomial generation (inside a finite ring), and many others.
As we mentioned before, for the purpose of this paper, we chose to explore the efficiency of non-structural attacks, that do not attempt to find or exploit the underlying code used as private key for the Niederreiter cryptosystem.
Since binary Goppa codes are one of the most popular choices when implementing this type of cryptosystems, our tests are based on public keys that are derived from the parity check matrices of such codes.
To generate the codes we used random irreducible polynomials of degree t with coefficients in $GF(2^m)$, where 2^m is the length of the code we want to generate and t represents the number of errors the code can correct.
We ran our test scenarios for 1000 iterations in order to obtain accurate statistics regarding the efficiency of the attack algorithms. For Prange's ISD, we gathered statistics both from inner loop and outer loop execution, while for our tests on the Lee-Brickell ISD, we chose to perform the runs on the generalized version, and collected only "outer" loop statistics.
Notation: For simplicity, each experiment that we present will have a name with the shape: <field>-<isd_algorithm><m>-<t>. For example, GF2-Prange-6-9 denotes the results of running Prange's ISD, on a binary Goppa code with $m = 6$ and $t = 9$.

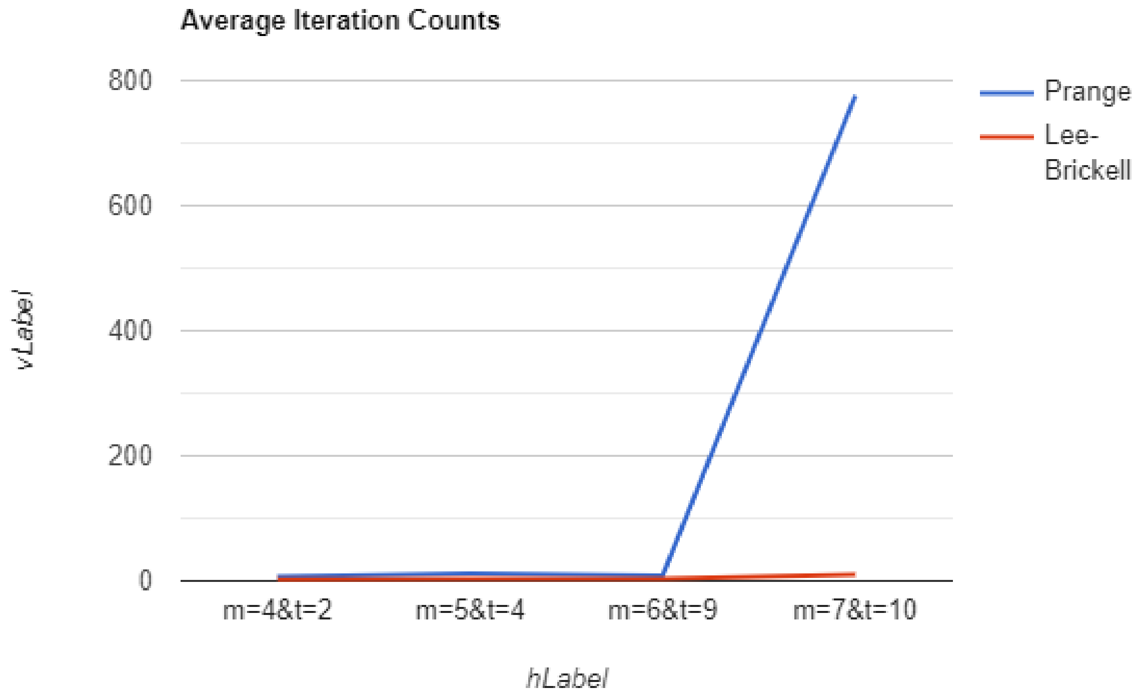
Now that we have established all of the required preliminaries, we can move on to the actual results.

Our results on $GF(2)$ point out the consistency that Lee-Brickell brings when compared to Prange. By allowing $p = 2$ positions to be affected by errors, it achieves much better results, with minimal compromise in regards to complexity. On average, an iteration of the Lee-Brickell algorithm when $m = 7$ and $t = 10$ runs in 0.16 seconds, so the avearage time it takes to decrypt a message is about 1.6 seconds, while in the case of Prange, the decryption time can vary from 1 second to a couple of minutes.
It is also worth mentioning that in the case of Prange, some runs on smaller sizes ($m = 4, t = 2$; $m = 5, t = 4$) take more iterations than on larger matrix sizes, and sometimes even fail to find a

Experiment	Outer loop mean	Outer loop St. Dev	Inner loop mean
GF2-Prange-4-2	6.5	6.6	1.3
GF2-Prange-5-4	12.02	11.2	1.1
GF2-Prange-6-9	8.4	8.4	1.1
GF2-Prange-7-10	776.4	742.1	1.1
GF2-LeeBrickell-4-2	2.3	1.7	-
GF2-LeeBrickell-5-4	3.3	2.6	-
GF2-LeeBrickell-6-9	3.7	3.3	-
GF2-LeeBrickell-7-10	10.3	9.5	-
GF2-Prange-6-9-lin	10.5	10.01	1.1

Table 1: Experimental results over $GF(2)$

result at all.
The last entry in the table is from an experiment that used a random linear code as a private key, and it is interesting to see that such a move can lead to worse results for our attack algorithms. However, something like this wouldn't be seen in a real environment, as random linear codes do not achieve the same error-correcting capabilities of Goppa Codes.



We also ran tests on non-binary codes ($GF(3)$), but this time using only Lee-Brickell. We ran the tests for $p = 2$ and $p = 3$ and these are the results:

Experiment	p	Iteration count mean	Iteration count St. Dev	Failure rate	Average iter. time
GF3-LeeBrickell-2-2	2	1.7	1.3	0.28	-
GF3-LeeBrickell-2-2	3	1.5	1.02	0.29	-
GF3-LeeBrickell-3-5	2	2.7	2.3	0.04	0.004s
GF3-LeeBrickell-3-5	3	1.9	1.3	0.06	0.014s
GF3-LeeBrickell-4-7	2	36.6	34.4	0.007	0.47s
GF3-LeeBrickell-4-7	3	8.8	7.3	0.01	15.01s

Table 2: Experimental results over $GF(3)$

We can clearly see that choosing p to be 3 helps decrease the average number of iterations required to decode an input, however it can hurt execution times, especially when working with larger keys.
For $GF(3)$, and for the key sizes we experimented on, exploring values for p that are higher than 3 would not make sense as it would slow down the algorithm too much.
It is interesting to observe that the rate of failure decreases with growth in size, which is similar to what we observed on Prange's algorithm when we experimented on binary codes.

Using codes based on larger fields: So far, our attempts to apply these algorithms on larger fields have been unsuccessful. We will continue to fine-tune our implementations and experiment with different algorithms that can overcome this problem.

5 Conclusions and Future Work

It is clear to see that the field of post-quantum cryptography is very relevant in our times, and code-based cryptosystems are one of the top candidates for asymmetric encryption.
This is why it is important to study the security of these cryptosytems, and continue to search for ways to reduce their key size without compromising their level of security.

Non-structural attacks on these cryptosystems are a very important topic of research, because their efficiency is greatest when working with keys of smaller size, and many researchers are trying to find new ways to make the keys for these systems as small as possible.

Our tests show that a simple algorithm like Prange starts to struggle pretty early when the size of these keys start to grow, but one like Lee-Brickell that can be calibrated according to the input can give good results for significantly larger keys. This is especially true when working with non-binary codes.

A potential direction for future study could be that of examining further optimizations that can be made to these algorithms, such as Stern's ISD algorithm, and ammending the current implementations in order to achieve results even when working with codes built on larger fields.

References

[1] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, Oct 1997.

[2] Mario Blaum. A short course on error-correcting codes, 2019.

[3] Marco Baldi, Alessandro Barenghi, Franco Chiaraluce, Gerardo Pelosi, and Paolo Santini. A finite regime analysis of information set decoding algorithms. *Algorithms*, 12(10), 2019.

[4] Christopher Roering. Coding theory-based cryptography: Mceliece cryptosystems in sage. 2013.

[5] Thomas Risse. How sage helps to implement goppa codes and mceliece pkcss. 2011.

[6] E. Berlekamp, R. McEliece, and H. van Tilborg. On the inherent intractability of certain coding problems (corresp.). *IEEE Transactions on Information Theory*, 24(3):384–386, 1978.

[7] S. Arora and B. Barak. *Computational Complexity—A Modern Approach.* 2009.

[8] H. NIEDERREITER. Knapsack-type cryptosystems and algebraic coding theory. *Prob. Contr. Inform. Theory*, 15(2):157–166, 1986.

[9] Christiane Peters. Information-set decoding for linear codes over f_q. In Nicolas Sendrier, editor, *Post-Quantum Cryptography, Third International Workshop, PQCrypto 2010, Darmstadt, Germany, May 25-28, 2010. Proceedings*, volume 6061 of *Lecture Notes in Computer Science*, pages 81–94. Springer, 2010.

[10] Robert Niebuhr, Edoardo Persichetti, Pierre-Louis Cayrel, Stanislav Bulygin, and Johannes Buchmann. On lower bounds for information set decoding over 61328_{q} and on the effect of partial knowledge. *Int. J. Inf. Coding Theory*, 4(1):47–78, 2017.