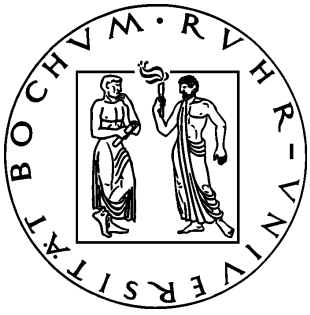
A Coding-Theoretic Approach to Cryptanalysis



Dissertation Thesis

by

Alexander Meurer
November 2012

Last modified 26.06.2013

Contents

1 Introduction **5**

2 Preliminaries **13**
2.1 Notation 13
2.2 Probability Theory 14
2.3 The q-ary Entropy Function 15
2.4 Coding Theory 18

3 Code-based Cryptography **25**
3.1 The Computational Syndrome Decoding Problem 25
3.2 Code-based One-way Functions 29

4 Information Set Decoding **33**
4.1 Introduction - Plain Information Set Decoding 33
4.2 Alternative Approaches 39
4.3 Stern's Algorithm and Ball-collision Decoding 43

5 A Generalised Model for Information Set Decoding **53**
5.1 Formal Definition and Basic Observations 56
5.2 Relations Between Recent Algorithms 61

6 Improved ISD Algorithms **85**
6.1 The Merge-Join Building Block 86
6.2 Beating Ball-Collision Decoding - A New ISD Algorithm 87
6.3 A Formal Proof of Superiority 98
6.4 Using Representations Iteratively 101
6.5 Discussion and Open Problems 109

7 Improved Information Set Decoding over $\mathbb{F}_q$ **113**
7.1 Stern's Algorithm 117
7.2 Using Representations over $\mathbb{F}_q$ 120
7.3 Conclusion and Open Problems 124

8 Applications **127**
8.1 Refined Complexity Analysis for $\mathbb{F}_2$ 128
8.2 Applications in Cryptanalysis 135

9 Noisy Integer Factorisation **147**
9.1 A Dyadic Factorisation Algorithm 150
9.2 A Noisy Integer Factorisation Algorithm 154
9.3 The Coding-Theoretic Link 161
9.4 Conclusion 165

Acknowledgements

First and for most, I would like to express my deepest gratitude to my supervisor Alexander May. Without his unwavering support and encouragement over the last three years, this work would not have been possible. Alex invested an immense amount of time to promote my scientific and personal development. I would like to thank him not only for giving important advice in scientific and technical issues but also for his great understanding in my personal situation, in particular during the last six month of my studies. Thank you very much.

I am also very grateful to Enrico Thomae who has been a very compassionate office mate over the last years. I thank him for the uncountable stimulating scientific discussions and for being a reliable co-author. But I also want to thank him for sharing a lot fun, for being a good friend and for pushing me to unimaginable sporting achievements!

Moreover, I am deeply indebted to Gottfried Herold for proofreading this thesis. In particular, he pointed out a technical issue that occurred in the runtime analysis of different algorithms presented in this work and he helped me out with finishing the important proof of Theorem 6.3.1. I also thank Saqib A. Kakvi for smoothing out the linguistic rough edges in this work. Furthermore, a special thank goes to all members, alumni and visitors of our group: Mathias Herrmann, Maike Ritzenhofen, Ilya Ozerov, Philipp Wagner, Carla Ràfols, Eike Kiltz, Stefan Heyse, Tibor Jager and everyone else missing in this list. A very special thanks goes to Marion Reinhardt-Kalender who helped me in navigating complex bureaucratic issues more than once.

During my PhD studies, I had the great honour to collaborate with many excellent researchers. In particular, I would like to thank all my co-authors for a trouble-free collaboration: Alexander May in [BJMM12, MMT11, HMM10], Anja Becker and Antoine Joux in [BJMM12] (I also thank Antoine for hosting my research visit in Versailles in May 2011), Enrico Thomae in [MMT11], Wilko Henecka in [HMM10] and Gottfried Herold in [HM12]. Besides this, I enjoyed fruitful scientific discussions with Christiane Peters (thank you for answering my annoying questions about your thesis and also for inviting me to Copenhagen), Dan Bernstein and Nicolas Sendrier (thank you for your interesting comments and suggestions on my work).

Furthermore, I am particularly thankful to the Ruhr-University Research School, funded by Germany's Excellence Initiative [DFG GSC 98/1], for the immense financial support of this work.

Most importantly, I would like to express my sincere gratitude to all my family and friends, most of all to my girlfriend Ana-Meritxell Joachim: I thank you for being who you are.

1 Introduction

A necessary condition for building modern cryptographic primitives, including secure encryption schemes and digital signatures, is the existence of *one-way functions*. Loosely speaking, a one-way function f is easy to compute but hard to invert (in an average-case sense). More formally,

- there is a polynomial-time algorithm that computes $f(x)$ given x,
- no probabilistic polynomial-time algorithm can compute a preimage of y under f with non-negligible probability (over the random choice of y and the algorithm's random coins).

There are generic constructions from one-way functions to powerful cryptographic primitives. One-way functions imply *pseudorandom generators* or *pseudorandom functions* which can further be used to construct even more powerful tools like CCA-secure private-key encryption. In contrast, for the public key setting the onewayness alone is *not* sufficient. More specifically, one often requires f to be injective and, as a further restriction, one needs to embed a *trapdoor*. Such a trapdoor allows the legitimate user to invert f efficiently while preserving the onewayness when the trapdoor is not known. Using generic transformations, e.g. *hardcore predicates*, one obtains CPA-secure public-key encryption in the standard model. If f is even a trapdoor *permutation* one can additionally construct CCA-secure public-key encryption and digital signatures in the random oracle model [BR93].

In summary, (trapdoor) one-way functions (and permutations) belong to the most important objects in cryptography. Consequently, the fact that we do not know how to prove the existence of one-way functions, even when making rather strong assumptions such as $\mathcal{P} \neq \mathcal{NP}$, has led cryptographers to establish a list of *candidate one-way functions* whose conjectured hardness is supported by extensive research in finding efficient inverting algorithms. Many of those candidates rely on number theoretic problems, namely:

- The well-known RSA function $f(x) = x^e \mod N$ or the Rabin Function $f(x) = x^2 \mod N$ rely on the hardness of the *factorisation problem*. In fact, both functions provide a *trapdoor* and can be made bijective by picking appropriate e and N thus providing *trapdoor one-way permuations* (cf. Chapter 9).

- The discrete exponentiation function $f(x) = g^x \mod p$ relies on the hardness of the *discrete logarithm problem* in the underlying cyclic group.

Unfortunately, both the factorisation and the discrete logarithm problem are known to be broken by quantum algorithms due to Shor [Sho97]. Although it is uncertain whether large-scale quantum computers will ever exist, the mere possibility must be considered a serious threat to many cryptographic primitives or protocols. Thus, a challenging task in cryptography is to come up with alternative candidate one-way function based on different, possibly quantum-resistant, assumptions. Besides the recently emerged field of *lattice-based cryptography* (cf. [MR08] for an introductory survey), the most serious proposals are related to the hardness of decoding random linear codes. For some modified code-based constructions it has not only been impossible to design efficient quantum algorithms thus far, but also there are serious indications that it will not be possible by using powerful techniques, e.g. quantum Fourier sampling, in the future [HMR11]. This renders code-based cryptography to one of the most promising research fields in post-quantum cryptography.

The most simple construction of a code-based one-way function uses random binary codes and can be described as follows (we assume that the reader is familiar with the basic coding-theoretic vocabulary and refer to Chapter 2 and 3 for clarification). For a random *generator matrix* $\mathbf{G} \in \mathbb{F}_2^{k\times n}$, simply define a function $f_{\mathbf{G}}$ that maps a tuple $(\mathbf{m}, \mathbf{e}) \in \mathbb{F}_2^k \times \mathbb{F}_2^n$ to a vector $\mathbf{c} \in \mathbb{F}_2^n$ by first computing a codeword $\mathbf{m}^\intercal\mathbf{G}$ and by then adding an appropriately chosen error vector $\mathbf{e} \in \mathbb{F}_2^n$ of Hamming weight ω. Given such a vector $\mathbf{c} = \mathbf{m}^\intercal\mathbf{G} + \mathbf{e}$, recovering the error vector $\mathbf{e}$ is sufficient in order to invert the function $f_{\mathbf{G}}$. Once $\mathbf{e}$ is known, the first part $\mathbf{m}$ of the input can easily be revealed by linear algebra. By elementary facts from coding theory (see Chapter 2 and 3), the problem of inverting $f_{\mathbf{G}}$ is equivalent to the (average case) hardness of the following computational problem.

Definition 1.0.1 (Computational Syndrome Decoding problem (CSD), informal)**.** Given a binary $(n-k)\times n$-matrix $\mathbf{H}$, a target weight $\omega \in \mathbb{N}$ and a vector $\mathbf{s} \in \mathbb{F}_2^{n-k}$, find a vector $\mathbf{e}$ of Hamming weight ω with $\mathbf{He} = \mathbf{s}$ (if such $\mathbf{e}$ exists).

Thus, the CSD problem can be seen as a purely combinatorial problem (we restate the problem in Definition 3.1.1 in its common coding-theoretic formulation) and the most naïve algorithm could simply enumerate all $\binom{n}{\omega}$ potential solutions. For an appropriate choice of the parameters n, k and ω, the CSD problem is hard and the most efficient algorithms run in time exponential in n (when k and ω are appropriately parametrised in n). Besides its relevance to the security of code-based cryptographic constructions, the average-case hardness of the CSD problem, or equivalently the hardness of decoding random binary codes, has been of great interest in coding theory in general. This sufficiently motivates the main objective of this work.

Goal: The design of asymptotically fastest generic decoding algorithms for random binary codes.

The major part of this work (Chapter 3 to 6) is devoted to achieving this goal and to discuss the practical impact of our improved algorithms to cryptography (Chapter 7 and 8). In the second (significantly) shorter part of this work (Chapter 9), we address a relatively unrelated question. We study the so-called **noisy integer factorisation problem** and thus turn back to one of the most important classical number theoretic computational problem used in cryptography, the *factorisation problem*. In simple terms, the noisy integer factorisation problem can be described as follows. Given a random RSA modulus $N = pq$ and two *noisy* copies $\tilde{p}, \tilde{q}$ of the unknown factorisation p, q, the task is to recover the prime factors p, q efficiently. The noise model in our work is simply the *binary symmetric channel*, i.e. each single bit of p, q is independently disturbed by a Bernoulli-distributed error e where $\mathbf{Pr}[e = 1] = \delta$ for some fixed noise rate $0 < \delta < \frac{1}{2}$. The study of this problem is practically motivated by so-called *side-channel attacks* where one obtains additional information in order to solve a theoretically hard computational problem. For example, such information is often gained from the physical implementation of a cryptosystem. As we will see, the noisy factorisation problem can be seen as the problem of decoding a particular (non-linear and highly non-random) code which allows to integrate the following second objective into the coding-theoretic framework of this thesis.

> **Goal: The construction of an efficient algorithm for the noisy integer factorisation problem.**

Main Part: Improved Generic Decoding Algorithms

We turn back to the major problem of our work which is to improve the performance of generic decoding algorithms. A very important class of generic decoding algorithms originates from the work of Prange in 1962 [Pra62], called **information set decoding**. These algorithms have attracted a lot of interest from coding theorists over the last 50 years. In this work, we will present improved variants of information set decoding. All improvements presented in this work are based on a generic technique that can be applied to a variety of combinatorial search problems. This technique, to which refer as the **representation technique**, was introduced by Howgrave-Graham and Joux at Eurocrypt 2010 [HGJ10] who presented improved generic algorithms for the well-known subset sum problem.

The Representation Technique

To understand the main idea behind the representation technique, let us first consider the following elementary algorithm for the CSD problem (which does not fit into the information set decoding framework). For simplicity, let us assume that ω is divisible by 2 and that there exists a unique solution $\mathbf{e}$ that can be written as $\mathbf{e} = (\mathbf{e}_1 || \mathbf{e}_2)$ where each vector $\mathbf{e}_i$ has length $\frac{n}{2}$ and Hamming weight $\frac{\omega}{2}$ (by $\mathbf{x}||\mathbf{y}$ we denote the concatenation of the vectors $\mathbf{x}$ and $\mathbf{y}$). By similarly splitting $\mathbf{H}$ into two halves $\mathbf{H}_1$ and $\mathbf{H}_2$ (which are both binary matrices of dimension $(n-k) \times \frac{n}{2}$) we can simply rewrite the equation $\mathbf{He} = \mathbf{s}$

as $\mathbf{H}_1\mathbf{e}_1 = \mathbf{H}_2\mathbf{e}_2 + \mathbf{s}$. In order to find the solution $\mathbf{e}$, we first compute a list $\mathcal{L}_1$ containing all candidates $\mathbf{e}_1$ of length $\frac{n}{2}$ and Hamming weight $\frac{\omega}{2}$ and order the list according to the values $\mathbf{H}_1\mathbf{e}_1$. We then proceed by searching for collisions in $\mathcal{L}_1$ for every candidate vector $\mathbf{e}_2$. That is, for every $\mathbf{e}_2$ we compute the value $\mathbf{H}_2\mathbf{e}_2 + \mathbf{s}$ and check whether there is an element $\mathbf{e}_1 \in \mathcal{L}_1$ with $\mathbf{H}_1\mathbf{e}_1 = \mathbf{H}_2\mathbf{e}_2 + \mathbf{s}$. Clearly, every collision $(\mathbf{e}_1, \mathbf{e}_2)$ yields a solution to $\mathbf{He} = \mathbf{s}$ and has the correct Hamming weight ω. Neglecting polynomial factors, the complexity of creating and sorting the list $\mathcal{L}_1$ is given by the number of possible $\mathbf{e}_1$ which is $\binom{n/2}{\omega/2}$. Similarly, the (worst-case) complexity of searching a collision can be estimated by the number of possible $\mathbf{e}_2$. We thus obtain an algorithm for the CSD problem with complexity $\binom{n/2}{\omega/2} \approx \binom{n}{\omega}^{1/2}$ and we have reduced the complexity of the naïve enumeration by almost a square-root factor.

Let us now see how the representation technique can be applied to obtain an improved algorithm. For this purpose, let us introduce the following notation. By $W_{n,\omega}$ we denote the discrete Hamming sphere in $\mathbb{F}_2^n$ of radius ω centred around $\mathbf{0}$. Note that the above algorithm solved the CSD problem by writing $W_{n,\omega}$ as a "direct sum" $W_{n,\omega} = W_1 \oplus W_2$ with appropriately defined sets W_1 and W_2. In particular, every element $\mathbf{e} \in W_{n,\omega}$ can be written *uniquely* as the sum of an element in W_1 with an element of W_2 and the above algorithm recovers a solution $\mathbf{e}$ by treating the set W_1 and W_2 "separately". As opposed to this, the representation technique chooses a different decomposition of $W_{n,\omega} \subset W_1 + W_2$ into W_1 and W_2 such that every element $\mathbf{e} \in W_{n,\omega}$ can be written in *many different ways* as the sum of an element in W_1 with an element of W_2. This simple modification yields to the following central definition.

Definition 1.0.2. Let $W_1, W_2 \subset W_{n,\omega}$ with $W_{n,\omega} \subset W_1 + W_2$ and $\mathbf{e} \in W_{n,\omega}$. Every pair $(\mathbf{e}_1, \mathbf{e}_2) \in W_1 \times W_2$ with $\mathbf{e} = \mathbf{e}_1 + \mathbf{e}_2$ is called a **representation** of $\mathbf{e}$.

For example, one can define $W_1 := W_2 := W_{n,\omega/2}$. Since $|W_1| = |W_2| = \binom{n}{\omega/2} \gg \binom{n/2}{\omega/2}$ this might look useless at a first sight. We significantly increased the size of the two separate search spaces W_i compared to the former "direct sum" construction. However, choosing W_1 and W_2 in this way introduces exactly $\rho := \binom{\omega}{\omega/2}$ different representations of the unique solution $\mathbf{e}$. The main idea of the representation technique is now very simple. In order to solve the original CSD problem, it is sufficient to find only *one* of the representations of the solution $\mathbf{e}$ and the main algorithmic problem is to find such a representation as efficiently as possible. More precisely, we aim to compute two lists $\mathcal{L}_1 \subset W_1$ and $\mathcal{L}_2 \subset W_2$ such that there exists at least one representation of $\mathbf{e}$ in $\mathcal{L}_1 \times \mathcal{L}_2$ with good probability. Since the number of representations is ρ and there is no additional structure to exploit, such lists can not be significantly smaller than

$$\frac{|W_i|}{\rho} = \frac{\binom{n}{\omega/2}}{\binom{\omega}{\omega/2}} \quad . \tag{1.1}$$

For the sake of argument, let us assume that we have access to an oracle that magically creates the lists $\mathcal{L}_i$ of minimal size (1.1). Given these lists, eventually finding a representation of $\mathbf{e}$ can be implemented by a straightforward collision search as explained

above. Neglecting polynomial factors again, the resulting algorithm would solve the CSD problem with complexity $|\mathcal{L}_i|$. Altogether, this would improve upon the preceding algorithm if

$$\frac{\binom{n}{\omega/2}}{\binom{\omega}{\omega/2}} \leq \binom{n/2}{\omega/2}$$

which holds for all $2 \leq \omega \leq n$. As we will see, this simplified description hides some important technical caveats. Most importantly, assuming the existence of an oracle that creates the lists $\mathcal{L}_i$ appears to be a very strong assumption. In fact, the main algorithmic contribution of our work is to present a recursive procedure that implements a slightly suboptimal oracle.

Our main result: Improved Information Set Decoding

So far, we did not tell how to employ the representation technique in the information set decoding framework. From a high-level perspective, an information set decoding algorithm transforms the original CSD instance $\mathbf{H}, \mathbf{e}$ with parameters n, k, ω into another, related CSD instance $\tilde{\mathbf{H}}, \tilde{\mathbf{e}}$ with decreased parameters $\tilde{n}, \tilde{k}, \tilde{\omega}$. One then solves the easier instance $\tilde{\mathbf{H}}, \tilde{\mathbf{e}}$ and tries to back transform the resulting solution(s) into a solution for the original problem. More precisely, the transformation is a randomised procedure that will mostly result in instances $\tilde{\mathbf{H}}, \tilde{\mathbf{e}}$ that do *not* provide any solution suitable to solve the original problem $\mathbf{H}, \mathbf{e}$. One resolves this problem by simply repeating the randomised transformation sufficiently many times.

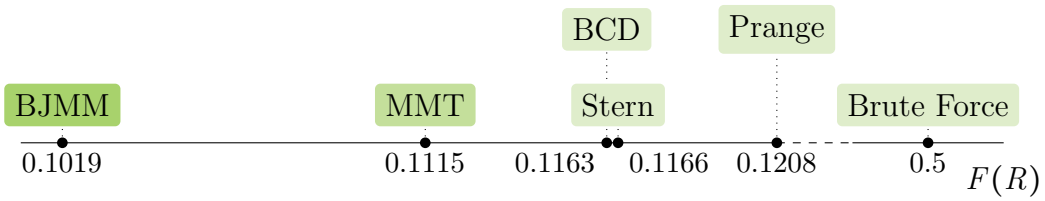


Figure 1.1: Recent development for "worst-case" complexity coefficients of information set decoding variants. MMT and BJMM have been published in [MMT11] and [BJMM12], respectively.

Over the last fifty years, many researches proposed different instantiations within this general framework. The first one is due to Prange [Pra62] who introduced the concept of information set decoding in 1962. Almost 30 years later, Lee and Brickell [LB88] and Leon [Leo88] showed how to improve upon Prange's algorithm by polynomial factors. Shortly after, Stern [Ste89] presented the first algorithm that offered an *exponential* improvement over Prange's original work. In the next twenty years, a variety of algorithms have been proposed that significantly improved the practical performance of information set decoding. However, none of these algorithms provided an exponential improvement over Stern's algorithm. More than twenty years later at Crypto 2011, Bernstein et al. presented a new method, called *ball-collision decoding*, that actually improved upon

Stern's algorithm by a little exponential factor [BLP11a]. As we will see in later chapters, there exist two alternative earlier proposals [Dum91] (due to Dumer) and [FS09] (due to Finiasz and Sendrier) that allow to achieve the same efficiency (but there existed no formal proof for this equivalence in the literature before our work). In this line of research, we presented two different improved information set decoding algorithms at Asiacrypt 2011 [MMT11] and Eurocrypt 2012 [BJMM12]. Both methods apply the representation technique, albeit to different extents, in order to solve the transformed problem $\tilde{\mathbf{H}}, \tilde{\mathbf{e}}$ more efficiently.

Let us conclude this first introductory part by emphasizing the *quantitative* improvement of our work. As we will explain in Chapter 3, it is possible to express the asymptotic running time of generic decoding algorithms as $2^{F(R)n+o(n)}$ where $F(R)$ is constant that only depends on the *rate* $R := \frac{k}{n}$ of the input code (where $0 < R < 1$). The constant $F(R)$ is often called the *complexity coefficient* and allows to compare the exponential behaviour of the respective algorithms. In Figure1.1 we compare the different complexity coefficients for the most important information set decoding algorithms.

Overview of Results and Thesis Organisation

We conclude the introduction with a more detailed overview of our results. We also explain the organisation of our work.

Chapter 2 and 3

The contribution of these two introductory chapters is twofold: We present the necessary basic concepts from coding theory and provide some useful technical tools from probability theory (Chapter 2). We further discuss some basic properties of the computational syndrome decoding problem and explain its use in cryptography (Chapter 3).

Chapter 4

We introduce the general concept of information set decoding and also discuss some alternative approaches for generic decoding algorithms. More specifically:

- We introduce Prange's original algorithm and use it as an example to discuss some technical caveats that arise in the analysis of information set decoding algorithms.
- We provide a general framework that covers the recent ball-collision decoding algorithm of Bernstein, Lange and Peters. This includes an extensive numerical optimisation of the algorithm parameters which will be used to (informally) compare BCD with our improved algorithms.

Chapter 5

For the sake of a formal competitive analysis of our improved algorithms, we present a generalised framework for information set decoding that serves the following purposes.

- The model yields a simplified version of Ball-collision decoding.
- It generalizes another ISD framework introduced by Finiasz and Sendrier in [FS09].
- It allows to relate different existing decoding methods.
- We instantiate a basic variant of our new algorithm within the framework.

In particular, embedding our improved algorithm into the general framework will be helpful in order to formally prove its superiority over BCD. The main technical result of this chapter is a running-time preserving parameter transformation that allows to turn optimal parameters of (a variant of) BCD into valid parameters of (a simplified variant of) our algorithm (Theorem 5.2.26).

Parts of this chapter appeared at Asiacrypt 2011 [MMT11] in a joint work with Alexander May and Enrico Thomae.

Chapter 6

This part contains the most important algorithmic contribution of this thesis. We propose different variants of information set decoding for random binary codes. All these variants are based on the representation technique and cover the algorithms presented in [MMT11] and [BJMM12]. We first provide a simplified variant of our algorithm that can easily be related to the general ISD framework of Chapter 5. We introduce this intermediate step in order to obtain a formal proof of superiority of our method over BCD. More specifically, one of our main results (Theorem6.3.1) shows that our method achieves a strictly smaller complexity coefficient $F(R)$ than BCD for any code rate $0 < R < 1$. Such a proof was completely missing in the original publication [BJMM12] and can be seen as one of the major theoretical contributions of our work. We then show how to further improve the simple variant by a recursive application of the representation technique which eventually yields the most efficient ISD algorithm to date (our presentation of the algorithm can be seen as a generalised description of the algorithm presented in [BJMM12] that allows for a simplified analysis).

Parts of this chapter appeared at Eurocrypt 2012 [BJMM12] in a joint work with Anja Becker, Antoine Joux and Alexander May.

Chapter 7 and 8

We provide a straightforward generalisation of the improved algorithm to larger base fields $\mathbb{F}_q$ and show that the improvement over more simple variants vanishes for growing q (Chapter 7). That is, the advantage of *any* generalised improved ISD algorithm over Prange's basic algorithm becomes arbitrarily small for large enough q. Moreover, in Chapter 8, we provide the first *practical* (heuristic) analysis for our improved algorithm. More specifically, we present a concrete simplified implementation of our algorithm that also takes into account several standard optimisations that reduce the running time

by polynomial factors. Altogether, we are able to derive a manageable formula for the binary work factor of our algorithm. We will then use this refined analysis in order to estimate the practical impact of our work for some interesting cryptographic assumptions. Namely, we show that:

- Long-term security levels of the *McEliece OWF* are noticeably affected by our work.
- Almost all low-noise instances of the *computational LPN problem* can be broken efficiently by ISD algorithms.

Chapter 9

We present a (heuristic) polynomial time algorithm for the noisy integer factorisation problem for noise rates $\delta < 0.0837$. Compared to [HMM10], where we presented a similar algorithm for a more general scenario related to the RSA trapdoor one-way permutation, the main focus in this work lies on the following two points.

- We point out a technical caveat of our original analysis in [HMM10] and provide an extended analysis that fixes the problem.
- We discuss a recently discovered coding-theoretic interpretation of the noisy factorisation problem that was introduced by Paterson et al. at Asiacrypt 2012 [PPS12].

The coding-theoretic link allows to derive heuristic upper bounds on the noise rate δ. For a particular class of *list-decoding* algorithms, one can not recover the factorisation when the error exceeds $\delta > 0.111$.

Parts of this chapter appeared at Crypto 2010 [HMM10] in a joint work with Wilko Henecka and Alexander May.

To summarize, in this work we study different coding-theoretic tools and discuss their applicability to various cryptographically motivated problems. Note that the major part of this work, the study of the hardness of the CSD problem, is related to one of the most promising candidates for post-quantum cryptography. However, we restrict our attention to the design of *classical* algorithms and completely ignore the question *whether efficient quantum attacks on coding-theoretic problems exist*. For example, studying the applicability of recent quantum techniques (e.g. quantum random walks) to known decoding algorithms is a promising (and already initiated) direction for future research that goes beyond the scope of this work.

2 Preliminaries

In this chapter, we will introduce some general notation (Section 2.1) and give basic definitions and results from probability theory (Section 2.2) and coding theory (Section 2.4) with a strong focus on properties of *random linear codes*.

2.1 Notation

Throughout the whole thesis, $\mathbb{F}_q$ denotes a finite field with q elements, i.e. $q = p^r$ is a prime power. We will mostly consider the *binary* case, i.e. $q = 2$. To clarify notation, we denote matrices and vectors by bold face letters, e.g. $\mathbf{G} \in \mathbb{F}_q^{k\times n}$ or $\mathbf{m} \in \mathbb{F}_q^k$, and we write all vectors as column vectors for convenience, i.e. $\mathbf{m} \in \mathbb{F}_q^{k\times 1} =: \mathbb{F}_q^k$. By $\mathbf{m}^\top \in \mathbb{F}_q^{1\times k}$ and $\mathbf{G}^\top \in \mathbb{F}_q^{n\times k}$ we denote the transpose of a vector $\mathbf{m} \in \mathbb{F}_q^k$ and matrix $\mathbf{G} \in \mathbb{F}_q^{k\times n}$, respectively.

By $\mathrm{d}(\mathbf{x}, \mathbf{y}) := |\{i \in [n] : x_i \neq y_i\}|$ we denote the **Hamming distance** of $\mathbf{x}, \mathbf{y} \in \mathbb{F}_q^n$. The Hamming distance can be extended to sets in the usual way, i.e. $\mathrm{d}(\mathbf{x}, \mathcal{S}) := \min_{\mathbf{y}\in\mathcal{S}} \mathrm{d}(\mathbf{x}, \mathbf{y})$ for an arbitrary set $\mathcal{S} \subset \mathbb{F}_q^n$. Moreover, let

$$\mathcal{B}_q(n, r) := \{\mathbf{e} \in \mathbb{F}_q^n : \mathrm{wt}(\mathbf{e}) \leq r\} \tag{2.1}$$

denote the n-dimensional **(Hamming) ball** of radius r where $\mathrm{wt}(\mathbf{e}) := \mathrm{d}(\mathbf{e}, \mathbf{0})$ is the **Hamming weight**. Sometimes we also make use of the n-dimensional **(Hamming) sphere**

$$W_{n,r}^q := \{\mathbf{e} \in \mathbb{F}_q^n : \mathrm{wt}(\mathbf{e}) = r\} \ , \tag{2.2}$$

i.e. the set containing all vectors of length n and Hamming weight *exactly* r. We will write $\mathcal{B}(n, r)$ and $W_{n,r}$ for the binary case (the unconventional notation $W_{n,r}$ is frequently used in the code-based cryptography literature). The volume of $\mathcal{B}_q(n, r)$ is denoted by $\mathrm{vol}_q(n, r)$ which is defined as

$$\mathrm{vol}_q(n, r) := \sum_{0\leq i\leq r} \binom{n}{i} (q-1)^i \ . \tag{2.3}$$

An appropriate asymptotic estimate for the volume of such balls helps to study properties of random linear codes and allows to analyse the running time of many of the proposed decoding algorithms. We provide such an estimate based on the q-ary entropy function in Section 2.3.

With $\log_q(x)$ we denote the logarithm to base q and we write $\log(x) := \log_2(x)$ and $\ln(x) := \log_e(x)$ (where e is Euler's number) for convenience.

2.2 Probability Theory

Almost all algorithms presented in this thesis are *probabilistic* or their performance is studied on randomly chosen input instances of a particular computational problem. In both cases, the analysis requires some basic tools from probability theory. For this purpose, let $\Omega = \{\omega_1, \ldots, \omega_m\}$ be a *finite* sample space with probability function $p : \Omega \to [0,1]$, i.e. $\sum_{i=1}^{m} p(\omega_i) = 1$. This implicitly defines a probability space (Ω, P) with probability measure $P : 2^\Omega \to [0,1]$, $P(A) := \sum_{\omega \in A} p(\omega)$ over the trivial σ-algebra $2^\Omega := \{A : A \subset \Omega\}$. Let $X : \Omega \to \mathbb{R}$ denote a (discrete) random variable on (Ω, P) with range $\{x_1, \ldots, x_n\}$, i.e. $p_i := \mathbf{Pr}[X = x_i] := P(X^{-1}(x_i))$ where $X^{-1}(x_i) = \{\omega \in \Omega : X(\omega) = x_i\}$ is the preimage of x_i under X. As usual, we define the first two moments of X as

$$\mathbb{E}[X] := \sum_{i=1}^{n} p_i x_i \tag{2.4}$$

and

$$\mathrm{Var}[X] := \mathbb{E}\left[(X - \mathbb{E}[X])^2\right] = \mathbb{E}\left[X^2\right] - \mathbb{E}[X]^2 \geq 0 \ . \tag{2.5}$$

We will often make use of the *linearity of expectation*, i.e. for two random variables X, Y and $a \in \mathbb{R}$ it holds

$$\mathbb{E}[aX + Y] = a\mathbb{E}[X] + \mathbb{E}[Y] \ . \tag{2.6}$$

In particular, $\mathbb{E}[\sum_{i=1}^{n} X_i] = \sum_{i=1}^{n} \mathbb{E}[X_i]$ for n random variables $X_1, \ldots, X_n$. Computing the variance of a sum of random variables is slightly more complicated due to dependencies amongst the X_i, i.e

$$\mathrm{Var}\left[\sum_{i=1}^{n} X_i\right] = \sum_{i,j=1}^{n} \mathrm{Cov}[X_i, X_j] = \sum_{i=1}^{n} \mathrm{Var}[X_i] + \sum_{i \neq j} \mathrm{Cov}[X_i, X_j] \tag{2.7}$$

where $\mathrm{Cov}[X, Y] := \mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y]$. Note that $\mathrm{Var}[\sum_{i=1}^{n} X_i] = \sum_{i=1}^{n} \mathrm{Var}[X_i]$ for *pairwise independent* X_i, i.e. $\mathbb{E}[X_i X_j] = \mathbb{E}[X_i]\mathbb{E}[X_j]$ for all $i \neq j$. A useful (rough) upper bound for the variance of a sum of random variables is given by the next lemma.

Lemma 2.2.1.

$$\mathrm{Var}\left[\sum_{i=1}^{n} X_i\right] \leq n^2 \max_i \mathrm{Var}[X_i]$$

Proof. Applying the Cauchy-Schwartz inequality $|\mathrm{Cov}[X, Y]|^2 \leq \mathrm{Var}[X]\mathrm{Var}[Y]$ yields

$$\begin{aligned} \mathrm{Var}\left[\sum_{i=1}^{n} X_i\right] &= \sum_{i,j=1}^{n} \mathrm{Cov}[X_i, X_j] \\ &\leq \sum_{i,j=1}^{n} \sqrt{\mathrm{Var}[X_i]\mathrm{Var}[X_j]} \leq n^2 \max_i \mathrm{Var}[X_i] \ . \end{aligned}$$

□

When analyzing probabilistic algorithms, one often needs to upper bound a certain number of elements that have to be processed at a particular step of the respective algorithm. This number might depend on the algorithm's randomness or the randomly chosen input instance and can appropriately be modelled by a random variable X. The following *tail inequalities* provide upper bounds on the probability that X exceeds a certain value. We omit all proofs which can be found in almost every standard textbook about probability theory, e.g. [Bau96].

Theorem 2.2.2 (Markov's inequality)**.** *For any random variable X and $a > 0$ it holds*

$$\mathbf{Pr}\left[|X| \geq a\right] \leq \frac{\mathbb{E}\left[|X|\right]}{a} \quad . \tag{2.8}$$

By applying Markov's inequality to the random variable $|X - \mathbb{E}[X]|^2$ one obtains the following

Theorem 2.2.3 (Chebychev's inequality)**.** *For any random variable X and $a > 0$ it holds*

$$\mathbf{Pr}\left[|X - \mathbb{E}\left[X\right]| \geq a\right] \leq \frac{\mathrm{Var}\left[X\right]}{a^2} \quad . \tag{2.9}$$

In order to bound the success or failure probability of probabilistic algorithms, sums of *indicator random variables* X_i are particularly interesting. That is, the range of the X_i is $\{0,1\}$ (indicating success or failure in a particular step of the algorithm) and we have $p_i := \mathbf{Pr}\left[X_i = 1\right] = \mathbb{E}\left[X_i\right]$. When all X_i are *independent Bernoulli trials* with identical success probability $p := \mathbf{Pr}\left[X_i = i\right]$, their sum $X := \sum_{i=1}^{n} X_i$ fulfils $\mathbb{E}\left[X\right] = np$ and the following important tail inequality gives an upper bound (exponentially decreasing in n) on the probability that X deviates from its expectation.

Theorem 2.2.4 (Hoeffding's inequality)**.** *Let $X_1, \ldots, X_n$ be a sequence of independent Bernoulli trials with identical success probability $\mathbf{Pr}\left[X_i = 1\right] = p$ for all i. Define $X := \sum_{i=1}^{n} X_i$. Then for every $0 < \gamma < 1$ we have*

i) $\mathbf{Pr}\left[X \geq n(p+\gamma)\right] \leq e^{-2n\gamma^2}$,

ii) $\mathbf{Pr}\left[X \leq n(p-\gamma)\right] \leq e^{-2n\gamma^2}$.

2.3 The q-ary Entropy Function

The main goal of this section is to obtain an asymptotically tight estimate for $\mathrm{vol}_q(n, r)$ as defined in Eq.(2.3). The appropriate technical tool for this purpose is the q-ary entropy function as defined next.

Definition 2.3.1. For $q \geq 2$ we define the **q-ary entropy function** $\mathrm{H}_q : [0,1] \to \mathbb{R}$ as

$$\mathrm{H}_q(x) := x \log_q(q-1) - x \log_q x - (1-x) \log_q(1-x) \ .$$

Particularly interesting is the **binary entropy function**

$$\mathrm{H}(x) := \mathrm{H}_2(x) = -x \log x - (1-x) \log(1-x) \ .$$

Note that we implicitly extend expressions of the form $x \log x$ continuously (but not differentiably) to 0 at $x = 0$, i.e. we obtain $\mathrm{H}(0) = \mathrm{H}(1) = \mathrm{H}_q(0) = 0$ and $\mathrm{H}_q(1) = \log_q(q-1)$. The binary entropy function is thus continuous and *symmetric* around $x = \frac{1}{2}$, i.e. $\mathrm{H}\left(\frac{1}{2} + x\right) = \mathrm{H}\left(\frac{1}{2} - x\right)$ for $x \in [0, \frac{1}{2}]$. The q-ary entropy function is also continuous and monotone increasing for $x \in [0, 1 - \frac{1}{q}]$ with $\mathrm{H}_q(0) = 0$ and $\mathrm{H}_q(1 - \frac{1}{q}) = 1$, see Figure 2.3 for an illustration.

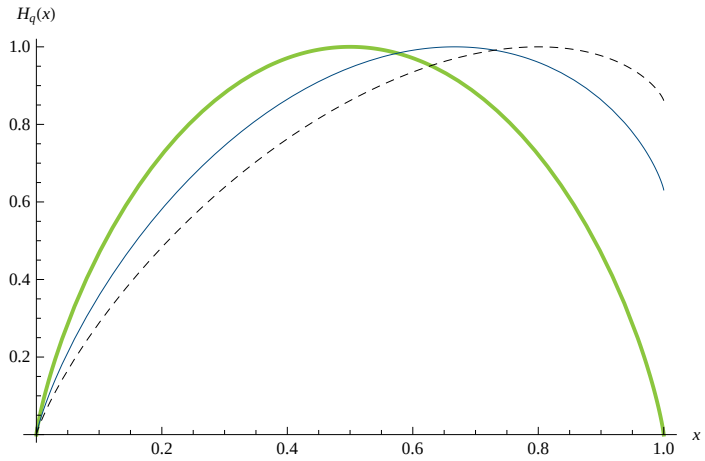


Figure 2.1: q-ary entropy function $\mathrm{H}_q(x)$ for $q = 2$ (thick curve), $q = 3$ (thin curve) and $q = 5$ (dashed curve)

If necessary, we can define the inverse of H_q by setting $\mathrm{H}_q^{-1}(y)$ to the *unique* $x \in [0, 1-\frac{1}{q}]$ satisfying $\mathrm{H}_q(x) = y$, for $y \in [0,1)$. Furthermore observe that

$$\mathrm{H}_q(x) = x \log_q(q-1) + \mathrm{H}(x) \log_q 2 \ . \tag{2.10}$$

H is differentiable for all $x \in (0,1)$ and its first derivative is given by

$$\frac{d\,\mathrm{H}}{dx} = \log(1-x) - \log(x) \ . \tag{2.11}$$

Another useful fact is that the entropy function is strictly concave, i.e.

$$\lambda \mathrm{H}_q(x) + (1-\lambda)\mathrm{H}_q(y) < \mathrm{H}_q(\lambda x + (1-\lambda)y) \tag{2.12}$$

for all $0 \leq x < y \leq 1$ and $\lambda \in (0,1)$. An immediate consequence is the next lemma which turns out to be very useful in many proofs of subsequent chapters. It can be seen as a q-ary asymptotic version of *Vandermonde's identity*.

Lemma 2.3.2 (Vandermonde's identity)**.** *Let $R, L > 0$. For $0 \le P \le R$ and $0 \le Q \le L$ it holds*

$$R\,\mathrm{H}_q\left(\frac{P}{R}\right) + L\,\mathrm{H}_q\left(\frac{Q}{L}\right) \le (R+L)\,\mathrm{H}_q\left(\frac{P+Q}{R+L}\right) \quad . \tag{2.13}$$

Furthermore, equality holds iff $\frac{P}{R} = \frac{Q}{L}$.

Proof. W.l.o.g. $\frac{P}{R} \le \frac{Q}{L}$. Consider Eq.(2.12) for $x = \frac{P}{R}$, $y = \frac{Q}{L}$ and $\lambda = \frac{R}{R+L}$. Multiplying both sides of the resulting equation with $R + L$ gives Eq.(2.13). Obviously $x = y$ implies equality. On the other hand, $R, L > 0$ implies $\lambda \in (0,1)$ and thus $x \ne y$ implies strict inequality.

□

In all the asymptotic analysis we will make heavy use of the following approximation formulas.

Theorem 2.3.3 (Stirling's formula)**.**

$$\begin{aligned}\log n! &= (n - \frac{1}{2})\log n - n + \frac{1}{2}\log(2\pi) + \mathrm{o}(1)\\ &= n\log n - n + \mathcal{O}(\log n)\end{aligned}$$

for $n \to \infty$.

Corollary 2.3.4. *For $\alpha \in [0,1]$ it holds*

$$\frac{1}{n}\log\binom{n}{\lfloor \alpha n \rfloor} = \mathrm{H}(\alpha) + \mathrm{o}(1) \tag{2.14}$$

for $n \to \infty$.

Proof. Denote $m := \lfloor \alpha n \rfloor$ and observe $m = \alpha n + \mathcal{O}(1)$ for $n \to \infty$. Using Theorem 2.3.3 yields

$$\begin{aligned}\frac{1}{n}\log\binom{n}{m} &= \frac{1}{n}\left[n\log n - m\log m - (n-m)\log(n-m) + \mathrm{o}(n)\right]\\ &= \log n - \alpha\log(\alpha n) - (1-\alpha)\log((1-\alpha)n) + \mathrm{o}(1) = \mathrm{H}(\alpha) + \mathrm{o}(1) \quad .\end{aligned}$$

□

Analogously it follows

$$\frac{1}{n}\log\binom{\lfloor \alpha n \rfloor}{\lfloor \beta n \rfloor} = \alpha\,\mathrm{H}(\beta/\alpha) + \mathrm{o}(1) \tag{2.15}$$

for sufficiently large n where $\alpha \in [0,1]$ and $\beta \in [0,\alpha]$.

Using Eq.(2.14) we can now give a sharp asymptotic estimate for $\mathrm{vol}_q(n, r)$ as defined in Eq.(2.3).

Lemma 2.3.5. *Let $q \geq 2$ be an integer and $r \in [0, 1 - \frac{1}{q}]$. Then*

$$\mathrm{vol}_q(n, rn) = q^{\mathrm{H}_q(r)n + \mathrm{o}(n)} \tag{2.16}$$

for $n \to \infty$.

Proof. One computes

$$\frac{\mathrm{vol}_q(n, rn)}{q^{\mathrm{H}_q(r)n}} = \sum_{0 \leq i \leq rn} \binom{n}{i} (q-1)^i (1-r)^n \left(\frac{r}{(q-1)(1-r)} \right)^{rn} .$$

Since $r \leq 1 - \frac{1}{q}$, it follows $\frac{r}{q-1} \leq \frac{1}{q} \leq 1 - r$ which implies $\frac{r}{(q-1)(1-r)} \leq 1$. Thus the above expression can be upper bounded by

$$\sum_{0 \leq i \leq rn} \binom{n}{i} (q-1)^i (1-r)^n \left(\frac{r}{(q-1)(1-r)} \right)^i = \sum_{0 \leq i \leq rn} \binom{n}{i} (1-r)^{n-i} r^i \leq 1$$

according to the binomial theorem which already gives $\mathrm{vol}_q(n, rn) \leq q^{\mathrm{H}_q(r)n}$. The remaining direction follows by using Eqs.(2.10) and (2.14) since

$$\begin{aligned} \mathrm{vol}_q(n, rn) \geq \binom{n}{\lfloor rn \rfloor} (q-1)^{rn} &= 2^{\mathrm{H}(r)n + \mathrm{o}(n)} q^{r \log_q(q-1)} \\ &= q^{[\mathrm{H}(r) \log_q 2 + r \log_q(q-1)]n + \mathrm{o}(n)} = q^{\mathrm{H}_q(r)n + \mathrm{o}(n)} . \end{aligned}$$

□

2.4 Coding Theory

In the introduction, we stated the CSD problem as a purely combinatorial problem. Given a (binary) matrix $\mathbf{H} \in \mathbb{F}_2^{(n-k) \times n}$, a (binary) vector $\mathbf{s} \in \mathbb{F}_2^{n-k}$ and a target weight $\omega \in \mathbb{N}$, find a solution $\mathbf{e}$ to $\mathbf{He} = \mathbf{s}$ of Hamming weight ω. In this section, we highlight its relation to coding theory. Therefore, we first provide basic definitions for linear codes and introduce the *minimum distance decoding problem* which can easily be related to the CSD problem. We then study some fundamental properties of *random* linear codes which will help to understand the hardness of the CSD problem for random linear codes in relation to the parameters of the underlying code. The key ingredient is to study the distribution of the number of codewords of a particular Hamming weight in random linear codes. Moreover, we provide some helpful results that will simplify the later analysis of our algorithms.

2.4.1 Linear Codes: Basic Definitions

We will now recall some basic definitions for linear codes. For a more in-depth presentation we refer to [vL98] or [Bar98]. The latter is an excellent reference for complexity issues in coding theory. We start with the following simple definition.

Definition 2.4.1 (Linear code)**.** A **q-ary linear code** $\mathcal{C}$ is a linear subspace of the vector space $\mathbb{F}_q^n$. If $\mathcal{C}$ has dimension k, we call $\mathcal{C}$ an $[n, k]$ code. The ratio $R := \frac{k}{n}$ is called the **(information) rate** of $\mathcal{C}$.

One usually specifies a linear code by the row-space of a $k \times n$ **generator matrix** $\mathbf{G}$, i.e. $\mathcal{C} = \{\mathbf{m}^\top \mathbf{G} : \mathbf{m} \in \mathbb{F}_q^k\}$, or as the kernel of an $(n-k) \times n$ **parity check matrix** $\mathbf{H}$, i.e. $\mathcal{C} = \{\mathbf{c} \in \mathbb{F}_q^n : \mathbf{H}\mathbf{c} = \mathbf{0}\}$. By applying Gaussian elimination to the rows of $\mathbf{G}$, one can always transform a generator matrix into *standard form*, i.e.

$$\mathbf{G} = \begin{pmatrix} \mathbf{I}_k & \mathbf{Q} \end{pmatrix} \tag{2.17}$$

where $\mathbf{I}_k$ is the k-dimensional identity matrix and $\mathbf{Q} \in \mathbb{F}_q^{k \times (n-k)}$. It is easy to see that $\mathbf{G}$ in standard form directly gives a parity check matrix $\mathbf{H} = \begin{pmatrix} -\mathbf{Q}^\top & \mathbf{I}_{n-k} \end{pmatrix}$ since (2.17) implies $\mathbf{G}\mathbf{H}^\top = \mathbf{0}$. Thus we obtain the following simple statement.

Lemma 2.4.2. *A generator matrix of an $[n, k]$-code can be transformed into a parity check matrix in time $\mathcal{O}(n^3)$, and vice versa.*

Every $[n, k]$-code $\mathcal{C}$ trivially gives an injective encoding $\mathsf{Encode} : \mathbb{F}_q^k \to \mathbb{F}_q^n$ defined by $\mathbf{m} \mapsto \mathbf{m}^\top \mathbf{G}$ which can be efficiently computed in time $\mathcal{O}(n^2)$. The crucial question in coding theory is how to define an efficient decoding function $\mathsf{Decode} : \mathbb{F}_q^n \to \mathcal{C}$ such that

$$\mathrm{d}(\mathbf{x}, \mathsf{Decode}(\mathbf{x})) = \mathrm{d}(\mathbf{x}, \mathcal{C})$$

for all $\mathbf{x} \in \mathbb{F}_q^n$. That is, the decoding function should assign a *closest* codeword to a given word $\mathbf{x} \in \mathbb{F}_q^n$. Clearly, finding a closest codeword to $\mathbf{x}$ can be equivalently formulated as follows:

i) Find a minimal weight representative of the coset $\mathbf{x} + \mathcal{C}$ (such an element is often called a **coset leader**).

ii) Find a minimal weight solution $\mathbf{e}$ to the equation $\mathbf{H}\mathbf{e} = \mathbf{s}$ where $\mathbf{s} := \mathbf{H}\mathbf{x}$ is called the **syndrome** of $\mathbf{x}$.

The problem of finding the closest codeword to $\mathbf{x}$ is called the **complete** or **minimum distance decoding problem** (MDD for short). Note that ii) already indicates its connection to the CSD problem. In the next definition, we will introduce two fundamental parameters of linear codes which are related to the decoding problem.

Definition 2.4.3. (Minimum distance, Covering radius) The **minimum distance** d of a linear $[n,k]$ code $\mathcal{C}$ is defined as the minimum Hamming-distance between every two distinct codewords, i.e.

$$d := d(\mathcal{C}) := \min_{\mathbf{x}\neq\mathbf{y}\in\mathcal{C}} \mathrm{d}(\mathbf{x},\mathbf{y}) = \min_{\mathbf{c}\in\mathcal{C}\setminus\{\mathbf{0}\}} \mathrm{wt}(\mathbf{c}) \ .$$

The **covering radius** $\rho(\mathcal{C})$ of $\mathcal{C}$ is

$$\rho := \rho(\mathcal{C}) := \max_{\mathbf{x}\in\mathbb{F}_q^n} \mathrm{d}(\mathbf{x},\mathcal{C}) \ .$$

We call a decoding function ω**-error correcting**, if for all error patterns $\mathbf{e} \in \mathcal{B}_q(n,\omega)$ and all $\mathbf{c} \in \mathcal{C}$ it holds $\mathsf{Decode}(\mathbf{c}+\mathbf{e}) = \mathbf{c}$. This means one can correctly decode any codeword that is perturbed by arbitrary error vectors of weight $\leq \omega$. Note that $\omega := \left\lfloor \frac{d-1}{2} \right\rfloor$ is the largest radius such that the balls $\mathbf{c}+\mathcal{B}_q(n,\omega)$ centered around all codewords are *disjoint*. Consequently, *all* error patterns in $\mathcal{B}_q(n,\omega)$ can uniquely be decoded and thus ω is the **error capability** of the code (by maximality of ω there are at least two distinct codewords with intersecting Hamming balls of radius $\omega+1$ and decoding will fail for some elements in the intersection). Similarly, the covering radius ρ resembles the smallest possible radius such that $\mathbb{F}_q^n = \bigcup_{\mathbf{c}\in\mathcal{C}}\{\mathbf{c}+\mathcal{B}_q(n,\rho)\}$, i.e. the Hamming balls of radius ρ centered around the codewords cover the entire space. Knowing the covering radius of a code helps to understand the complexity of the complete decoding problem due to the following trivial observation which relates the maximal weight of a coset leader to $\rho(C)$.

Lemma 2.4.4.

$$\rho(\mathcal{C}) = \max_{\mathbf{x}\in\mathbb{F}_q^n} \min_{\mathbf{y}\in\mathbf{x}+\mathcal{C}} \mathrm{wt}(\mathbf{y})$$

Proof. Follows immediately from $\min_{\mathbf{c}\in\mathcal{C}} \mathrm{wt}(\mathbf{x}+\mathbf{c}) = \mathrm{d}(\mathbf{x},\mathcal{C})$. □

2.4.2 Properties of Random Linear Codes

Since we will later compare the asymptotic complexity of decoding algorithms, we need to study the typical behaviour of *long codes*, namely families $\{\mathcal{C}_n\}_{n\in\mathbb{N}}$ of codes with growing length n and dimension $k := k(n)$ (where we consider q to be constant independent of n). Particularly interesting is the ensemble of **random linear codes** of *constant* rate $0 < R < 1$, i.e. the ensemble of codes given by parity check matrices $\mathbf{H} \in_R \mathbb{F}_q^{(n-k(n))\times n}$ with $\lim_{n\to\infty} \frac{k(n)}{n} = R$ whose entries are chosen uniformly at random. Note that $R > 0$ implies that a random matrix of dimension $(n-k(n)) \times n$ will have full (row) rank $n-k$ with probability $1-e^{-\Omega(n)}$ for sufficiently large n. In this work, we will study the probability that a certain property is satisfied by a randomly chosen $[n,k]$-code. We will say that **almost all** codes satisfy a certain property if this probability converges to

1 exponentially in n for $n \to \infty$. For simplicity we will stick to codes with *constant rate* and *constant minimum distance*, i.e. codes where $\lim_{n\to\infty} \frac{k(n)}{n} = R$ and $\lim_{n\to\infty} \frac{d(n)}{n} = D$ for some non-zero $R, D \in \mathbb{R}$, respectively. We will abusively call R and D the rate and minimum distance of $\{\mathcal{C}_n\}_{n\in\mathbb{N}}$.

Remark 2.4.5. In coding theory, it is standard to study the asymptotic properties of families $\{\mathcal{C}_n\}_{n\in\mathbb{N}}$ with *constant* rate and minimum distance. We point out that this is not exactly compatible with the study of many codes used in code-based cryptography, e.g. binary Goppa codes provide $D = \frac{1-R}{\log n}$ and thus D slowly decreases to 0 as $n \to \infty$. Nevertheless, our algorithms will also provide better running times for such families.

Let $\mathcal{C}$ denote a randomly chosen code of length n and dimension k with parity check matrix $\mathbf{H}$. We start with a very simple observation about random linear codes: For a fixed vector $\mathbf{x} \in \mathbb{F}_q^n$ and syndrome $\mathbf{s} \in \mathbb{F}_q^{n-k}$ we have

$$\mathbf{Pr}\left[\mathbf{Hx} = \mathbf{s}\right] = \frac{1}{q^{n-k}} \tag{2.18}$$

where the probability space is defined over the random choice $\mathbf{H} \in_R \mathbb{F}_q^{(n-k)\times n}$. A single equation $\langle \mathbf{h}_i, \mathbf{x}\rangle = s_i$ holds with probability $\frac{1}{q}$ and all $n-k$ equations are independent. In particular, we have $\mathbf{Pr}\left[\mathbf{x} \in \mathcal{C}\right] = \frac{1}{q^{n-k}}$.

We now formulate the main technical lemma that gives the first two moments of the following random variable: Let $\mathcal{S} \subset \mathbb{F}_q^n$ and define $N(\mathcal{S}) := |\mathcal{S} \cap \mathcal{C}|$, i.e. the number of codewords contained in $\mathcal{S}$. Clearly $N(\mathcal{S})$ is a random variable (over the random choice of $\mathcal{C}$). Using Eq.(2.18) the following statement is easy to prove.

Lemma 2.4.6. *Suppose* $0 \notin \mathcal{S}$. *Then*

i) $\mathbb{E}\left[N(\mathcal{S})\right] = \frac{|\mathcal{S}|}{q^{n-k}}$

ii) $\mathrm{Var}\left[N(\mathcal{S})\right] \le \frac{(q-1)|\mathcal{S}|}{q^{n-k}} \le (q-1)\mathbb{E}\left[N(\mathcal{S})\right]$.

Proof. Set $\sigma := |\mathcal{S}|$ and arbitrarily order the elements of $\mathcal{S} = \{\mathbf{x}_1, \ldots, \mathbf{x}_\sigma\}$. Define $\{0,1\}$-variables χ_i where $\chi_i = 1$ iff $\mathbf{x}_i \in \mathcal{C}$, i.e. $N(\mathcal{S}) = \sum \chi_i$. Clearly, $\mathbb{E}\left[\chi_i\right] = q^{-(n-k)}$ and i) follows by linearity of expectation. For ii) we need to deal with dependencies amongst the χ_i. One computes

$$\mathrm{Var}\left[N(\mathcal{S})\right] = \sum_{i=1}^{\sigma} \mathrm{Var}\left[\chi_i\right] + \sum_{i\neq j} \mathrm{Cov}\left[\chi_i, \chi_j\right] \tag{2.19}$$

and further expands the first sum into

$$\sum_{i=1}^{\sigma} \mathbb{E}\left[\chi_i\right] - \mathbb{E}\left[\chi_i\right]^2 = \frac{\sigma}{q^{n-k}}\left(1 - \frac{1}{q^{n-k}}\right) \tag{2.20}$$

using $\chi_i^2 = \chi_i$. If $\mathbf{x}_i$ and $\mathbf{x}_j$ are linearly independent then χ_i and χ_j are independent random variables and thus $\mathrm{Cov}[\chi_i, \chi_j] = 0$. To upper bound the second sum in Eq.(2.19) we can simply assume the worst-case, i.e. for every $\mathbf{x} \in \mathcal{S}$ we also have $\lambda\mathbf{x} \in \mathcal{S}$ for all $\lambda \in \mathbb{F}_q^*$. For those $\mathbf{x}_i = \lambda\mathbf{x}_j$ we further have $\chi_i = \chi_j$ and thus $\mathrm{Cov}[\chi_i, \chi_j] = \mathrm{Var}[\chi_i]$ which gives

$$\sum_{i \neq j} \mathrm{Cov}[\chi_i, \chi_j] \leq \sigma(q-2)\,\mathrm{Var}[\chi_i] = \sigma(q-2)\frac{1}{q^{n-k}}\left(1 - \frac{1}{q^{n-k}}\right) \ . \tag{2.21}$$

Combining Eqs.(2.19)-(2.21) finally proves ii). □

Remark 2.4.7. A similar analysis can be done for the case $0 \in \mathcal{S}$ and yields

i) $\mathbb{E}[N(\mathcal{S})] = \frac{|\mathcal{S}|-1}{q^{n-k}} + 1$

ii) $\mathrm{Var}[N(\mathcal{S})] \leq \frac{(q-1)(|\mathcal{S}|-1)}{q^{n-k}} \leq (q-1)\mathbb{E}[N(\mathcal{S})]$.

Also note that Lemma 2.4.6 can be easily generalised to the random variable $N(\mathcal{S}) := |\mathcal{S} \cap (\mathbf{x} + \mathcal{C})|$ for any $\mathbf{x} \in \mathcal{C}$, i.e. we can obtain similar estimates for the expected number of elements in $\mathcal{S}$ that are contained in a fixed coset of $\mathcal{C}$.

Moreover, Lemma 2.4.6 allows to study another random variable related to the minimum distance $d(\mathcal{C})$ and the covering radius $\rho(\mathcal{C})$: Consider the ball $\mathcal{B}_q(n, \omega)$ and define

$$N_\omega := N\left(\mathcal{B}_q(n, \omega)\right) \tag{2.22}$$

as the number of codewords of weight $\leq \omega$. Clearly, N_ω is monotone increasing in ω and one expects N_ω to become exponentially large in n as soon as ω exceeds a particular value. Depending on the code rate R, this particular value is exactly given by the *relative Gilbert-Varshamov distance* as defined next.

Definition 2.4.8 (Relative GV distance)**.** Let $0 < R < 1$. The **relative Gilbert-Varshamov distance** $\mathrm{D}_{\mathsf{GV}}(R) \in \mathbb{R}$ is the *unique* solution in $0 \leq x \leq 1 - \frac{1}{q}$ of the equation

$$\mathrm{H}_q(x) = 1 - R \ . \tag{2.23}$$

Combining Lemma 2.4.6 with the tail inequalities presented in Section 2.2 allows to prove the following important theorem.

Theorem 2.4.9. *Almost all linear codes meet the relative Gilbert-Varshamov distance, i.e. for almost all linear codes $\mathcal{C}$ of rate R it holds*

$$d(\mathcal{C}) \geq \lfloor \mathrm{D}_{\mathsf{GV}}(R)n \rfloor \ .$$

Proof. We consider the family $\{\mathcal{C}_n\}_{n\in\mathbb{N}}$ of random linear codes with fixed rate R, i.e. $\lim_{n\to\infty}\frac{k(n)}{n} = R$. Let $\varepsilon > 0$ and set $\omega := \lfloor (\mathrm{D}_{\mathsf{GV}}(R) - \varepsilon)n \rfloor$. Due to Lemma 2.3.5 we know $\mathrm{vol}_q(n,\omega) = q^{\mathrm{H}_q(\omega/n)n+\mathrm{o}(n)}$. By definition of $\mathrm{D}_{\mathsf{GV}}(R)$ and by monotonicity of H_q it holds

$$\mathrm{H}_q\left(\frac{\omega}{n}\right) \leq \mathrm{H}_q(\mathrm{D}_{\mathsf{GV}}(R) - \varepsilon) = 1 - R - \varepsilon'$$

for some $\varepsilon' > 0$. We can now apply Lemma 2.4.6 and obtain

$$\mathbb{E}[N_\omega] = q^{n(\mathrm{H}_q(\omega/n)-(1-R))+\mathrm{o}(n)} = q^{-\varepsilon' n+\mathrm{o}(n)} \quad . \tag{2.24}$$

Now the Markov inequality implies $\mathbf{Pr}[N_\omega \geq 1] \leq \mathbb{E}[N_\omega]$ which decreases exponentially in n for every fixed $\varepsilon > 0$. In other words, the fraction of codes with minimum distance $D \leq \mathrm{D}_{\mathsf{GV}}(R) - \varepsilon$ tends to zero. □

Note that the proof of Theorem 2.4.9 also works for $\omega > \lfloor \mathrm{D}_{\mathsf{GV}}(R)n \rfloor$ in which case Eq.(2.24) becomes $\mathbb{E}[N_w] = q^{\varepsilon n+\mathrm{o}(n)}$ for some $\varepsilon > 0$. The Chebychev inequality then guarantees

$$\mathbf{Pr}[|N_\omega - \mathbb{E}[N_w]| \geq 2\mathbb{E}[N_w]] \leq \frac{q-1}{4} q^{-\varepsilon n} \quad . \tag{2.25}$$

We thus see that N_ω is tightly concentrated around $\mathbb{E}[N_\omega]$ for almost all codes. Observe that for very large $\omega \geq \left(1 - \frac{1}{q}\right)n$, Lemma 2.3.5 can no longer be applied and we do only have the trivial estimate $\mathbb{E}[N_w] = q^{nR-\mathrm{o}(n)}$, i.e. almost all codewords are contained in $\mathcal{B}_q(n,\omega)$. We summarize these findings in the following corollary.

Corollary 2.4.10. *For almost all linear codes of rate R it holds*

$$N_w = \begin{cases} q^{n(\mathrm{H}_q(\omega/n)-(1-R))+\mathrm{o}(n)}, & \mathrm{D}_{\mathsf{GV}}(R)n < \omega < \left(1 - \frac{1}{q}\right)n \\ q^{nR-\mathrm{o}(n)}, & \left(1 - \frac{1}{q}\right)n \leq \omega \leq n \quad . \end{cases} \tag{2.26}$$

In summary, $N_\omega = 0$ for all $\omega < \mathrm{D}_{\mathsf{GV}}(R)n$ but $N_\omega = 2^{\mathcal{O}(n)}$ for all $\omega > \mathrm{D}_{\mathsf{GV}}(R)n$ with high probability and for sufficiently large n. A similar but refined analysis allows to tightly estimate the covering radius $\rho(\mathcal{C})$ as stated in the next theorem whose proof is omitted.

Theorem 2.4.11 (Theorem 3.4 in [Bar98])**.** *For almost all linear codes of rate R it holds $\rho(\mathcal{C}) = \mathrm{D}_{\mathsf{GV}}(R)n + \mathrm{o}(n)$.*

3 Code-based Cryptography

As mentioned in the introduction, there is a dire need to construct candidate (trapdoor) one-way functions based on assumptions other than the classical number theoretic ones. The aim of this chapter is to explain how this can be achieved by using hard problems from coding theory. Namely, in Section 3.1, we will introduce the *computational syndrome decoding problem* (CSD for short), which is the fundamental problem in code-based cryptography, and we will show two classical constructions of (trapdoor) one-way functions based on this problem in Section 3.2. For pure one-way functions (i.e. no trapdoors), one can use random linear codes, i.e. codes without any particular mathematical structure. For those codes, the most efficient algorithms for the CSD problem belong to the class of *information set decoding* algorithms whose study is the main topic of this thesis. For *trapdoor* one-way functions, one has to use particularly structured codes that allow for efficient decoding. One then hopes that the public description of the corresponding one-way function (which can be seen as a randomly transformed version of the structured code) looks like a random code. If this is the case, the problem of inverting the function becomes nearly as difficult as in the purely random setting. This highly motivates the primary goal of this thesis:

Improving the efficiency of algorithms for the problem of decoding random linear codes.

3.1 The Computational Syndrome Decoding Problem

The computational syndrome decoding problem is the standard computational problem in the code-based cryptography literature. The security of essentially all code-based cryptographic constructions relies on the intractability of the CSD problem for random linear codes. Besides giving a formal definition of the CSD problem we will discuss

- basic useful properties,
- the significance of the parameter ω and
- the relation to the minimum distance decoding (MDD) problem.

Definition 3.1.1 (CSD problem)**.** The **computational syndrome decoding problem (CSD)** is defined as follows: Given a parity check matrix $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$, a syndrome $\mathbf{s} \in \mathbb{F}_q^{n-k}$ and a target weight $\omega \in \mathbb{N}$, find a vector $\mathbf{e} \in \mathbb{F}_q^n$ with $\mathbf{He} = \mathbf{s}$ and $\mathrm{wt}(\mathbf{e}) \leq \omega$ (if such $\mathbf{e}$ exists). We denote a particular instance of this problem by $\mathrm{CSD}_\omega(\mathbf{H}, \mathbf{s})$.

The running time of an algorithm for the CSD problem clearly depends on the parameters n,k,ω and q which we denote it by $T(n, k, \omega, q)$. Of course, the notion of *running time* has to be defined with respect to some computational model: We assume that the field operations $+, \times$ as well as the inversion of field elements can be carried out in *unit time* and we define the running time of an algorithm to be the *number of field operations* it executes. This is a reasonable model when q is bounded by a polynomial in n: The simplest way to implement the field operations (in polynomial time in n) is to use look-up tables of polynomial size and these polynomial factors will disappear in our analysis in any case as we will discuss next. Moreover, most of the algorithms presented in this work will be probabilistic and we are interested in their *average-case* performance. That is, the actual running time of these algorithm is a random variable defined over the random choice of the input instance and the internal random coins of the algorithm. We refer to Remark 4.1.3 for a detailed discussion about how to analyse these algorithms.

For an appropriate choice of the parameters, the CSD problem for random linear codes is hard and all generic algorithms will have exponential running time in n. One usually compares the performance of generic decoding algorithms for long codes of rate $0 < R < 1$ where one additionally parametrizes $\omega = \lfloor Wn \rfloor$ for some $W > 0$. In this setting, the standard metric to compare such algorithms is given by the so-called *complexity coefficient* $F(R, W, q)$ as introduced in [CG90], i.e.

$$F(R, W, q) := \lim_{n\to\infty} \frac{1}{n} \log_q T(n, \lfloor Rn \rfloor, \lfloor Wn \rfloor, q) \ , \tag{3.1}$$

see also Chapter 7 for a short critical discussion for the case $q > 2$. As mentioned above, this metric ignores polynomial factors $p(n)$ as $\lim \frac{1}{n} \log p(n) = 0$. Throughout this thesis, we will call two decoding algorithms with equal complexity coefficients **(asymptotically) equivalent**. Some general remarks about the CSD problem are in order.

Remark 3.1.2.

1. The above definition does *not* require to find a *minimal weight solution*, every solution of weight $\leq \omega$ is admissible.

2. Compared to other computational problems used in cryptography (e.g. the *Diffie-Hellman problem*) the CSD problem is equivalent to its decisional variant: Let $\mathcal{O}(\mathbf{H}, \mathbf{s}, \omega)$ be an oracle that tells whether a solution $\mathbf{e}$ with $\mathbf{He} = \mathbf{s}$ and $\mathrm{wt}(\mathbf{e}) \leq \omega$ exists or not. To determine e_1, one simply invokes $\mathcal{O}$ on $\mathbf{H}' := \mathbf{H}_{[2,n]}$ (i.e. the

matrix consisting of all but the first column of $\mathbf{H}$), $\mathbf{s}' := \mathbf{s} - e_1\mathbf{h}_1$ (where $\mathbf{h}_1$ is the first column of $\mathbf{H}$) and $\omega' := \omega - 1$ for all $e_1 = 1, \ldots, q-1$ (recall that q is constant). If $\mathcal{O}$ gives a positive answer for some $e_1 \in \mathbb{F}_q^*$, we fix e_1 to that value and proceed with $\mathbf{H}'$, $\mathbf{s}'$ and ω'. Otherwise, we set $e_1 := 0$ and continue with $\mathbf{H}'$, $\mathbf{s}$ and ω.

The next lemma states a very useful property of the CSD problem, that is one can always perform certain linear algebra transformations without changing the problem. Specifically one can

- permute the columns of $\mathbf{H}$ (and the coordinates of $\mathbf{e}$) or
- apply elementary row operations on $\mathbf{H}$ (and the syndrome $\mathbf{s}$).

Lemma 3.1.3. *Let $\mathbf{P} \in \mathbb{F}_q^{n\times n}$ be a permutation matrix and $\mathbf{T} \in \mathbb{F}_q^{(n-k)\times(n-k)}$ invertible. Then*

$$\mathbf{e} \text{ solves } \mathrm{CSD}(\mathbf{H}, \mathbf{s}, \omega) \quad \text{if and only if} \quad \tilde{\mathbf{e}} \text{ solves } \mathrm{CSD}(\tilde{\mathbf{H}}, \tilde{\mathbf{s}}, \omega)$$

where $\tilde{\mathbf{e}} = \mathbf{P}^{-1}\mathbf{e}$, $\tilde{\mathbf{H}} = \mathbf{THP}$ and $\tilde{\mathbf{s}} = \mathbf{Ts}$.

Proof. Clearly, $\mathrm{wt}(\mathbf{e}) = \mathrm{wt}(\tilde{\mathbf{e}})$ and

$$\mathbf{He} = \mathbf{s} \Leftrightarrow \mathbf{THPP}^{-1}\mathbf{e} = \mathbf{Ts} \Leftrightarrow \tilde{\mathbf{H}}\tilde{\mathbf{e}} = \tilde{\mathbf{s}} \ .$$

□

On the one hand, this property enables us to transform $\mathbf{H}$ into standard form. On the other hand, it allows to *rerandomize* the error positions of a CSD instance. Those properties, as simple as the may be, lie at the heart of all ISD algorithms.

The Parameter ω

Let us now discuss the hardness of the CSD problem for long codes of rate R depending on the choice of ω. Clearly, the problem is easy if

- ω is constant or
- ω is very large, namely for $\omega > \left(1 - \frac{1}{q}\right) n$ a randomly chosen preimage of $\mathbf{s}$ has weight $\leq \omega$ with good probability.

In the asymptotic setting for long codes of rate R and $\omega = \lfloor Wn \rfloor$, it is often heuristically assumed that the CSD problem becomes hardest for fixed $W = \mathrm{D}_{\mathsf{GV}}(R)$. This heuristic is justified as follows: Let us assume that at least one solution always exists (this assumption is always true in cryptographic settings). For $W < \mathrm{D}_{\mathsf{GV}}(R)$ the proof of Theorem 2.4.9 implies *uniqueness* of the solution with high probability whereas the search space $B_q(n, \omega)$ clearly grows proportionally with ω (and exponentially in n). On

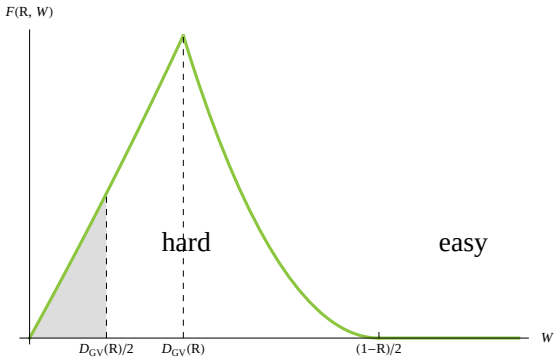


Figure 3.1: Complexity of ISD algorithms for varying $0 < W < \frac{1}{2}$ (binary case). The gray region covers instances where W lies below the error capability of the code.

the other hand, Corollary 2.4.10 implies that there is an exponential number of solutions for $W > \mathrm{D}_{\mathsf{GV}}(R)$ which renders the problem easier. Consequently, the hardest case might be expected for $W = \mathrm{D}_{\mathsf{GV}}(R)$.

To support this argument, we could further consider the behaviour of the best known algorithms for the CSD problem for fixed n and R and varying W. Principally, the best known algorithms fall into the class of *information set decoding* algorithms. As we will see in Chapter 4.1.1, the simplest ISD variant is due to Prange which, for the binary case, has complexity coefficient

$$F(R, W) = \begin{cases} \mathrm{H}(W) - (1-R)\,\mathrm{H}\left(\frac{W}{1-R}\right) , & W \leq \mathrm{D}_{\mathsf{GV}}(R) \\ 1 - R - (1-R)\,\mathrm{H}\left(\frac{W}{1-R}\right) , & \mathrm{D}_{\mathsf{GV}}(R) < W < \frac{1-R}{2} \end{cases} . \tag{3.2}$$

Note that this formula also distinguishes the cases "one solution" vs. "many solutions". As we will shortly see, all ISD algorithms will repeatedly try to recover error vectors $\mathbf{e}$ with a particular *weight distribution* (e.g. $\mathbf{e}$ must have Hamming weight 0 on its first k coordinates for Prange's algorithm). For a fixed solution $\mathbf{e}$, the success probability of a single iteration of such an algorithm can easily be computed (e.g. the first case in Eq.(3.2) exactly represents the expected number of iterations until the unique solution is found). If there are (exponentially) many solutions, the expected number of iterations can be divided by the number of solutions. In terms of complexity coefficients, the second case of Eq.(3.2) is simply obtained from the first case by subtracting the number of solutions (which is given by $\mathrm{H}(W) - (1-R)$ due to Corollary 2.4.10).

Note that $F(R, W)$ achieves its maximum value at $W = \mathrm{D}_{\mathsf{GV}}(R)$, as illustrated in Figure 3.1, and that the problem becomes easy for $W = \frac{1-R}{2}$. In summary, the CSD problem is most interesting for error weights $0 < W \leq \mathrm{D}_{\mathsf{GV}}(R)$ and we restrict to this range for all asymptotic analysis of ISD algorithms presented in this work. The resulting relation between W and R is illustrated in Figure 3.2.

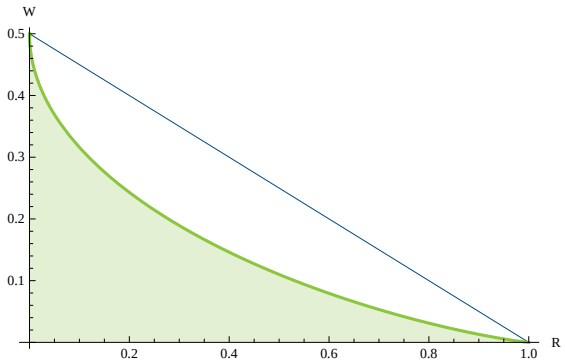


Figure 3.2: Range of error weights W below the relative GV bound in relation to R (shaded area) and upper bound $\frac{1-R}{2}$ (frequently used in proofs of subsequent chapters).

Relation to the MDD problem

We conclude this section by relating the CSD problem to the MDD problem. When studying the complexity of the MDD problem from an asymptotic point of view it is legitimate to restrict to decoding algorithms that can correct up to $\mathrm{D}_{\mathsf{GV}}(R)n$ errors due to Theorem 2.4.11. Let $\mathcal{A}$ be an algorithm for the CSD problem with complexity coefficient $F_{\mathcal{A}}(R, W, q)$. For long codes, Theorem 2.4.11 implies that decoding errors up to weight $\Omega := \lceil (\mathrm{D}_{\mathsf{GV}}(R) + \varepsilon)n \rceil$ is sufficient to solve the MDD problem (for any $\varepsilon > 0$). Thus, repeatedly invoking $\mathcal{A}$ at most Ω times for growing values $\omega = 1, 2, \ldots$ allows to find a minimum weight solution for the MDD problem As we have seen before, $\mathcal{A}$'s running time is likely to increase with ω. Consequently the overall complexity coefficient for the resulting MDD algorithm can be upper bounded by $F_{\mathcal{A}}(R, \mathrm{D}_{\mathsf{GV}}(R) + \varepsilon, q)$. Similarly, we can assume w.l.o.g. that the actual weight of the solution to a given CSD instance is always *known*.

3.2 Code-based One-way Functions

For our purposes, a rather informal definition of *one-way functions* (OWF for short) is sufficient. A one-way function is a mapping $f : X \to Y$ with domain X and range Y such that the following two properties are fulfilled:

i) f is efficiently computable (for all $x \in X$).

ii) For almost all $y \in Y$ it is (computationally) infeasible to find a preimage $x \in f^{-1}(y)$.

There are essentially two dual ways of building a one-way function based on the hardness of the CSD problem. For simplicity we only consider the binary case from now on. Recall that $W_{n,\omega}$ is the n-dimensional Hamming sphere of radius ω. The first function, due to McEliece [McE78], is described by a generator matrix and simply works by mapping a codeword $\mathbf{c}$ and an error vector $\mathbf{e}$ to $\mathbf{c} + \mathbf{e}$. The second one, due to Niederreiter [Nie86], is described by a parity check matrix and simply maps error vectors $\mathbf{e}$ to syndromes $\mathbf{He}$, see Figure 3.3.

$$f_{\mathbf{G}}: \begin{array}{ccccc} \mathbb{F}_2^k & \times & W_{n,\omega} & \to & \mathbb{F}_2^n \\ (\ \mathbf{m} & , & \mathbf{e}\ ) & \mapsto & \mathbf{m}^\top\mathbf{G} + \mathbf{e}^\top \end{array} \qquad f_{\mathbf{H}}: \begin{array}{ccc} W_{n,\omega} & \to & \mathbb{F}_2^{n-k} \\ \mathbf{e} & \mapsto & \mathbf{He} \end{array}$$

Figure 3.3: The McEliece OWF (left) and the Niederreiter OWF (right)

Note that when choosing $\mathbf{G}$ or $\mathbf{H}$ at random, the hardness of inverting the resulting OWFs actually requires the *average-case hardness* of the CSD. Clearly both functions are equally hard to invert. If we employ random codes of length n and rate R, the observation made in Section 3.1 suggests fixing $W = \mathrm{D}_{\mathsf{GV}}(R)$ (where $\omega := \lfloor Wn \rfloor$). Note that both functions will be

- injective if $W = \frac{\mathrm{D}_{\mathsf{GV}}(R)-\varepsilon}{2}$ or,
- surjective if $W = \mathrm{D}_{\mathsf{GV}}(R) + \varepsilon$

with high probability for arbitrary $\varepsilon > 0$ and sufficiently large n. Unfortunately, we almost never get bijective functions. For example, the Niederreiter OWF can be used to build the *identification scheme* due to Stern [Ste93]. Besides this, Fischer and Stern [FS96] showed how to construct a *pseudo-random generator* based solely on the onewayness of the Niederreiter OWF. To obtain more powerful tools, one needs to extend the OWFs into *injective trapdoor one-way functions* (T-OWF for short). We will now show how to turn the McEliece OWF into a T-OWF. A similar construction exists for the Niederreiter OWF.

How to Embed a Trapdoor

In addition to an ordinary one-way function, a *trapdoor one-way function* allows for efficient inversion by using some auxiliary *trapdoor information*. That is, one has to efficiently generate a public description of a OWF $f : X \to Y$ together with a secret description of a function f^{-1} such that $f^{-1}(f(x)) = x$ for all $x \in X$.

Consequently, we can no longer use the class of random linear codes in order to construct a T-OWF as we do not know how to generate f^{-1} (an efficient algorithm for the CSD problem, for $f_{\mathbf{G}}$ when $\mathbf{G}$ is a randomly chosen generator matrix). The basic strategy to resolve this problem is simple:

- Take a secret code $\mathcal{C}$ which belongs to a family of codes equipped with an efficient decoding algorithm Decode. The description of the decoding algorithm is publicly known (but it needs the secret *structured* description of $\mathcal{C}$ as input).
- Randomly transform $\mathcal{C}$ into an *equivalent code* $\tilde{\mathcal{C}} = \Psi(\mathcal{C})$, where Ψ is meant to mask the structure of $\mathcal{C}$.

- $\tilde{\mathcal{C}}$ defines the public function f whereas $\mathcal{C}$ and Ψ allows one to efficiently compute f^{-1}.

More concretely, let $\mathcal{C}$ be an $[n, k, d]$-code with a structured generator matrix $\mathbf{G}_{\mathsf{sec}}$ and assume Decode is ω-error correcting given $\mathbf{G}_{\mathsf{sec}}$, i.e. $\mathsf{Decode}_{\mathbf{G}_{\mathsf{sec}}}(\mathbf{m}^\top\mathbf{G}_{\mathsf{sec}} + \mathbf{e}^\top) = \mathbf{m}^\top\mathbf{G}_{\mathsf{sec}}$ for all $\mathbf{e} \in W_{n,\omega}$ and all $\mathbf{m} \in \mathbb{F}_2^k$. To generate an equivalent random-looking code, one randomly permutes the columns of $\mathbf{G}_{\mathsf{sec}}$ and applies a random change of basis, i.e. one sets $\mathbf{G}_{\mathsf{pub}} := \mathbf{T}\mathbf{G}_{\mathsf{sec}}\mathbf{P}$ for a randomly chosen permutation matrix $\mathbf{P} \in \mathbb{F}_2^{n\times n}$ and an invertible transformation matrix $\mathbf{T} \in \mathbb{F}_2^{k\times k}$. Then f is simply the usual McEliece OWF $f_{\mathbf{G}_{\mathsf{pub}}}$ (with error weight ω). The trapdoor, i.e. the secret description of f^{-1}, consists of $\mathbf{G}_{\mathsf{sec}}$ together with the secret transformation matrices $\mathbf{T}$ and $\mathbf{P}$. Given $\mathbf{y} = f_{\mathbf{G}_{\mathsf{pub}}}(\mathbf{m}, \mathbf{e})$, one first computes

$$\tilde{\mathbf{y}} = \mathbf{y}\mathbf{P}^{-1} = (\mathbf{m}^\top\mathbf{T})\mathbf{G}_{\mathsf{sec}} + \mathbf{e}^\top\mathbf{P}^{-1} \ .$$

Since $\mathrm{wt}(\mathbf{e}^\top\mathbf{P}^{-1}) = \mathrm{wt}(\mathbf{e}) = \omega$, it follows $\mathsf{Decode}(\tilde{\mathbf{y}}) = \mathbf{m}^\top\mathbf{T}\mathbf{G}_{\mathsf{sec}}$ which eventually reveals $\mathbf{m}$ and $\mathbf{e}$ and allows to successfully invert f.

In contrast to the pure OWF based on random linear codes, the onewayness of the above construction is no longer *equivalent* to the average-case hardness of the CSD problem for random linear codes. Clearly, an algorithm that solves the CSD probem can still be used to invert f but the converse is no longer true. That is to say an inversion algorithm for f merely solves the CSD problem for a particular class of codes (equivalent to an efficiently decodable class).

Structural Attacks

One way to invert the McEliece T-OWF is to run a so-called *structural attack* which means that one may try to recover $\mathbf{G}_{\mathsf{sec}}$ from $\mathbf{G}_{\mathsf{pub}}$. A necessary condition to avoid such attacks is to choose an *exponentially large* family of codes. The *support splitting algorithm (SSA)* [Sen00] allows to decide whether two linear codes $\mathcal{C}$ and $\tilde{\mathcal{C}}$ are *permutation-equivalent*, i.e. it determines whether the codewords of $\tilde{\mathcal{C}}$ can be obtained by permuting the coordinates of the codewords in $\mathcal{C}$. If so, the algorithm also recovers the permutation. Thus, given $\mathbf{G}_{\mathsf{pub}}$ one could simply guess $\mathbf{G}_{\mathsf{sec}}$ and run the SSA. If one guesses correctly, the SSA outputs $\mathbf{P}$ (which also reveals $\mathbf{T}$ by linear algebra). Furthermore, there are many examples of families of codes that appeared to be vulnerable to specific structural attacks, e.g.

- Generalised Reed-Solomon codes (broken by Sidelnikov and Shestakov in [SS92]),
- Reed-Muller Codes (broken by Minder and Shokrollahi in [MS07]),
- Quasi-cyclic alternant codes and quasi-dyadic Goppa codes (broken by Faugère et al. in [FOPT10]).

Fortunately, the family of general *binary Goppa codes*, as originally proposed by McEliece, resists all known structural attacks. Thus the following assumption is frequently used in the literature (e.g. in [FS01] or [NIKM08]).

Assumption 3.2.1 (Goppa-Code Indistinguishability, informal)**.** A randomly transformed generator matrix $\mathbf{G}_{\mathsf{pub}}$ of a binary $[n, k]$ Goppa code is computationally indistinguishable from a random generator matrix $\mathbf{G} \in \mathbb{F}_2^{k \times n}$.

Under this assumption, one can simply ignore the mathematical structure of the underlying code when attacking the McEliece T-OWF. Consqeuently we do not need to introduce Goppa codes and the central goal of this work can be reformulated as follows.

Find improved inversion algorithms for the McEliece T-OWF.

As an application of our work, we will give refined security estimates for practical concrete parameter sets of the McEliece T-OWF (as proposed in [BLP08]) in Section 8.

n	k	ω	Security Level
1632	1269	34	80 Bit
2960	2288	57	128 Bit
6624	5129	117	256 Bit

Table 3.1: Parameters for the McEliece T-OWF for binary Goppa codes.

Remark 3.2.2.

1. Historically, the McEliece T-OWF was introduced as a *public-key encryption scheme*. However, the function does not fulfil any of the modern security notions for public-key encryption. Clearly, a McEliece "ciphertext" $\mathbf{c} = \mathbf{m}^\top \mathbf{G}_{\mathsf{pub}} + \mathbf{e}^\top$ is *malleable*. Adding arbitrary codewords $\tilde{\mathbf{m}}^\top \mathbf{G}_{\mathsf{pub}}$ to $\mathbf{c}$ gives a new valid ciphertext $\tilde{\mathbf{c}}$ which directly breaks the *CCA-security* of the scheme. Furtheremore, one does not even need a *full* decryption oracle. So-called *reaction attacks* [HGS99] are easily possible: A server that rejects malformed ciphertexts (e.g. ciphertexts with false error weight) can be seen as an oracle for the decisional CSD problem which allows to recover $\mathbf{e}$ (see Remark 3.1.2). Even worse, also the weaker notion of *CPA-security* can be attacked as follows: Given a challenge ciphertext $\mathbf{c} = \mathbf{m}_b^\top \mathbf{G}_{\mathsf{pub}} + \mathbf{e}$ which is the encryption of one of two *known* messages $\mathbf{m}_0$ and $\mathbf{m}_1$, one can determine the actual bit b: Simply add $\mathbf{m}_0^\top \mathbf{G}_{\mathsf{pub}}$ to $\mathbf{c}$. If $b = 0$, the resulting vector will have weight ω. Otherwise, the weight will be much larger .
2. There are several ways to transform the McEliece T-OWF into a real public-key encryption scheme with strong security guarantees, e.g. the Kobara-Imai transformation [KI01] which gives CCA2-security in the *random oracle model* or even *standard model* variants like the proposal of Dowsley et al. [DMQN09] or the recent work of Persichetti [Per12]. Moreover, the weaker notion of CPA-security in the standard model can be achieved very easily: Pad the message $\mathbf{m}$ with a random bit-string of appropriate length [NIKM08]. However, *all* proposals are based on the hardness of inverting the McEliece T-OWF and are thus theoretically vulnerable to the improved attacks presented in this thesis.

4 Information Set Decoding

> "A mathematical theory is not to be considered complete until you have made it so clear that you can explain it to the first man whom you meet on the street."
>
> *Carl Friedrich Gauss*

4.1 Introduction

Information set decoding (ISD for short) can be seen as a class of generic decoding algorithms for the CSD problem for *random* linear codes – or equivalently – for codes without any particular structure. Throughout this chapter (and even the next two) we will only consider the *binary* case and we will show how to generalise all our results to arbitrary q-ary codes in a straightforward way in Chapter 7. Recall that we can assume without loss of generality that the wanted error vector has weight *exactly* ω and we do know the value ω.

Obviously, when designing decoding algorithms for *random* binary linear codes, there is essentially no structure to exploit except

- the vector space characterisation of linear codes,
- the knowledge of the target weight ω of the error vector $\mathbf{e}$.

Essentially, knowledge of ω merely allows to define the brute force search space as the set $W_{n,\omega}$ and to restrict the search to *particular* error patterns that occur with a certain probability (which can be easily computed). Exploiting the vector space structure essentially corresponds to applying elementary transformations to $\mathbf{H}$ as explained in Lemma 3.1.3, in particular permuting the columns of $\mathbf{H}$ allows one to randomly shuffle the error positions of $\mathbf{e}$. In summary, ISD simply provides a clever way to

Reduce the brute-force search space by linear algebra.

In 1962, Prange introduced the first ISD algorithm [Pra62], which is often called **plain information set decoding**, and thus created the basis for all the subsequent improvements. Plain ISD simply repeatedly permutes the coordinates of $\mathbf{e}$ until all errors finally occur in the last $n-k$ positions only (which is rather restrictive). We will explain Prange's algorithm in full detail in the next section. As a precursor, we shall estimate the *brute-force complexity* of the CSD problem for a long code of rate R and the worst-case weight $W = \mathrm{D}_{\mathsf{GV}}(R)$, see Eq.(2.23). Clearly, this complexity serves as a trivial benchmark for all ISD algorithms. Depending on R, a brute-force algorithm would

- *either* enumerate all error vectors $\mathbf{e}$ of weight ω (which costs $2^{\mathrm{H}(W)n+\mathrm{o}(n)}$ operations, i.e. $2^{(1-R)n+\mathrm{o}(n)}$ operations for $W = \mathrm{D}_{\mathsf{GV}}(R)$)
- *or* search a minimum weight representative of the coset $\mathbf{y}+\mathcal{C}$ for some $\mathbf{y}$ in the preimage of $\mathbf{s}$ (which costs $2^{Rn+\mathrm{o}(n)}$ operations).

Therefore, the resulting complexity coefficient $F_{\mathsf{BF}}(R) := F_{\mathsf{BF}}(R, \mathrm{D}_{\mathsf{GV}}(R))$ is given by

$$F_{\mathsf{BF}}(R) = \begin{cases} R & ,0 \le R \le \frac{1}{2} \\ 1-R & \frac{1}{2} < R \le 1 \end{cases} \tag{4.1}$$

which is illustrated in Figure 4.1. Note that, in contrast to all ISD algorithms, the above brute-force algorithm is *deterministic* and will always output a solution after at most $2^{F_{\mathsf{BF}}(R)n+\mathrm{o}(n)}$ steps.

Remark 4.1.1. In the above description, we ignored the costs of some operations that occur in the brute-force algorithm, e.g. for every error vector $\mathbf{e}$ one would have to compute $\mathbf{He}$ and compare the result to the given syndrome $\mathbf{s}$. Similarly, the costs for computing some $\mathbf{y}$ in the preimage of $\mathbf{s}$ have been neglected. Cleary, both operations can be implemented in time polynomial in n and thus do *not* affect the complexity coefficient at all. In most chapters, we will ignore such costs since we are mainly interested in the asymptotic behaviour of the different ISD algorithms. This will significantly simplify the runtime analysis for the more sophisticated algorithms. For practical applications, it is actually important to take all these costs into account and we will give a refined analysis of our new algorithm in Section 8.

Roadmap

The rest of this chapter is organised as follows: As already stated, we begin with a description of Prange's plain ISD in the next section. In Section 4.1.2 we will use plain ISD as an example to treat the technical issue of *defective permutations* which slightly complicates a rigorous runtime analysis of ISD algorithms. After the introductory section we will sketch some alternative approaches to design generic decoding algorithms in Section 4.2. Finally, we extend plain ISD to less restrictive error patterns in Section 4.3. This generalised description will cover many known ISD algorithms like *Stern's ISD algorithm* [Ste89] or the recently proposed *Ball-collision decoding* [BLP11a] which was considered to be the fastest ISD algorithm.

4.1.1 Prange's Algorithm - Plain Information Set Decoding

The main idea of plain ISD is fairly simple:

- Compute the standard form $\tilde{\mathbf{H}}$ of a randomly column-permuted version of $\mathbf{H}$.
- Hope that *no errors* occur in the first k coordinates (in that case the syndrome $\tilde{\mathbf{s}}$ will reveal the permuted version $\tilde{\mathbf{e}}$ of $\mathbf{e}$).

Thus, repeatedly computing such randomised standard forms until a solution is found yields a a simple *randomised Las-Vegas algorithm* for the CSD problem.

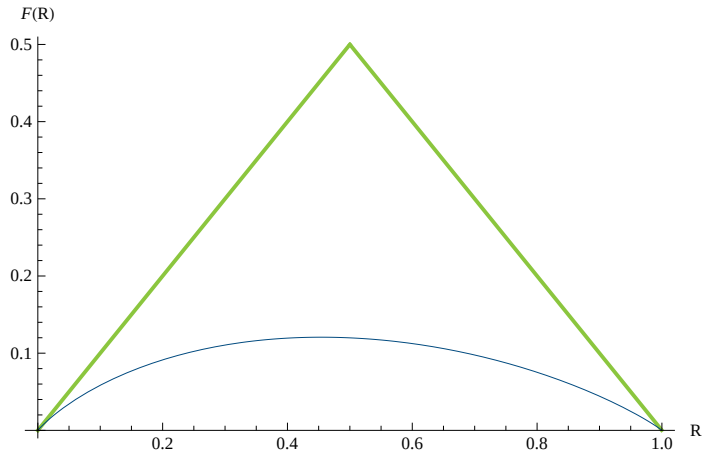


Figure 4.1: Brute-force complexity coefficients F_{BF} (thick line) and F_{Pra} for Prange's algorithm (thin curve) where $W = \mathrm{D}_{\mathsf{GV}}(R)$ (complete decoding).

More precisely, we first compute

$$\tilde{\mathbf{H}} := \mathbf{THP} = \begin{pmatrix} \mathbf{Q} & \mathbf{I}_{n-k} \end{pmatrix} \tag{4.2}$$

for a non-singular $k \times k$-matrix $\mathbf{T}$ and a random, $n \times n$ permutation matrix $\mathbf{P}$ (and similarly consider $\tilde{\mathbf{s}} := \mathbf{Ts}$ according to Lemma 3.1.3). Let us now assume that the corresponding permuted error vector $\tilde{\mathbf{e}} := \mathbf{P}^{-1}\mathbf{e}$ exclusively contains errors in the last $n-k$ coordinates. Since $\tilde{\mathbf{H}}$ is in standard form, i.e. $\tilde{\mathbf{H}} = \begin{pmatrix} \mathbf{Q} & \mathbf{I}_{n-k} \end{pmatrix}$, it follows

$$\tilde{\mathbf{s}} = \tilde{\mathbf{H}}\tilde{\mathbf{e}} = \tilde{\mathbf{e}}_{[k+1,n]} \ . \tag{4.3}$$

Thus $\tilde{\mathbf{s}}$ will reveal $\tilde{\mathbf{e}} = (\mathbf{0}, \tilde{\mathbf{s}})$ (where $\mathbf{0}$ denotes the zero vector of length k) whenever we picked a *good* permutation $\mathbf{P}$, where a good permutation is one that maps all of $\mathbf{e}$'s non-zero positions to the last $n-k$ coordinates. Moreover, it is easy to determine whether we picked a good permutation by simply checking if $\tilde{\mathbf{s}}$ has the correct weight, i.e. if $\mathrm{wt}(\tilde{\mathbf{s}}) = \omega$. The overall algorithm is summarised in Algorithm 1 (where Σ_n denotes the *symmetric group* of n elements and $x \leftarrow_R X$ means "assign x to a random value from X"). Since the initial transformation $\tilde{\mathbf{H}} = \mathbf{THP}$ will be reused in later algorithms, we denote the first three lines of the inner loop of Algorithm 1 by $(\tilde{\mathbf{H}}, \tilde{\mathbf{s}}, \mathbf{P}) \leftarrow \textsc{Init}(\mathbf{H}, \mathbf{s})$.

Algorithm 1: Plain-ISD

input : Parity check matrix $\mathbf{H}$, syndrome $\mathbf{s}$ and target weight ω.
output: Error vector $\mathbf{e}$ with $\mathbf{He} = \mathbf{s}$ and $\mathrm{wt}(\mathbf{e}) = \omega$.

```
found ← false;
while not found do
    P ←_R Σ_n;
    H̃ ← THP = (Q  I_{n−k});
    s̃ ← Ts;
    if wt(s̃) = ω then
        ẽ ← (0‖s̃);
        found ← true;
return Pẽ;
```

Heuristically, the running time of Algorithm 1 can be estimated as follows: The probability of picking a *good* permutation is given by

$$\frac{\binom{n-k}{\omega}}{\binom{n}{\omega}} \ . \tag{4.4}$$

Note that $\mathbf{P}^{-1}\mathbf{e}$ is a random word of length n and weight ω and amongst those, there are exactly $\binom{n-k}{\omega}$ *good* cases. Intuitively, the inverse of this probability, which we will denote by

$$N_{\mathsf{Pra}}(n,k,\omega) := \binom{n}{\omega}\binom{n-k}{w}^{-1} \ , \tag{4.5}$$

will give a good estimate for the necessary number of iterations, i.e. the number of guessed permutations $\mathbf{P}$ until a good one is found. Thus, for long codes of rate R and $W \le \mathrm{D}_{\mathsf{GV}}(R)$ (unique solution), one can estimate the complexity coefficient as

$$\begin{aligned} F_{\mathsf{Pra}}(R,W) &= \lim_{n\to\infty} \frac{1}{n} \log N_{\mathsf{Pra}}(n, \lfloor Rn \rfloor, \lfloor Wn \rfloor) \\ &= \mathrm{H}(W) - (1-R)\,\mathrm{H}\left(\frac{W}{1-R}\right) \ , \end{aligned} \tag{4.6}$$

which coincides with Eq.(3.2).

The main technical problem with this heuristic analysis stems from the fact that we completely ignored the issue of a *failed standard form computations*. It will happen (even with constant probability) that a random permutation $\mathbf{P}$ yields a column-permuted parity-check matrix that contains a *singular* submatrix in its last $n-k$ columns. Our next goal is to prove a rigorous theorem as stated next.

Theorem 4.1.2 (Plain ISD). *Let $0 < R < 1$ and $0 < W \leq \mathrm{D}_{\mathsf{GV}}(R)$. For almost all binary $[n, \lfloor Rn \rfloor]$-codes it holds: Plain information set decoding successfully terminates after $2^{F_{\mathsf{Pra}}(R,W)n+\mathrm{o}(n)}$ iterations for every error vector $\mathbf{e} \in W_{n,\lfloor Wn \rfloor}$ with overwhelming probability.*

Remark 4.1.3. Let us clarify the formulation of Theorem 4.1.2: In general, Plain ISD (and all subsequent ISD algorithm) are *Las Vegas algorithms*. A Las Vegas algorithm is a randomised algorithms that *never errs* and there are normally two ways of formalising their definition: On the one hand, one may require that the algorithm *always outputs a correct solution if it terminates*. In this case, the running time becomes a random variable (over the algorithm's randomness) and one naturally restricts to algorithms with *finite expected running time*. On the other hand, one may prefer to define an algorithm with *bounded worst-case running time*, e.g. by simply aborting computation after a predefined number of steps and by outputting a *failure message*. In this case, the quality of the algorithm is additionally measured by its *failure probability* (to avoid confusion we point out that the term *failure* does not mean that the algorithm outputs a *false* solution). With this in mind, the above theorem (and all subsequent theorems) shall be understood as follows: First of all, we always need to exclude a particular (negligibly small) set of "bad" input matrices. For those input matrices, we are not able to prove rigorous bounds on the algorithm's running time. In the case of Plain ISD, those input matrices correspond to matrices containing submatrices of unnaturally high co-rank, see Section 4.1.2. Secondly, when given a "good" input matrix and arbitrary error vector $\mathbf{e}$, we always give a worst-case bound on the algorithm's running time and bound the failure probability of the algorithm by some function that decreases exponentially in n. Note that the pseudo code presentation of all algorithms in this thesis does not properly cover the second point: To avoid an additional parameter that upper bounds the allowed number of iterations, we stick to formulating all algorithms using **while**-loops. We point out that one could easily fix this problem by replacing the **while**-loop by a **for**-loop that is executed at most $\mathcal{O}(n \cdot N_A)$ times where N_A denotes expected number of iterations according to the probability of guessing a good permutation, compare Eq.(4.5) and Lemma 4.1.4 for Plain ISD.

Let us now clarify the issue of "bad" input matrices. Imagine the following (very unlikely) worst-case scenario: Assume that $\mathbf{H}$ contains ω linearly dependent columns. The error vector whose error positions exactly match those columns will *never* be found by Algorithm 1: *Any* good permutation yields a singular submatrix in $\mathbf{H}$ and the standard form computation will always fail. We will show a simple solution (inspired by [CG90]) for plain ISD in the next section. Since all subsequent ISD algorithms will suffer from the same problem, we will implicitly assume that a similar modification could easily be carried out for the respective algorithms.

4.1.2 Dealing with Defective Permutations

For a fixed generator matrix $\mathbf{H}$, we call a permutation $\mathbf{P}$ **defective** if the submatrix $(\mathbf{HP})_{[k+1,n]}$ has rank $< n-k$. Our aim is to provide a way that still allows to carry out a standard form computation for those defective permutations and to search for the vector $\tilde{\mathbf{e}} = \mathbf{P}^{-1}\mathbf{e}$ in time comparable to Algorithm 1. Once such a procedure is found, the inner loop of plain ISD will actually work for all *permutations*. In this setting, the number of iterations can indeed be estimated by $N_{\mathsf{Pra}}(n,k,\omega)$ as stated in the next lemma.

Lemma 4.1.4. *Let $\mathbf{e} \in W_{n,\omega}$ and define $f(n,k,\omega) := N_{\mathsf{Pra}}(n,k,\omega)\cdot n$. Let $\mathcal{P} \subset \Sigma_n$ denote a set of $f(n,k,\omega)$ randomly chosen permutations. Then there is a good permutation $\mathbf{P} \in \mathcal{P}$ (w.r.t. to $\mathbf{e}$) with overwhelming probability.*

Proof. The proof is straightforward. Define random variables $X_i = 1$ iff the i-th permutation is good (w.r.t. $\mathbf{e}$). Clearly, $\mathbf{Pr}[X_i = 1] = N_{\mathsf{Pra}}(n,k,\omega)^{-1}$. Now define $X := \sum_{i=1}^{f(n,k,\omega)} X_i$. It holds

$$\mathbf{Pr}[X = 0] = \left(1 - N_{\mathsf{Pra}}^{-1}(n,k,\omega)\right)^{f(n,k,\omega)} \leq e^{-n}$$

using $(1-\frac{1}{n})^n \leq e^{-1}$ for all $n \in \mathbb{N}$. Thus there will be at least one good $\mathbf{P}$ in $\mathcal{P}$ with probability $\geq 1 - e^{-n}$. □

The following procedure deals with a defective permutation $\mathbf{P}$. Let $\mathbf{H}' := (\mathbf{HP})_{[k+1,n]}$ denote the corresponding singular submatrix for a fixed parity check matrix $\mathbf{H}$.

- Add the next $\ell := \ell(n,k)$ columns to $\mathbf{H}'$ until $\mathrm{rank}(\mathbf{H}') = n-k$.
- Let $I \subset [k-\ell+1, n]$ denote a set of $n-k$ linearly independent indices and $J := [k-\ell+1, n] \setminus I$ its complement. Transform the whole matrix $\mathbf{HP}$ such that the submatrix $(\mathbf{HP})_I$ is the identity matrix (and let $\mathbf{T}$ denote the corresponding transformation matrix).
- Enumerate all 2^ℓ error patterns corresponding to indices in J: For every $E \subset J$ compute $\mathbf{s}' := \sum_{i\in E}(\mathbf{THP})_i$ and check whether $\mathrm{wt}(\mathbf{s}'+\tilde{\mathbf{s}}) = \omega - |E|$ (where $\tilde{\mathbf{s}} = \mathbf{Ts}$ according to the second step). If the weight check succeeds, we set

$$\tilde{e}_i := 1 \text{ for all } i \in E \ ,$$
$$\tilde{e}_{i_k} := 1 \text{ for all } k \text{ with } (\mathbf{s}+\mathbf{s}')_k = 1$$

  where $I = \{i_1, \ldots, i_{n-k}\}$. Finally, we output $\mathbf{e} := \mathbf{P}\tilde{\mathbf{e}}$ as before.

Figure 4.2: Dealing with defective permuations.

The above procedure outputs a correct solution, since

$$\mathbf{THP}\tilde{\mathbf{e}} = \sum_{i \in E} (\mathbf{THP})_i + \sum_{\substack{i_k \\ (\tilde{\mathbf{s}}+\mathbf{s}')_k = 1}} (\mathbf{THP})_{i_k} = \mathbf{s}' + (\tilde{\mathbf{s}} + \mathbf{s}') = \tilde{\mathbf{s}} = \mathbf{Ts} \ .$$

We used the fact $(\mathbf{THP})_{i_k}$ is the k-th unit vector of length $n-k$.

To show that this procedure does not affect the asymptotic running time, it is sufficient to show that $2^{\ell(n,k)}$ grows subexponential in n, i.e. the corank of $(\mathbf{HP})_I$ has to be small. This is stated in the following lemma, whose proof is omitted.

Lemma 4.1.5 ([BKvT99, Lemma 3])**.** *For almost all a random, binary* $(n-k) \times n$ *matrices* $\mathbf{H}$ *it holds*

$$\operatorname{corank}(\mathbf{H}_I) \leq \sqrt{\log \binom{n}{n-k}}$$

for every $I \subset [n]$ *of size* $|I| = n-k$.

Thus for almost all codes we have $\ell(n,k) \leq \sqrt{\log \binom{n}{n-k}}$, i.e. $2^{\ell(n,k)}$ is indeed subexponential in n. Combining Lemma 4.1.4 and Lemma 4.1.5 eventually proves Theorem 4.1.2.

4.2 Alternative Approaches

There are four main concepts apart from ISD for designing generic decoding algorithms, which we will describe very concisely. The first two approaches both employ precomputed *test sets* which depend on the input code. In many cases, the precomputation of the test set is more expensive than a direct application of ISD algorithms. Nevertheless, once a test set has been created, decoding can become very fast (and thus the algorithms might be useful in settings where one has to decode the same code *repeatedly*). In the following denote by $\mathbf{x} = \mathbf{c} + \mathbf{e}$ a received word with $\mathrm{wt}(\mathbf{e}) = \omega$.

Statistical Decoding (SD). Let $\mathbf{h}$ be a parity check equation. The crucial observation, made by Al Jabri in [AlJ01], is as follows: Assume $\langle \mathbf{h}, \mathbf{x} \rangle = 1$. Since $\langle \mathbf{h}, \mathbf{c} \rangle = 0$, the non-zero positions of $\mathbf{h}$ reveal some information about the error positions (i.e. an odd number of error positions coincides with non-zero entries of $\mathbf{h}$). Collecting this information for many different $\mathbf{h}$'s may allow to recover $\mathbf{e}$. In contrary to ISD algorithms, statistical decoding gives a *Monte-Carlo algorithm* that might output false solutions with a certain probability. This probability heavily depends on choice of the test set $\mathcal{H} = \{\mathbf{h}_1, \mathbf{h}_2, \ldots\}$. Generally,

- $\mathcal{H}$ should provide a particular structure that allows to estimate the statistical properties of the test,
- $\mathcal{H}$ should be large enough to guarantee sufficient reliability of the statistical test.

Al Jabri proposed using $\mathcal{H}_\Omega := W_{n,\Omega}$. That is one uses a large number of parity check equations of fixed weight Ω for the statistical test. Clearly, computing such a test set requires to compute codewords of fixed weight in the dual code (i.e. the code generated by the parity check matrix $\mathbf{H}$), which is a computationally hard problem. In contrary to ISD algorithms, there exists no proper asymptotic analysis of SD and it remains open how to choose Ω optimally and how to compute $\mathcal{H}_\Omega$ efficiently. In [Ove06], Overbeck presents a concrete analysis for some McEliece parameters and concludes that there is no choice of Ω that achieves both correct decoding with good probability *and* competitive performance.

Gradient Decoding (GD). In [AB98], Ashikhmin and Barg proposed the following general framework that is based on a test set $\mathcal{H} = \{\mathbf{c}_1, \mathbf{c}_2, \ldots\}$ of codewords $\mathbf{c}_i$ with the following property: For every word $\mathbf{x}$

- there is either a codeword $\mathbf{c} \in \mathcal{H}$ with $\mathrm{wt}(\mathbf{x}+\mathbf{c}) < \mathrm{wt}(\mathbf{x})$
- or $\mathbf{x}$ lies in the *Voronoi region* $\mathcal{V}(\mathbf{0})$, i.e. the closest codeword to $\mathbf{x}$ is $\mathbf{0}$.

Clearly, given such a test set $\mathcal{H}$ it is easy to design a decoding algorithm: Successively update $\mathbf{x}$ by adding codewords from $\mathcal{H}$ as long as possible. Let us assume that the procedure terminates after m steps, then $\mathbf{x}+\mathbf{c}_{i_1}+\ldots+\mathbf{c}_{i_m} \in \mathcal{V}(\mathbf{0})$ and thus $\mathbf{c}_{i_1}+\ldots+\mathbf{c}_{i_m}$ is the closest codeword to $\mathbf{x}$. The running time of this method is given by $\mathcal{O}(n^2|\mathcal{H}|)$: Every step cancels at least one additional coordinate and requires at most $|\mathcal{H}|$ Hamming weight computations. Consequently, finding good test sets $\mathcal{H}$, i.e. those that are both small and efficiently computable, is the crucial point when implementing this approach. There are essentially two concrete instantiations of gradient decoding, namely:

- *Minimal vectors decoding [Hwa79].* A minimal vector is a codeword $\mathbf{c}$ whose support can not be *properly* covered by any other codeword, i.e. there is no $\tilde{\mathbf{c}}$ with $\mathrm{supp}(\tilde{\mathbf{c}}) \subsetneq \mathrm{supp}(\mathbf{c})$. The set of all minimal vectors forms an appropriate test set. According to [AB98], for long codes, minimal vectors decoding is not considered to improve over the naive brute-force complexity.
- *Zero neighbours decoding [LH85].* Loosely speaking, a *zero neighbour* is a codeword $\mathbf{c}$ whose Voronoi region $\mathcal{V}(\mathbf{c})$ is adjacent to $\mathcal{V}(\mathbf{0})$. Given a word $\mathbf{x} \notin \mathcal{V}(\mathbf{0})$, consider a sequence of words $\mathbf{x}_i$ with $\mathbf{x}_0 := \mathbf{x}$ and $\mathrm{wt}(\mathbf{x}_i) = \mathrm{wt}(\mathbf{x}_{i-1}) - 1$. For some index j we will have $\mathbf{x}_{j+1} \in \mathcal{V}(\mathbf{0})$ and $\mathbf{x}_j \in \mathcal{V}(\mathbf{c}) \setminus \mathcal{V}(\mathbf{0})$ for a zero neighbour $\mathbf{c}$. It follows

  $$\mathrm{wt}(\mathbf{x}+\mathbf{c}) = \mathrm{d}(\mathbf{x},\mathbf{c}) \le \mathrm{d}(\mathbf{x},\mathbf{x}_j) + \mathrm{d}(\mathbf{x}_j,\mathbf{c}) < \mathrm{d}(\mathbf{x},\mathbf{x}_j) + \mathrm{d}(\mathbf{x}_j,\mathbf{0}) = \mathrm{wt}(\mathbf{x})$$

  which shows that the set of zero neighbours forms an appropriate test set. It was shown in [LH85], that the complexity coefficient of zero neighbours decoding for fixed $W = \mathrm{D}_{\mathsf{GV}}(R)$ is given by

  $$F_{\mathsf{ZN}}(R) = \begin{cases} R & ,0 \le R \le 1-\mathrm{H}(1/4) \\ \mathrm{H}(2\,\mathrm{D}_{\mathsf{GV}}(R)) - (1-R) & ,1-\mathrm{H}(1/4) < R \le 1 \end{cases} \tag{4.7}$$

  as illustrated in Figure 4.3.

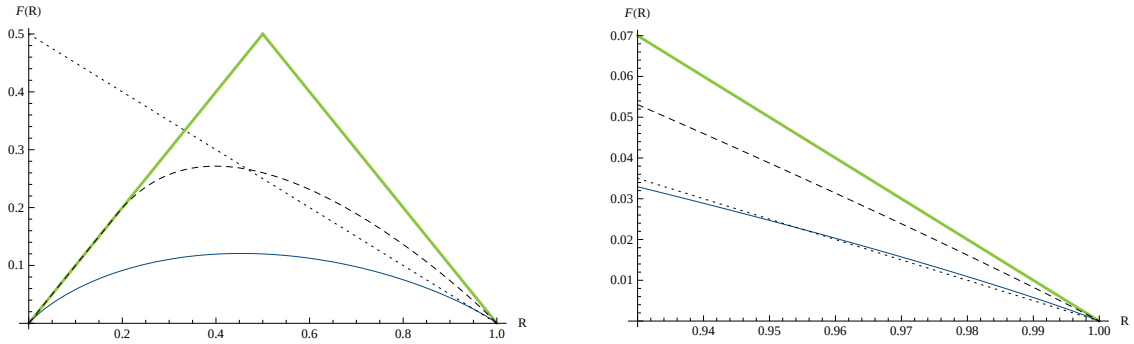


Figure 4.3: Comparison between F_{ZN} (dashed line) and F_{SSD} (dotted line) with brute-force (thick line) and plain ISD (thin line). The right figure shows an enlarged view for high code rates $R \geq 0.93$.

Split Syndrome Decoding (SSD). In [Dum89], Dumer introduced a simple idea that is based on the linearity of the equation $\mathbf{He} = \mathbf{s}$: Decompose $\mathbf{H}$ and $\mathbf{e}$ into two parts $\left(\mathbf{H}_1 \| \mathbf{H}_2\right)$ and $\left(\mathbf{e}_1 \| \mathbf{e}_2\right)$, respectively. Then split the equation into $\mathbf{H}_1\mathbf{e}_1 = \mathbf{H}_2\mathbf{e}_2 + \mathbf{s}$ and build a list of all vectors $\mathbf{e}_1$ of a certain weight ω_1 and store the corresponding $\mathbf{H}_1\mathbf{e}_1$ values. Do the same for $\mathbf{e}_2$ and eventually search for collisions in those lists. As we will see in Section 4.3, a similar idea is used in improved ISD algorithms. Since SSD does not transform $\mathbf{H}$ into standard form, it performs less efficient than those algorithms. Intuitively, balancing the size of both lists gives the best performance and allows to reduce the complexity of naive syndrome decoding by a square-root factor. This was rigorously confirmed in [BKvT99, Theorem 6] which gives the corresponding complexity coefficient as

$$F_{\mathsf{SSD}}(R) = \frac{1-R}{2} \quad . \tag{4.8}$$

Observe that for $R \geq 0.954$, SSD beats plain ISD as illustrated in Figure 4.3. This property motivated Dumer to give an improved variant in [Dum91] which is called *punctured split syndrome decoding (PSSD)*. The name derives from the fact, that Dumer proposed to *puncture* a given code $\mathcal{C}$, so that the rate becomes large. For high-rate codes one can gain in complexity by applying SSD. The algorithm is equipped with two parameters α and β, i.e. the size of the punctured code is $\lfloor \beta n \rfloor$ and the error weight that is assigned to these coordinates is $\lfloor \alpha n \rfloor$ (see Section 5.2.2 for a deeper description). For the sake of completeness, we give the corresponding complexity coefficient (cf. [Bar98, Theorem 3.25]):

$$\begin{aligned} F_{\mathsf{PSSD}}(R) = \min_{\alpha,\beta} \max \Bigg\{ & (1-\beta)\left[1 - \mathrm{H}\left(\frac{\mathrm{D}_{\mathsf{GV}}(R) - \alpha}{1-\beta}\right)\right], \\ & 1 - R - \frac{\beta}{2}\mathrm{H}\left(\frac{\alpha}{\beta}\right) - (1-\beta)\,\mathrm{H}\left(\frac{\mathrm{D}_{\mathsf{GV}}(R) - \alpha}{1-\beta}\right)\Bigg\} \end{aligned} \tag{4.9}$$

where the minimisation has to be done under the natural constraints $R < \beta < 1$ and $\max\{0, \mathrm{D}_{\mathsf{GV}}(R) + \beta - 1\} < \alpha < \min\{\mathrm{D}_{\mathsf{GV}}(R), \beta\}$.

A small contribution of this work is to relate recent ISD algorithms to PSSD: Corollary 5.2.17 proves that PSSD is asymptotically at least as efficient as *Ball-collision decoding*, see Section 4.3.

Supercode Decoding (SCD). Barg et al. proposed a different and interesting approach in [BKvT99] that shares some similarities with our new algorithm as presented in Chapters 5 and 6. Similarly to ISD algorithms, it works on the standard form of $\mathbf{H}$ which is decomposed into σ equal-sized slices each containing $\rho := \frac{n-k}{\sigma}$ rows. Note that for every slice, the error vector $\mathbf{e}$ matches the respective slice of the syndrome. Furthermore, let us assume that the error vector has weight ω_1 on its first k coordinates and thus weight $\omega - \omega_1$ on the remaining $n-k$ coordinates. Every slice i of $\mathbf{H}$ is of the form

$$\begin{pmatrix} \mathbf{Q} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{I}_\rho & \mathbf{0} & \dots & \mathbf{0} \end{pmatrix}$$

where $\mathbf{Q}$ is the appropriate $(\rho \times k)$ submatrix of $\mathbf{H}$ an $\mathbf{I}_\rho$ is a ρ-dimensional identity matrix block between coordinates $k + i\rho$ and $k + (i+1)\rho - 1$. According to this decomposition of slice i, we denote by $(\mathbf{e}_1, \mathbf{e}_2)$ the respective decomposition of $\mathbf{e}$, i.e. $\mathbf{e}_2$ contains the ρ coordinates of $\mathbf{e}$ according to the $\mathbf{I}_\rho$ block in the slice. Define $\omega_2 := \left\lceil \frac{\omega - \omega_1}{\sigma - b + 1} \right\rceil$ and observe that there must be at least one selection of b slices such that every $\mathbf{e}_2$ part in each slice contains at most ω_2 many errors (otherwise, there must be at least $\sigma - b + 1$ slices with corresponding $\mathbf{e}_2$ parts of weight $> \omega_2$ which would imply a total weight $> \omega - \omega_1$ of $\mathbf{e}$ on its last $n-k$ coordinates). Here, b is another constant, integer parameter of the SCD algorithm.

In summary, for every slice i, SCD computes a set $\mathcal{K}_i$ of candidate vectors $(\mathbf{e}_1, \mathbf{e}_2)$ (matching the respective slice of the syndrome) with weight ω_1 in the first k coordinates and weight $\leq \omega_2$ in coordinates $k + i\rho, \dots, k + (i+1)\rho - 1$. Afterwards, one computes intersections of b candidate sets, where the intersection is computed w.r.t. the first k coordinates $\mathbf{e}_1$. As we have seen, at least one of these intersections will contain the first k coordinates of the wanted error vector $\mathbf{e}$. Clearly, the first k error positions of the received word $\mathbf{x}$ allows one to recover $\mathbf{c}$ by linear algebra.

The authors of [BKvT99] claimed that their method provides a better asymptotic complexity than all other known decoding algorithms (including the former state of the art ISD algorithms) but as indicated by Bernstein et al. in [BLP11a, Section 4], the analysis of SCD is flawed. More precisely, the authors estimated the output size of the intersections $\bigcap_{k=1}^{b} \mathcal{K}_{i_k}$, see [BKvT99, Corollary 12], as

$$\binom{k}{\omega_1}\binom{\rho}{\omega_2} 2^{-b\rho} \ . \tag{4.10}$$

Note that every candidate $(\mathbf{e}_1, \mathbf{e}_2)$ matches a given slice of the syndrome with probability $2^{-\rho}$ and thus the sets $\mathcal{K}_i$ have expected size $S := \binom{k}{\omega_1}\binom{\rho}{\omega_2} 2^{-\rho}$. Since the intersection is computed w.r.t. the first k coordinates, i.e. one only forces the $\mathbf{e}_1$ parts to intersect, the

expected size of $\bigcap_{k=1}^{b} \mathcal{K}_{i_k}$ is given by

$$\binom{k}{\omega_1}\left(S\binom{k}{\omega_1}^{-1}\right)^{b} = \binom{k}{\omega_1}\binom{\rho}{\omega_2}^{b} 2^{-b\rho} \tag{4.11}$$

which deviates by an exponential factor $\binom{\rho}{\omega_2}^{b-1}$ from Eq.(4.10). Note that this correction is in accordance with the incidental remark made by Bernstein et al. in [BLP11a]. A careful corrected numerical analysis of SCD shows that no further improvement upon PSSD is possible.

We conclude this chapter with an interesting question for future research.

Is it possible to combine different generic decoding algorithms?

For example, we tried to use statistical decoding to precompute some information about the error positions of the received word. We hoped that even for rather small test sets, SD might yield some information about the error positions in a received word. This information could, inter alia, be used to improve the probability of guessing a good permutation in ISD algorithms. We tested this approach experimentally for rather small random codes. Unfortunately, SD did not provide any useful information about the error positions for reasonably sized test sets.

4.3 Stern's Algorithm and Ball-collision Decoding

We will now extend plain ISD to a more general class of algorithms. This class will cover both Stern's algorithm [Ste89] and the recently proposed Ball-collision decoding (BCD for short) [BLP11a]. Recall from plain ISD that we transformed $\mathbf{H}$ into standard form, i.e. $\tilde{\mathbf{H}} = \begin{pmatrix}\mathbf{Q} & \mathbf{I}_{n-k}\end{pmatrix}$. We extend plain ISD based on the following two observations.

> *1. It is very unlikely that the permuted vector $\tilde{\mathbf{e}}$ exclusively contains non-zero coordinates in its last $n-k$ coordinates.*

This observation was first made by Lee and Brickell in [LB88] and was improved by Leon in [Leo88]. Lee and Brickell proposed an algorithm that simply assigns a certain weight p to the first k coordinates, i.e. in every iteration of plain ISD one tests every linear combination of exactly p columns of $\mathbf{Q}$ and computes its Hamming distance from $\tilde{\mathbf{s}}$. If this distance is exactly $\omega - p$, we can add to our p columns of $\mathbf{Q}$ the $\omega - p$ unit vectors from $\mathbf{I}_{n-k}$ that yield $\tilde{\mathbf{s}}$. Peters showed in [Pet11] that the Lee-Brickell approach improves upon plain ISD by a polynomial factor. Moreover, the optimal choice for p only depends on R and W and is thus *independent* of the code length n. This implies that both methods are asymptotically equivalent. To gain an exponential factor, Stern was the first who additionally proposed to

> *2. Compute the colum sums in every iteration more efficiently by a "Meet-in-the-Middle approach" .*

This idea shares some similarities with supercode decoding: We consider the supercode given by the first ℓ rows of $\tilde{\mathbf{H}}$, i.e. we consider the slice

$$\begin{pmatrix} \mathbf{Q}' & \mathbf{I}_\ell & \mathbf{0} \end{pmatrix} \tag{4.12}$$

together with the projected syndrome $\tilde{\mathbf{s}}' := \tilde{\mathbf{s}}_{[1,\ell]}$. Analogously, we decompose the error vector $\tilde{\mathbf{e}}$ into $(\tilde{\mathbf{e}}_1 \| \tilde{\mathbf{e}}_2 \| \tilde{\mathbf{e}}_3)$ where each part corresponds to the respective block of Eq.(4.12). Let us further assume that there occur exactly p errors in $\tilde{\mathbf{e}}_1$ and q errors in $\tilde{\mathbf{e}}_2$ (thus there must be $\omega - p - q$ errors in $\tilde{\mathbf{e}}_3$). Since the error vector fulfils the equation

$$\mathbf{Q}'\tilde{\mathbf{e}}_1 + \tilde{\mathbf{e}}_2 = \tilde{\mathbf{s}}' \tag{4.13}$$

one might compute all solutions $(\tilde{\mathbf{e}}_1, \tilde{\mathbf{e}}_2)$ with $\mathrm{wt}(\tilde{\mathbf{e}}_1) = p$ and $\mathrm{wt}(\tilde{\mathbf{e}}_2) = q$ as a first step. In contrast to a naive enumeration of all such vectors (as done by Leon for the case $q = 0$), one can use the linearity of the equation as already discussed for split syndrome decoding in Section 4.2, i.e. one splits both $\mathbf{Q}'$ and $\mathbf{I}_\ell$ into two half-sized submatrices and analogously decomposes $\tilde{\mathbf{e}}_1$ and $\tilde{\mathbf{e}}_2$ into two halves each containing half of the p and q errors, respectively. More precisely:

- Decompose $\mathbf{Q}' = \begin{pmatrix} \mathbf{Q}'_1 & \mathbf{Q}'_2 \end{pmatrix}$ into two $\ell \times \frac{k}{2}$ submatrices and analogously split $\tilde{\mathbf{e}}_1 = \begin{pmatrix} \tilde{\mathbf{e}}_{1,1} & \tilde{\mathbf{e}}_{1,2} \end{pmatrix}$.
- Decompose $\mathbf{I}_\ell = (\mathbf{J}_1 \| \mathbf{J}_2) := \begin{pmatrix} \mathbf{I}_{\ell/2} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{\ell/2} \end{pmatrix}$ and analogously $\tilde{\mathbf{e}}_2 = \begin{pmatrix} \tilde{\mathbf{e}}_{2,1} & \tilde{\mathbf{e}}_{2,2} \end{pmatrix}$.
- Compute two lists $\mathcal{L}_1$ and $\mathcal{L}_2$ containing elements

  $$(\mathbf{Q}'_1\tilde{\mathbf{e}}_{1,1} + \mathbf{J}_1\tilde{\mathbf{e}}_{2,1} \ , \ \tilde{\mathbf{e}}_{1,1} \ , \ \tilde{\mathbf{e}}_{2,1}) \tag{4.14}$$

  and

  $$(\mathbf{Q}'_2\tilde{\mathbf{e}}_{1,2} + \mathbf{J}_2\tilde{\mathbf{e}}_{2,2} + \tilde{\mathbf{s}}' \ , \ \tilde{\mathbf{e}}_{1,2} \ , \ \tilde{\mathbf{e}}_{2,2}) \ , \tag{4.15}$$

  where $\tilde{\mathbf{e}}_{1,j} \in W_{\frac{k}{2},\frac{p}{2}}$ and $\tilde{\mathbf{e}}_{2,j} \in W_{\frac{\ell}{2},\frac{q}{2}}$ for $j = 1, 2$.
- Search for elements in $\mathcal{L}_1 \times \mathcal{L}_2$ with matching first entries.

From now on, we will call the first entries of the respective list elements in $\mathcal{L}_1$ and $\mathcal{L}_2$ a **label**. By definition of the above labels, every collision between $\mathcal{L}_1$ and $\mathcal{L}_2$ gives rise to a solution $\tilde{\mathbf{e}}_1 := (\tilde{\mathbf{e}}_{1,1} \| \tilde{\mathbf{e}}_{1,2})$ and $\tilde{\mathbf{e}}_2 := (\tilde{\mathbf{e}}_{2,1} \| \tilde{\mathbf{e}}_{2,2})$ of Eq.(4.13). For every such candidate solution, we have to check whether one of them can be extended to *all* $n - k$ rows of $\tilde{\mathbf{H}}$: We compute the Hamming distance from $\mathbf{Q}\tilde{\mathbf{e}}_1 + (\tilde{\mathbf{e}}_2 \| \mathbf{0})$ to $\tilde{\mathbf{s}}$ over *all* $n - k$ coordinates (where $\tilde{\mathbf{e}}_2$ has to be padded with $n - k - \ell$ zeros). If this distance is $\omega - p - q$, we can appropriately fill up the remaining error positions of $\tilde{\mathbf{e}}_3$. The overall algorithm is summarised in Algorithm 2 where Init is defined as in Algorithm 1 and Merge-Join is a procedure that efficiently computes the matchings between $\mathcal{L}_1$ and $\mathcal{L}_2$, see Chapter 6 for a formal definition and runtime analysis.

Algorithm 2: Ball-Collision Decoding

input : Parity check matrix $\mathbf{H}$, syndrome $\mathbf{s}$ and target weight ω.
output : Error vector $\mathbf{e}$ with $\mathbf{He} = \mathbf{s}$ and $\mathrm{wt}(\mathbf{e}) = \omega$.
params: $0 \leq \ell \leq n-k$, $0 \leq q \leq \min\{\ell, \omega\}$,
$\max\{0, k+\ell+\omega-q-n\} \leq p \leq \min\{\omega-q, k\}$.

```
found ← false;
while not found do
    (H̃, s̃, P) ← Init(H, s);
    Compute L_1 and L_2 according to Eq.(4.14) and (4.15);
    L ← Merge-Join(L_1, L_2);
    forall (ẽ_1, ẽ_2) ∈ L do
        if wt(Qẽ_1 + (ẽ_2||0) + s̃) = ω − p − q then
            ẽ ← (ẽ_1||Qẽ_1 + s̃);
            found ← true;
return Pẽ;
```

We point out that the algorithm is well-defined for some interesting special cases: For $p = q = \ell = 0$ the lists $\mathcal{L}_1$ and $\mathcal{L}_2$ will only contain the all-zero vector of length $\frac{k}{2}$. In this case, the weight check degenerates to $\mathrm{wt}(\tilde{\mathbf{s}}) = \omega$ and we exactly recover Plain ISD. Moreover, the case $q = 0$ corresponds to Stern's agorithm. In fact, the introduction of the parameter q was the main contribution of [BLP11a].

Clearly, the algorithm outputs a *correct* solution: It holds that

$$\tilde{\mathbf{H}}\tilde{\mathbf{e}} = \begin{pmatrix} \mathbf{Q} & \mathbf{I}_{n-k} \end{pmatrix} \begin{pmatrix} \tilde{\mathbf{e}}_1 \\ \mathbf{Q}\tilde{\mathbf{e}}_1 + \tilde{\mathbf{s}} \end{pmatrix} = \tilde{\mathbf{s}}$$

which implies $\mathbf{He} = \mathbf{s}$. Note that $\mathbf{Q}\tilde{\mathbf{e}}_1 + (\tilde{\mathbf{e}}_2 \| \mathbf{0}) + \tilde{\mathbf{s}} = \mathbf{0}$ holds on the first ℓ coordinates which implies

$$\mathrm{wt}(\mathbf{Q}\tilde{\mathbf{e}}_1 + \tilde{\mathbf{s}}) = \mathrm{wt}(\mathbf{Q}\tilde{\mathbf{e}}_1 + (\tilde{\mathbf{e}}_2 \| \mathbf{0}) + \tilde{\mathbf{s}}) + \mathrm{wt}(\tilde{\mathbf{e}}_2) = \omega - p - q + q = \omega - p$$

and we finally obtain

$$\mathrm{wt}(\mathbf{e}) = \mathrm{wt}(\mathbf{P}\tilde{\mathbf{e}}) = \mathrm{wt}(\tilde{\mathbf{e}}) = \mathrm{wt}(\tilde{\mathbf{e}}_1 \| \mathbf{Q}\tilde{\mathbf{e}}_1 + \tilde{\mathbf{s}}) = p + \omega - p = \omega \ .$$

We now parametrize $p = \lfloor Pn \rfloor$, $q = \lfloor Qn \rfloor$ and $\ell = \lfloor Ln \rfloor$ in order to prove the following asymptotic statement. We define $0 \cdot \mathrm{H}\left(\frac{x}{0}\right) := 0$ for convenience.

Theorem 4.3.1 (Ball-collision Decoding)**.** *Let* $0 < R < 1$ *and* $0 < W \leq \mathrm{D}_{\mathsf{GV}}(R)$*. For almost all binary* $[n, \lfloor Rn \rfloor]$*-codes it holds: Ball-collision decoding successfully terminates in time* $2^{F_{\mathsf{Ball}}(R,W)n+o(n)}$ *for every error vector* $\mathbf{e} \in W_{n,\lfloor Wn \rfloor}$ *with overwhelming probability where*

$$F_{\mathsf{Ball}}(R, W) = \min_{P,Q,L} \{N_{\mathsf{Ball}}(R, W, P, Q, L) + \max\{\Lambda_{\mathsf{Ball}}(R, W, P, Q), 2\Lambda_{\mathsf{Ball}}(R, W, P, Q) - L\}\} \quad , \tag{4.16}$$

$$N_{\mathsf{Ball}}(R, W, P, Q, L) = \mathrm{H}(W) - R\,\mathrm{H}\left(\frac{P}{R}\right) - L\,\mathrm{H}\left(\frac{Q}{L}\right) - (1 - R - L)\,\mathrm{H}\left(\frac{W - P - Q}{1 - R - L}\right) \quad , \tag{4.17}$$

$$\Lambda_{\mathsf{Ball}}(R, W, P, Q) = \frac{R}{2}\,\mathrm{H}\left(\frac{P}{R}\right) + \frac{L}{2}\,\mathrm{H}\left(\frac{Q}{L}\right) \tag{4.18}$$

with $0 \leq L \leq 1 - R$*,* $0 \leq Q \leq \min\{L, W\}$ *and* $\max\{0, R + L + W - Q - 1\} \leq P \leq \min\{R, W - Q\}$*.*

Remark 4.3.2. Since we defined $0 \cdot \mathrm{H}\left(\frac{x}{0}\right) := 0$, the formulas in Theorem 4.3.1 are also well-defined for parameters at the corners of the parameter space, e.g. for $P = Q = L = 0$ one recovers Plain ISD.

The proof of Theorem 4.3.1 is rather similar to the proof of Theorem 4.1.2. In particular, one might treat *defective permutations* analogously to Section 4.1.2. We further make use of the following proposition whose proof will follow from the generalised Merge-Join procedure of Chapter 6.

Proposition 4.3.3. Merge-Join *computes* $\mathcal{L}$ *in time* $\mathcal{O}\left(\max\{|\mathcal{L}_i| \log |\mathcal{L}_i|, |\mathcal{L}|\}\right)$*.*

Proof of Theorem 4.3.1. The probability of guessing a good permutation is given by

$$\binom{k/2}{p/2}^2 \binom{\ell/2}{q/2}^2 \binom{n-k-\ell}{\omega-p-q} \binom{n}{\omega}^{-1} \tag{4.19}$$

since the way we compute solutions to Eq.(4.13) puts the following restrictions on the error distribution of $\tilde{\mathbf{e}}$. We enumerate only those $\tilde{\mathbf{e}}_1$ that can be split into two vectors of length $\frac{k}{2}$ and weight $\frac{p}{2}$ and similarly enumerate only those $\tilde{\mathbf{e}}_2$ that split into two vectors of length $\frac{\ell}{2}$ and weight $\frac{q}{2}$. Consequently, the remaining $\omega - p - q$ error positions must be contained in the last $n - k - \ell$ coordinates. Thus, the above numerator exactly counts the number of error patterns we can find. Using Eq.(2.15), the contribution of the number

of iterations to F_{Ball} is given by

$$N_{\mathsf{Ball}}(R, W, P, Q, L) := \mathrm{H}(W) - R\,\mathrm{H}\left(\frac{P}{R}\right) - L\,\mathrm{H}\left(\frac{Q}{L}\right) - (1-R-L)\,\mathrm{H}\left(\frac{W-P}{1-R-L}\right) . \tag{4.20}$$

It remains to estimate the complexity of one iteration. It holds $|\mathcal{L}_i| = \binom{k/2}{p/2}\binom{\ell/2}{q/2}$. To estimate $|\mathcal{L}|$, we need to upper bound the number of collisions between $\mathcal{L}_1$ and $\mathcal{L}_2$. Clearly, the expected number of collisions is $\frac{|\mathcal{L}_1|\cdot|\mathcal{L}_2|}{2^\ell}$. In order to prove that the actual number of collisions does not deviate significantly from this number, we adapt the proof of Corollary 2.4.10: Consider the random $[k+\ell, k]$-code $\mathcal{C}$ defined by the parity check matrix $\mathbf{H}' := (\mathbf{Q}', \mathbf{I}_\ell)$ according to Eq.(4.12) and define the set

$$\mathcal{S} := W_{\frac{k}{2},\frac{p}{2}} \times W_{\frac{k}{2},\frac{p}{2}} \times W_{\frac{\ell}{2},\frac{q}{2}} \times W_{\frac{\ell}{2},\frac{q}{2}} ,$$

i.e. $|\mathcal{S}| = |\mathcal{L}_1|\cdot|\mathcal{L}_2|$. Using the same notation as in Corollary 2.4.10, the number of collisions is a random variable $N(\mathcal{S})$ counting the number of elements in $\mathcal{S}$ that are contained in the coset of $\mathcal{C}$ defined by $\tilde{\mathbf{s}}'$. Thus, applying Lemma 2.4.6 gives $\mathbb{E}[N(\mathcal{S})] = \frac{|\mathcal{S}|}{2^\ell}$ and $\mathrm{Var}[N(\mathcal{S})] \le \mathbb{E}[N(\mathcal{S})]$. In the asymptotic setting, we have

$$\Lambda_{\mathsf{Ball}}(R, W, P, Q, L) := \lim_{n\to\infty} \frac{1}{n} \log|\mathcal{L}_i| = \frac{R}{2}\,\mathrm{H}\left(\frac{P}{R}\right) + \frac{L}{2}\,\mathrm{H}\left(\frac{Q}{L}\right) \tag{4.21}$$

and thus $\mathbb{E}[N(\mathcal{S})] = 2^{(2\Lambda-L)n+o(n)}$. If $L < 2\Lambda$, $\mathbb{E}[N(S)]$ is exponentially increasing in n and by Chebychev's inequality we get

$$\mathbf{Pr}\left[|N(\mathcal{S}) - \mathbb{E}[N(\mathcal{S})]| \ge \mathbb{E}[N(\mathcal{S})]\right] \le \frac{1}{\mathbb{E}[N(\mathcal{S})]} ,$$

i.e. $N(\mathcal{S}) = 2^{(2\Lambda-L)n+o(n)}$ for almost all codes. If $L \ge 2\Lambda$, we fix some $\varepsilon > 0$ and obtain

$$\mathbf{Pr}\left[|N(\mathcal{S}) - \mathbb{E}[N(\mathcal{S})]| \ge 2^{\varepsilon n}\right] \le 2^{(2\Lambda-L-\varepsilon)n+o(n)} = 2^{-\varepsilon' n+o(n)}$$

for some $\varepsilon' > 0$, i.e. $N(\mathcal{S}) \le 2^{\varepsilon n+o(n)}$ for $\varepsilon > 0$ arbitrarily small. Altogether, we obtain

$$\lim_{n\to\infty} \frac{1}{n} \log|\mathcal{L}| \le \begin{cases} 2\Lambda - L & L < 2\Lambda \\ \varepsilon & L \ge 2\Lambda \end{cases} \tag{4.22}$$

for $\varepsilon > 0$ arbitrarily small with probability $1-2^{-\Omega(n)}$ *for a single iteration of the algorithm.* However, one normally repeats the inner loop of the algorithm *exponentially many times.* It is thus conceivable that the expected number of *bad iterations*, in which the actual value of $N(\mathcal{S})$ exceeds $\mathbb{E}[N(\mathcal{S})]$ by an exponential factor, is large. Since a *single* bad iteration is enough to contradict our "worst-case" statement about the running time, we need to carry out the following artificial modification of the algorithm. Simply abort every bad iteration as soon as the number of collisions grows to large. This clearly allows to rigorously upper bound the (considered) number of collisions for *all* iterations.

Unfortunately, this solution introduces another technical caveat. One might fail in finding a fixed solution $\mathbf{e}$ due to an aborted iteration. However, for a fixed solution $\mathbf{e}$, this failure probability can be bounded as follows. Consider the first iteration that originates from a good permutation (w.r.t. $\mathbf{e}$). As we have seen before (see Lemma 4.1.4), one picks at least one good permutation amongst $2^{(N_{\mathsf{Ball}}+o(1))n}$ iterations with probability $1 - 2^{-\Omega(n)}$. In order to apply the above concentration result for $N(\mathcal{S})$ w.r.t. the code $\mathcal{C}$ generated by the submatrix $(\mathbf{Q}', \mathbf{I}_\ell)$ in this specific iteration, it is sufficient to argue that $\mathcal{C}$ is in fact a random code (independent from $\mathbf{e}$). This holds since the submatrix $(\mathbf{Q}', \mathbf{I}_\ell)$ is generated from the uniform input matrix $\mathbf{H}$ as follows:

1. Pick a good permutation σ (w.r.t. $\mathbf{e}$) and consider the submatrix $\mathbf{H}'$ of $\mathbf{H}$ containing columns $\sigma(1), \ldots, \sigma(k + \ell)$. The distribution of this submatrix is clearly uniform (and independent from $\mathbf{e}$ since *all* columns of $\mathbf{H}$ have been chosen independent from $\mathbf{e}$).

2. Consider the submatrix $\mathbf{H}''$ of $\mathbf{H}'$ containing the first ℓ rows of $\mathbf{H}'$ and transform $\mathbf{H}'$ into standard form $(\mathbf{Q}', \mathbf{I}_\ell)$. Since $\mathbf{H}''$ still generates a uniform random code (independent from $\mathbf{e}$), so does $(\mathbf{Q}', \mathbf{I}_\ell)$.

Altogether, this implies that the number of collisions in the first good iteration is bounded by Eq.(4.22) with probability $1 - 2^{-\Omega(n)}$. Consequently, the first good iteration (w.r.t. $\mathbf{e}$) will almost never be aborted and the overall probability of finding a fixed solution $\mathbf{e}$ is $1 - 2^{-\Omega(n)}$.

For the sake of completeness, we point out that the extra logarithmic factors $\log|\mathcal{L}_i|$ that appear in the running time of Merge-Join due to Proposition 4.3.3 do not affect the complexity coefficient of the algorithm. Combining Eq.(4.20)-(4.22) eventually finishes the proof.

□

In contrast to plain ISD, the above proof shows that the extended algorithm has exponential *space* complexity Λ_{Ball}. Note that the merged list $\mathcal{L}$ must not be entirely stored since every candidate solution $(\tilde{\mathbf{e}}_1, \tilde{\mathbf{e}}_2)$ can be checked on-the-fly.

Comparison between Stern's algorithm and BCD

Unfortunately, we do not know explicit formulas (depending on R and W) for optimal parameters P, Q and L that occur in Theorem 4.3.1. Consequently, we can not give a closed-form expression for the complexity coefficient. To compare different algorithms, one usually runs a numerical optimisation and computes best parameters for different code rates R. One then interpolates the complete complexity curve for all $0 < R < 1$. In Table 4.1 we present optimal parameter choices for Stern's algorithm whose complexity coefficient is denoted F_{Ste} and is obtained by fixing $Q = 0$ in Eq.(4.16).

R	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
P	.00368	.00598	.00763	.00863	.00923	.00929	.0088	.00758	.00516
L	.01138	.01938	.02565	.03005	.03317	.03459	.0341	.03093	.02291
F_{Ste}	.05742	.08923	.1075	.11569	.11547	.10771	.09279	.07064	.04056
Λ_{Ste}	.01138	.01939	.02566	.03006	.03318	.0346	.0341	.03094	.02292

Table 4.1: Optimal parameters P, L for Stern's algorithm for $W = \mathrm{D}_{\mathsf{GV}}(R)$ and resulting time and space complexities F_{Ste} and Λ_{Ste} respectively.

In the above table, the values for P and L are rounded to the nearest multiple of 10^{-5} whereas the time and space complexity coefficients are always rounded *up* to the next multiple of 10^{-5}. Note that, up to rounding differences, it always holds $L = \Lambda_{\mathsf{Ste}}$ which perfectly balances the two arguments that occur in the maximum of Eq.(4.16). The same holds for BCD as presented in the next table.

R	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
P	.00415	.0065	.00787	.00871	.00996	.01035	.00893	.00746	.00525
Q	.00062	.00078	.00077	.00054	.00069	.00065	.00053	.00032	.00013
L	.01432	.02315	.02884	.03224	.03771	.04011	.0365	.03179	.02384
F_{Ball}	.05729	.089	.10721	.1154	.11516	.10743	.09255	.07047	.04048
Λ_{Ball}	.01432	.02315	.02884	.03224	.03771	.04012	.0365	.0318	.02384

Table 4.2: Optimal parameters P, L, Q for BCD for $W = \mathrm{D}_{\mathsf{GV}}(R)$.

Note that for all tested values of R it holds $F_{\mathsf{Ball}}(R) < F_{\mathsf{Ste}}(R)$. Also note that this improvement comes at the cost of an increased space consumption, i.e. $\Lambda_{\mathsf{Ball}}(R) > \Lambda_{\mathsf{Ste}}(R)$, and one might ask the question whether Ball-Collision is merely a time-memory trade-off for Stern's algorithm. This issue will further be discussed in comparison to our new algorithms in Chapter 6.

Clearly, the above numerical comparison does not provide any proof for the general superiority of BCD over Stern's algorithm. A formal proof for the suboptimality of $Q = 0$ *for all* R was given by Bernstein et al. as subsumed in Theorem 4.3.4. Providing a similar result for our new algorithm is one of the central goals of this work.

Theorem 4.3.4 ([BLP11a, Theorem 7.1]). *For every* $0 < R < 1$ *and* $0 < W \leq \mathrm{D}_{\mathsf{GV}}(R)$ *it holds*

$$F_{\mathsf{Ball}}(R, W) < F_{\mathsf{Ste}}(R, W) \ .$$

In particular, for optimal BCD parameters (P^*, Q^*, L^*) *it must always hold* $Q^* > 0$ *(and thus* $L^* > 0$*).*

We conclude this chapter with the following informative illustration.

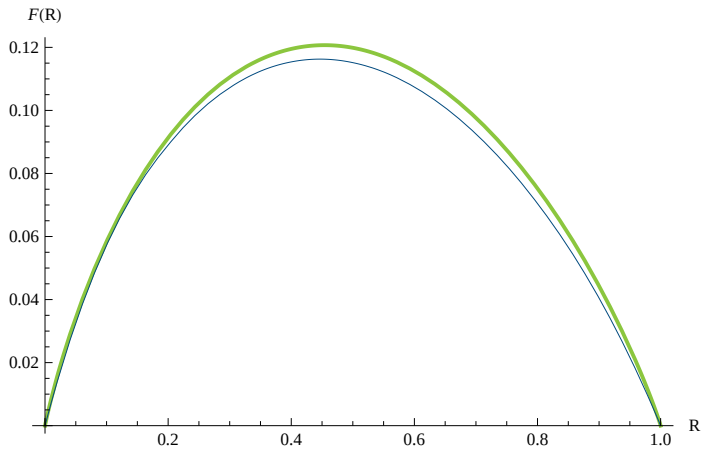


Figure 4.4: F_{Pra} (thick curve) and interpolated F_{Ball} (thin curve) for $W = \mathrm{D}_{\mathsf{GV}}(R)$.

We did not plot the interpolated curve for Stern's algorithm which is almost identical to the BCD curve. The respective maxima over all rates $0 < R < 1$ are obtained for slightly varying R as summarised in the table below.

Algorithm	$\max_R F(R)$	$\arg\max F(R)$
Prange	.1208	.4540
Stern	.1166	.4468
BCD	.1163	.4466

Table 4.3: "Worst-case" complexities $F(R)$ and corresponding rates R.

Quick Reference to Notation

The next two pages serve as a quick reference that aims to facilitate the further reading of this thesis. We summarize the notation that is used for the description and the analysis of all subsequent algorithms. The particular intention of this compact overview is to ease the understanding of the technical proofs in Chapter 5-7.

General Parameters

The first table shows absolute parameters (first column), relative parameters w.r.t. n (second column) and their algorithmic meaning.

abs.	**rel.**	**meaning**
k	R	input code rate
ω	W	input target weight
ℓ	L	number of rows of submatrix $\mathbf{Q}'$ (cf. Figure 5.1)
p	P	Hamming weight of partial solution $\tilde{\mathbf{e}}_1$, see Eq.(5.2)
q	Q	Hamming weight amongst coordinates $k+1,\ldots,k+\ell$ (only for BCD)
-	α	controls the (relative) number of overlapping coordinates
δ	Δ	controls the number of intersecting error positions within the overlapping coordinates
δ_i	Δ_i	controls the number of intersecting error positions on layer i (only occurs in the recursive variant)

Ball-collision Decoding

- **Number of iterations**

$$N_{\mathsf{Ball}}(R,W,P,Q,L) = \mathrm{H}(W) - R\,\mathrm{H}\left(\frac{P}{R}\right) - L\,\mathrm{H}\left(\frac{Q}{L}\right) - (1-R-L)\,\mathrm{H}\left(\frac{W-P-Q}{1-R-L}\right)$$

- **List size**

$$\Lambda_{\mathsf{Ball}}(R,W,P,Q) = \frac{R}{2}\,\mathrm{H}\left(\frac{P}{R}\right) + \frac{L}{2}\,\mathrm{H}\left(\frac{Q}{L}\right)$$

- **Complexity**

$$\begin{aligned} T_{\mathsf{Ball}}(R,W,P,Q,L) &= N_{\mathsf{Ball}} + \max\{\Lambda_{\mathsf{Ball}}, 2\Lambda_{\mathsf{Ball}} - L\} \\ &= \mathrm{H}\,W - (1-R-L)\,\mathrm{H}\left(\frac{W-P-Q}{1-R-L}\right) + \max\{-L, \Lambda_{\mathsf{Ball}}\} \end{aligned}$$

- **Number of iterations**

$$N(R, W, L, P) = \mathrm{H}(W) - (R+L)\,\mathrm{H}\left(\frac{P}{R+L}\right) - (1-R-L)\,\mathrm{H}\left(\frac{W-P}{1-R-L}\right)$$

- **Number of representations**

$$\mathfrak{P}(R, L, P, \alpha, \Delta) = 2\alpha\left(P + (R+L-P)\,\mathrm{H}\left(\frac{\Delta}{2\alpha(R+L-P)}\right)\right)$$

- Size of the sets W_1 and W_2 (used to construct / sample base lists)

$$\Sigma(R, L, P, \alpha, \Delta) = \left(\frac{1}{2}+\alpha\right)(R+L)\,\mathrm{H}\left(\frac{\frac{P}{2}+\Delta}{(\frac{1}{2}+\alpha)(R+L)}\right)$$

FS-ISD $\alpha = \Delta = 0$

- **List size**

$$\Lambda_{\mathsf{FS-ISD}}(R, L, P) = \frac{R+L}{2}\,\mathrm{H}\left(\frac{P}{R+L}\right)$$

- **Complexity**

$$\begin{aligned} T_{\mathsf{FS-ISD}}(R, W, L, P) &= N(R, W, L, P) + \max\{\Lambda_{\mathsf{FS-ISD}}, 2\Lambda_{\mathsf{FS-ISD}} - L\} \\ &= \mathrm{H}(W) - (1-R-L)\,\mathrm{H}\left(\frac{W-P}{1-R-L}\right) + \max\{-L, -\Lambda_{\mathsf{FS-ISD}}\} \end{aligned}$$

BasicReps $\alpha = \frac{1}{2}, \Delta > 0$

- **Size of base lists**

$$\frac{\Sigma}{2} = \frac{R+L}{2}\,\mathrm{H}\left(\frac{\frac{P}{2}+\Delta}{R+L}\right)$$

- **Size of merged lists**

$$\Sigma - \mathfrak{P} = (R+L)\,\mathrm{H}\left(\frac{\frac{P}{2}+\Delta}{R+L}\right) - \left[P + (R+L-P)\,\mathrm{H}\left(\frac{\Delta}{R+L-P}\right)\right]$$

- **Number of collisions**

$$2\Lambda - L = 2\Sigma - \mathfrak{P} - L = 2(R+L)\,\mathrm{H}\left(\frac{\frac{P}{2}+\Delta}{R+L}\right) - \left[P + (R+L-P)\,\mathrm{H}\left(\frac{\Delta}{R+L-P}\right)\right] - L$$

- **Complexity**

$$T_{\mathsf{BReps}}(R, W, L, P, \Delta) = N(R, W, L, P) + \max\left\{\frac{\Sigma}{2}, \Sigma - \mathfrak{P}, 2\Lambda - L\right\}$$

The reader is friendly advised to leave a thumb (or another more favourable finger) on this page.

5 A Generalised Model for Information Set Decoding

"All generalisations are false, including this one."

Marc Twain

Introduction

Let us recall the main principle of the ISD algorithms presented so far. Essentially, these algorithms can be divided into three steps:

- Compute a randomly column-permuted standard form $\tilde{\mathbf{H}}$.
- Consider the slice $\begin{pmatrix}\mathbf{Q}' & \mathbf{I}_\ell & \mathbf{0}\end{pmatrix}$ containing the firs ℓ rows of $\tilde{\mathbf{H}}$. Create a list $\mathcal{L}$ of candidate solutions $(\tilde{\mathbf{e}}_1, \tilde{\mathbf{e}}_2)$ to the equation $\mathbf{Q}'\tilde{\mathbf{e}}_1 + \tilde{\mathbf{e}}_2 = \tilde{\mathbf{s}}'$ where $\tilde{\mathbf{e}}_1$ and $\tilde{\mathbf{e}}_2$ have weight p and q, respectively, and $\tilde{\mathbf{s}}'$ is the syndrome projected onto the first ℓ coordinates.
- Try to extend candidate solutions from $\mathcal{L}$ to all $n-k$ coordinates.

Furthermore, all methods presented in Section 4.3 *deterministically* solved the second step by splitting the $\tilde{\mathbf{e}}_i$ into two *disjoint* halves, i.e. we decomposed $\tilde{\mathbf{e}}_i = \tilde{\mathbf{e}}_{i,1} + \tilde{\mathbf{e}}_{i,2}$ with $\mathrm{supp}(\tilde{\mathbf{e}}_{i,1}) \cap \mathrm{supp}(\tilde{\mathbf{e}}_{i,2}) = \varnothing$. In this chapter, we will show how modifying the first two steps yields a generalised ISD framework that offers the following advantages:

- The model yields a simplified version of Ball-collision decoding where the parameter q is removed.
- It generalizes another ISD framework introduced by Finiasz and Sendrier in [FS09].
- It allows to relate different decoding methods (e.g. we can proof the equivalence of BCD and PSSD).
- We can instantiate a basic variant of our new algorithm within the framework, see Chapter 6, and formally proof its superiority.

Let us now begin with a high-level overview of the main modifications of our generalised ISD framework in comparison to the ISD algorithms presented thus far.

Quasi-standard form. As first observed by Finiasz and Sendrier in [FS09], it is not necessary to transform $\mathbf{H}$ into diagonal form on coordinates $k+1,\ldots,k+\ell$ in order to compute a list of candidate error vectors. More precisely, one might alternatively consider a *quasi-standard form*

$$\tilde{\mathbf{H}} := \begin{pmatrix} \mathbf{Q}' & \mathbf{0} \\ \mathbf{Q}'' & \mathbf{I}_{n-k-\ell} \end{pmatrix} , \tag{5.1}$$

where $\mathbf{Q}'$ is a $\ell \times (k+\ell)$ matrix, $\mathbf{Q}''$ is a $(n-k-\ell)\times(k+\ell)$ matrix and $\mathbf{0}$ is a $\ell\times(n-k-\ell)$-dimensional zero-matrix, and solve the equation

$$\mathbf{Q}'\tilde{\mathbf{e}}_1 = \tilde{\mathbf{s}}' \tag{5.2}$$

where $\tilde{\mathbf{e}}_1$ is a vector of length $k+\ell$ and weight p (and we analogously define $\tilde{\mathbf{e}}_2$ as a $(n-k-\ell)$-dimensional vector of weight $\omega - p$), see Figure 5.1 for an illustration. For ease of presentation, we will sometimes write $\mathbf{Q} = \begin{pmatrix}\mathbf{Q}' & \mathbf{Q}''\end{pmatrix}^\top$.

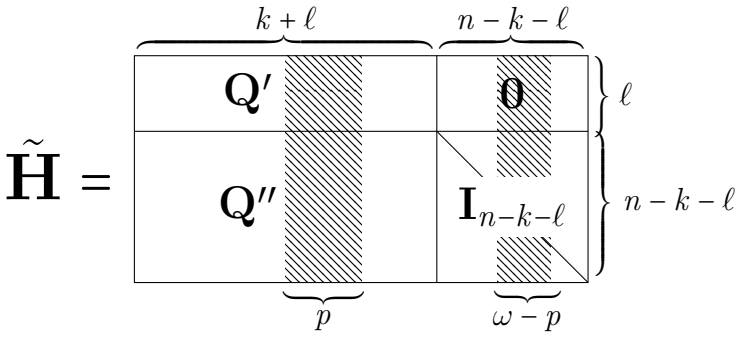


Figure 5.1: Parity check matrix $\tilde{\mathbf{H}}$ in quasi-systematic form. The shaded area represents an exemplary selection of p and $\omega - p$ columns according to $\tilde{\mathbf{e}}_1$ and $\tilde{\mathbf{e}}_2$, respectively.

Intuitively, this transformation allows to remove the BCD parameter q which can implicitly be covered by a larger choice of p. More formally, in Section 5.2.2 we provide a concrete instantiation and we prove that it achieves at least the same efficiency as BCD. Given a quasi-standard form $\tilde{\mathbf{H}}$, the main computational task is to compute the list of candidate solutions $\mathcal{L}$ for Eq.(5.2). The remaining modifications have one goal only:

Find more efficient algorithms to compute $\mathcal{L}$.

More precisely, we will allow for an arbitrary, possibly randomised procedure FindCandidates that may fail in finding a solution with a certain probability, even if we picked a "good" permutation during the initial randomisation phase. This approach is based on the following two points.

More flexible decomposition of $\tilde{\mathbf{e}}_1$: Introducing Representations. Instead of decomposing $\tilde{\mathbf{e}}_1$ into support-disjoint vectors $\tilde{\mathbf{e}}_{1,i}$, we allow them to overlap in a 2α-fraction of their coordinates, i.e. for some $\alpha \in [0, \frac{1}{2}]$ we fix

$$\begin{aligned} \mathrm{supp}(\tilde{\mathbf{e}}_{1,1}) &\subset \{1, \dots, \lfloor (1/2 + \alpha)\,(k+\ell) \rfloor\} \\ \mathrm{supp}(\tilde{\mathbf{e}}_{1,2}) &\subset \{\lfloor (1/2 - \alpha)\,(k+\ell) \rfloor + 1, \dots, k+\ell\} \ . \end{aligned} \tag{5.3}$$

Remark 5.0.5. For the sake of a better readability, we drop all rounding operations from now on and implicitly assume that all parameters can be chosen as integers if needed.

The main consequence of this modification is that one introduces an exponential number of **representations** of $\tilde{\mathbf{e}}_1$ (see Definition 1.0.2). That is we obtain many ways to represent $\tilde{\mathbf{e}}_1$ as a sum $\tilde{\mathbf{e}}_{1,1} + \tilde{\mathbf{e}}_{1,2}$. Loosely speaking, every 1 within the $2\alpha(k+\ell)$ overlapping coordinates can either be represented as $1+0$ or $0+1$, depending on whether we assign the error to either $\tilde{\mathbf{e}}_{1,1}$ or $\tilde{\mathbf{e}}_{1,2}$. The general idea of introducing representations can already be found in [FS09] but the proposed algorithm does not fully exploit their existence. We point out that introducing the parameter α serves the purpose of embedding different algorithms into the generalised framework that exploit representations to different extents. The most important cases will be $\alpha = 0$ (no representations) and $\alpha = \frac{1}{2}$ (maximal number of representations). In Section 5.2.4 we show that $\alpha = 0$ is suboptimal for sampling-based algorithms. Furthermore, it is not hard to see that $\alpha = \frac{1}{2}$ yields the best results for all algorithms based on the representation technique.

In addition to this, we provide a way to increase the number of representations: We introduce another parameter $\delta \in \{0, \dots, 2\alpha(k+\ell-p)\}$ and add δ many additional 1's to both $\tilde{\mathbf{e}}_{1,i}$. If those additional 1's occur amongst the overlapping coordinates of the $\tilde{\mathbf{e}}_{1,i}$, they might cancel out in their sum $\tilde{\mathbf{e}}_{1,1} + \tilde{\mathbf{e}}_{1,2}$ and yield a vector $\tilde{\mathbf{e}}_1$ of the correct Hamming weight p. Put simply, the extra parameter δ allows to represent δ many 0's within the overlapping coordinates of $\tilde{\mathbf{e}}_1$ as $0 = 1 + 1$, which explains the title of [BJMM12]. A precise statement about the number of representations can be found in Lemma 5.1.5. Altogether, we require both $\tilde{\mathbf{e}}_{1,i}$ to have Hamming weight $\frac{p}{2} + \delta$, namely:

$$\begin{aligned} \tilde{\mathbf{e}}_{1,1} \in W_1 &:= W_{(\frac{1}{2}+\alpha)(k+\ell), \frac{p}{2}+\delta} \times \{0\}^{(\frac{1}{2}-\alpha)(k+\ell)} \ , \\ \tilde{\mathbf{e}}_{1,2} \in W_2 &:= \{0\}^{(\frac{1}{2}-\alpha)(k+\ell)} \times W_{(\frac{1}{2}+\alpha)(k+\ell), \frac{p}{2}+\delta} \ , \end{aligned} \tag{5.4}$$

see Figure 5.2 for an illustration.

$$\begin{aligned} \tilde{\mathbf{e}}_1 &= \boxed{10100}\boxed{00010000100100000000}\boxed{00100} \\ \tilde{\mathbf{e}}_{1,1} &= \boxed{10100}\boxed{00000010010000001000}\boxed{00000} \\ \tilde{\mathbf{e}}_{1,2} &= \boxed{00000}\boxed{00010010000010001000}\boxed{00100} \end{aligned}$$

Figure 5.2: A possible representation $\tilde{\mathbf{e}}_{1,1} + \tilde{\mathbf{e}}_{1,2}$ of $\tilde{\mathbf{e}}_1$ for $k+\ell = 30$, $p = 6$, $\delta = 2$ and $\alpha = \frac{1}{3}$.

For a given $\mathbf{Q}'$ and $\tilde{\mathbf{s}}'$ we are eventually interested in the corresponding *set of candidate solutions* (where $W := W_{k+\ell,p}$)

$$\mathcal{S} := \{\tilde{\mathbf{e}}_1 \in (W_1 + W_2) \cap W \ : \ \mathbf{Q}'\tilde{\mathbf{e}}_1 = \tilde{\mathbf{s}}'\} \ . \tag{5.5}$$

Randomised construction of lists $\mathcal{L}_1$ and $\mathcal{L}_2$. Within the randomised procedure FindCandidates, we allow for an arbitrary computation of the candidate lists $\mathcal{L}_1$ and $\mathcal{L}_2$ whose entries will form an appropriately chosen subset of vectors $\tilde{\mathbf{e}}_{1,i} \in W_i$. Of particular interest are the following opposing approaches:

- Randomly *sample* $\lambda(\ell, p, \alpha, \delta)$ many vectors $\tilde{\mathbf{e}}_{1,i}$ (in Section 5.2.1 we explain how to choose λ).
- *Construct* small, particularly structured lists $\mathcal{L}_1$ and $\mathcal{L}_2$. This is done by cleverly exploiting the existence of the exponentially many representations.

To conclude this introductory section, we indicate how the generalised framework relates to different ISD algorithms:

- For $\alpha = \frac{1}{2}$ (full overlap), $\delta = 0$ (no extra representations) and *sampling* the $\mathcal{L}_i$, one obtains the original description of the Finiasz-Sendrier framework, see Section 5.2.4 for a detailed discussion.
- For $\alpha = 0$, $\delta = 0$ (i.e. *no* representations) and *constructing* the $\mathcal{L}_i$ we obtain a variant comparable to BCD, see Section 5.2.2.
- $\alpha = \frac{1}{2}$ and $\delta = 0$ and *constructing* the $\mathcal{L}_i$ covers the algorithm presented in [MMT11].
- $\alpha = \frac{1}{2}$ and $\delta > 0$ (i.e. *full* representations) yields the improved algorithm [BJMM12].

Roadmap

In the next section, we start with a formal definition of the generalised ISD framework and make some basic observations about representations. In Section 5.2 we show how to relate different ISD algorithms to the generalised framework. Of particular interest is the BCD variant of Section 5.2.2. For the purpose of the superiority proof of our new algorithm in Chapter 6, we provide basic properties of optimal parameters for the BCD variant in Section 5.2.3. As a last preparatory step we introduce a sampling-based algorithm in Section 5.2.5 that exploits representations in a suboptimal way.

5.1 Formal Definition and Basic Observations

We now formally introduce our generalised ISD framework. We denote by Init the initial transformation that brings $\mathbf{H}$ into *quasi-standard form* with randomly permuted columns as defined in Eq.(5.1) and illustrated in Figure 5.1. The crucial subroutine of the framework is the procedure FindCandidates: As input, FindCandidates

gets $\mathbf{Q}'$ and $\tilde{\mathbf{s}}'$ as well as the parameters p, α and δ. It outputs a list $\mathcal{L}$ of candidate solutions for Eq.(5.2). Depending on α, most of the candidate solutions $\tilde{\mathbf{e}}_1 \in \mathcal{S}$ will have exponentially many *representations* since there will be exponentially many ways of writing $\tilde{\mathbf{e}}_1 = \tilde{\mathbf{e}}_{1,1} + \tilde{\mathbf{e}}_{1,2}$ with $\tilde{\mathbf{e}}_{1,i} \in W_i$. The main property of FindCandidates is to exploit those representations in order to cleverly construct two lists $\mathcal{L}_1$ and $\mathcal{L}_2$. Intuitively, it seems reasonable to pick rather small lists $|\mathcal{L}_i| < |W_i|$ while keeping at least one representation of each candidate solution with "good" probability. The smaller the lists $\mathcal{L}_1$ and $\mathcal{L}_2$ get, the more efficient becomes the final merging procedure that yields the output list $\mathcal{L}$. The overall framework is summarised in Algorithm 3.

Algorithm 3: Generalised ISD

input : Parity check matrix $\mathbf{H}$, syndrome $\mathbf{s}$ and target weight ω.
output : Error vector $\mathbf{e}$ with $\mathbf{He} = \mathbf{s}$ and $\mathrm{wt}(\mathbf{e}) = \omega$.
params: $0 \le \ell \le n-k$, $\max\{0, k+\ell+\omega-n\} \le p \le \max\{k+\ell, \omega\}$, $\alpha \in [0, \frac{1}{2}]$ and $0 \le \delta \le 2\alpha(k+\ell-p)$.

```
found ← false;
while not found do
    (H̃, s̃, P) ← Init(H, s);
    L ← FindCandidates(p, α, δ, Q', s̃');
    forall ẽ_1 ∈ L do
        if wt(Qẽ_1 + s̃) = ω − p then
            ẽ ← (ẽ_1 ‖ (Qẽ_1 + s̃)_[ℓ+1,n−k]);
            found ← true;
return Pẽ;
```

Note that the algorithm always outputs a *correct* solution: It holds

$$\begin{aligned}\tilde{\mathbf{H}}\tilde{\mathbf{e}} &= \begin{pmatrix}\mathbf{Q}' & \mathbf{0}\\ \mathbf{Q}'' & \mathbf{I}_{n-k-\ell}\end{pmatrix}\begin{pmatrix}\tilde{\mathbf{e}}_1\\ (\mathbf{Q}\tilde{\mathbf{e}}_1+\tilde{\mathbf{s}})_{[\ell+1,n-k]}\end{pmatrix}\\ &= (\mathbf{Q}'\tilde{\mathbf{e}}_1 \;\|\; \mathbf{Q}''\tilde{\mathbf{e}}_1 + (\mathbf{Q}\tilde{\mathbf{e}}_1+\tilde{\mathbf{s}})_{[\ell+1,n-k]}) = (\tilde{\mathbf{s}}' \;\|\; \tilde{\mathbf{s}}_{[\ell+1,n-k]}) = \tilde{\mathbf{s}}\end{aligned}$$

since $\tilde{\mathbf{s}}' = \tilde{\mathbf{s}}_{[1,\ell]}$ and $(\mathbf{Q}\tilde{\mathbf{e}}_1)_{[\ell+1,n-k]} = \mathbf{Q}''\tilde{\mathbf{e}}_1$ by definition. This implies $\mathbf{He} = \mathbf{s}$ and $\mathbf{e}$ obviously has the correct weight due to $\mathrm{wt}(\mathbf{e}) = \mathrm{wt}(\mathbf{P}\tilde{\mathbf{e}}) = \mathrm{wt}(\tilde{\mathbf{e}}) = p + \omega - p = \omega$.

In order to estimate the running time, it is important to precisely estimate the number of representations, as they heavily influence both the efficiency and success probability of FindCandidates. Note that the number of representations for a given $\tilde{\mathbf{e}}_1$ will depend on its actual weight-distribution and we will call a vector (i,j)-distributed if it has Hamming weight exactly i and j on its first and last $(\frac{1}{2}-\alpha)(k+\ell)$ coordinates, respectively. Thus, if $\tilde{\mathbf{e}}_1$ is (i,j)-distributed it has weight $p-(i+j)$ within the $2\alpha(k+\ell)$ overlapping coordinates, see Figure 5.3 for an illustration. Note that in an extreme case, $\tilde{\mathbf{e}}_1$ could be $(\frac{p}{2},\frac{p}{2})$-distributed, i.e. $\tilde{\mathbf{e}}_1$ has weight 0 within the overlapping coordinates. Consequently, for $\delta = 0$, there would be no representations for this $\tilde{\mathbf{e}}_1$ at all.

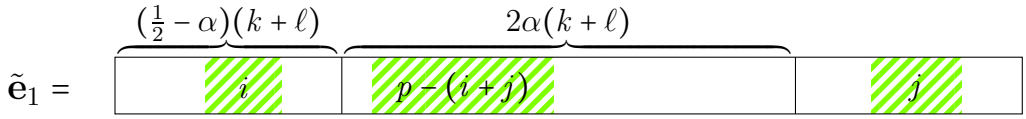


Figure 5.3: Illustration of a (i,j)-distributed vector $\tilde{\mathbf{e}}_1$ for $\alpha = \frac{1}{4}$ where the striped regions present the error positions.

To simplify the later analysis, we will restrict to those solutions $\tilde{\mathbf{e}}_1$ that come with a natural weight distribution. If $\tilde{\mathbf{e}}_1$ is a random vector of (large enough) length $k+\ell$ and weight p and one considers a subset of $(\frac{1}{2}-\alpha)(k+\ell)$ coordinates. One expects $\tilde{\mathbf{e}}_1$ to have proportional weight $(\frac{1}{2}-\alpha)p$ on the respective coordinates. This is covered by the following definition.

Definition 5.1.1. A vector $\tilde{\mathbf{e}}_1 \in W_{k+\ell,p}$ is called **balanced** if it is $\left((\frac{1}{2}-\alpha)p, (\frac{1}{2}-\alpha)p\right)$-distributed.

Note that there are no unbalanced vectors for the most important case where $\alpha = \frac{1}{2}$. For $\alpha < \frac{1}{2}$, restricting to balanced vectors further limits the set of "good" permutations. For a balanced vector (and $\alpha > 0$), there will be exponentially many representations and the exact number is easy to compute. Both the probability of guessing "good" permutations and the number of representations are *independent* of the concrete implementation of FindCandidates and will be determined next.

"Good" Permutations and the Number of Representations.

To estimate the number of iterations, we need to compute the probability of guessing a "good" permutation. In our generalised framework, a "good" permutation is defined as follows.

Definition 5.1.2. A permutation $\mathbf{P} \in \mathbb{F}_2^{n\times n}$ is called **good** (with respect to a fixed $\mathbf{e}$) if the permuted error vector $\tilde{\mathbf{e}} = \mathbf{P}^{-1}\mathbf{e}$ can be written as $\tilde{\mathbf{e}} = (\tilde{\mathbf{e}}_1, \tilde{\mathbf{e}}_2)$ where $\tilde{\mathbf{e}}_1$ is balanced and $\tilde{\mathbf{e}}_2 \in W_{n-k-\ell,\omega-p}$, see Figure 5.4 for an illustration.

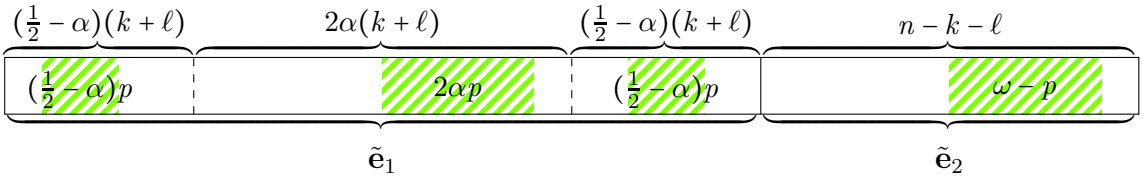


Figure 5.4: Weight distribution of a permuted error vector $\tilde{\mathbf{e}}$ resulting from a good permutation for $\alpha = \frac{1}{4}$. Striped regions represent error positions.

In our asymptotic analysis we parametrise $k = \lfloor Rn \rfloor$, $\omega = \lfloor Wn \rfloor$, $\ell = \lfloor Ln \rfloor$, $p = \lfloor Pn \rfloor$ and $\delta = \lfloor \Delta n \rfloor$. As in Section 4.1.1 we denote by $N(R, W, L, P)$ the contribution of the number of iterations to the overall complexity coefficient. This number is given by the next theorem.

Theorem 5.1.3. *Let* $0 < R < 1$ *and* $0 < W \leq \mathrm{D}_{\mathsf{GV}}(R)$. *For* $0 \leq L \leq 1 - R$ *and* $\max\{0, R + L + W - 1\} \leq P \leq \min\{R + L, W\}$ *it holds*

$$\lim_{n\to\infty} \frac{1}{n} \log N(R, W, L, P) = \mathrm{H}(W) - (R + L)\,\mathrm{H}\left(\frac{P}{R + L}\right) - (1 - R - L)\,\mathrm{H}\left(\frac{W - P}{1 - R - L}\right). \tag{5.6}$$

Proof. A permutation is good with probability

$$\frac{\binom{(\frac{1}{2}-\alpha)(k+\ell)}{(\frac{1}{2}-\alpha)p}^2 \binom{2\alpha(k+\ell)}{2\alpha p}\binom{n-k-\ell}{\omega-p}}{\binom{n}{\omega}} \tag{5.7}$$

and the results follows immediately from Corollary 2.3.4. □

Remark 5.1.4. Theorem 5.1.3 implies that restricting to permutations that yield *balanced* $\tilde{\mathbf{e}}_1$ does not affect the asymptotic behaviour of the algorithm: The least restrictive notion of a "good" permutation would be to only require $\mathrm{wt}(\tilde{\mathbf{e}}_1) = p$ and $\mathrm{wt}(\tilde{\mathbf{e}}_2) = \omega - p$. But this weakened definition would yield the same N (up to polynomial factors).

Let us now consider a balanced vector $\tilde{\mathbf{e}}_1$ and estimate the number of combinations $(\tilde{\mathbf{e}}_{1,1}, \tilde{\mathbf{e}}_{1,2}) \in W_1 \times W_2$ with $\tilde{\mathbf{e}}_1 = \tilde{\mathbf{e}}_{1,1} + \tilde{\mathbf{e}}_{1,2}$. This number will be a function of the parameters k,ℓ,p,α and δ and is denoted by $\rho(k, \ell, p, \alpha, \delta)$.

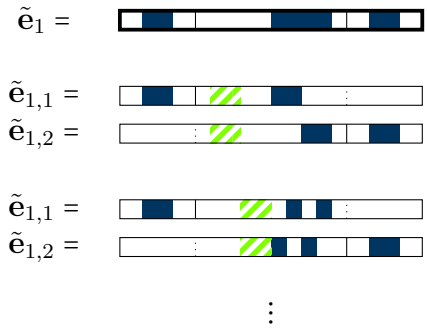


Figure 5.5: Different ways of decomposing a balanced $\tilde{\mathbf{e}}_1$. The filled regions represent error positions, the striped regions represent intersecting positions.

Lemma 5.1.5. *Let* $\tilde{\mathbf{e}}_1 \in (W_1 + W_2) \cap W$ *be balanced. Then*

$$\rho(k, \ell, p, \alpha, \delta) = \binom{2\alpha p}{\alpha p}\binom{2\alpha(k+\ell-p)}{\delta} \ . \tag{5.8}$$

Proof. The proof follows from Figure 5.5: Clearly, the $\tilde{\mathbf{e}}_{1,i}$ must have exactly δ common 1-coordinates (otherwise $\mathrm{wt}(\tilde{\mathbf{e}}_{1,1} + \tilde{\mathbf{e}}_{1,2}) \neq p$). Moreover, since $\tilde{\mathbf{e}}_1$ is balanced, $\tilde{\mathbf{e}}_{1,1}$ must have weight $(\frac{1}{2} - \alpha)p$ on its first $(\frac{1}{2} - \alpha)(k + \ell)$ coordinates (a similar statement holds for $\tilde{\mathbf{e}}_{1,2}$). Consequently, the $\tilde{\mathbf{e}}_{1,i}$ have weight $\alpha p + \delta$ on the overlapping coordinates. Thus, we can arbitrarily assign αp out of the $2\alpha p$ many 1's within the overlapping coordinates of $\tilde{\mathbf{e}}_1$ to $\tilde{\mathbf{e}}_{1,1}$ (and the remaining 1's to $\tilde{\mathbf{e}}_{1,2}$). This explains the first binomial coefficient in the above formula. Now, we can arbitrarily choose δ many intersecting positions amongst the $2\alpha(k + \ell) - 2\alpha p$ many 0-positions of $\tilde{\mathbf{e}}_1$ within the overlapping coordinates and assign a 1 to those coordinates in both $\tilde{\mathbf{e}}_{1,1}$ and $\tilde{\mathbf{e}}_{1,2}$.

□

Corollary 5.1.6. *Let* $0 < R < 1$ *and* $\alpha \in [0, \frac{1}{2}]$. *For* $0 \leq P \leq R + L$ *and* $0 \leq \Delta \leq 2\alpha(R + L - P)$ *it holds*

$$\begin{aligned} \mathfrak{P}(R, L, P, \alpha, \Delta) &:= \lim_{n\to\infty} \frac{1}{n} \log \rho(\lfloor Rn \rfloor, \lfloor Ln \rfloor, \lfloor Pn \rfloor, \alpha, \lfloor \Delta n \rfloor) \\ &= 2\alpha \left(P + (R + L - P)\, \mathrm{H}\left(\frac{\Delta}{2\alpha(R + L - P)} \right) \right) \ . \end{aligned} \tag{5.9}$$

Basic Properties of FindCandidates**.**

Let $\mathcal{B} := \{\tilde{\mathbf{e}}_1 \in \mathcal{S} : \tilde{\mathbf{e}}_1 \text{ balanced}\}$ be the set of balanced candidate solutions to Eq.(5.2). Unless $\alpha = 0$, every $\tilde{\mathbf{e}}_1 \in \mathcal{B}$ has exponentially many representations. For FindCandidates, the probability of finding a fixed $\tilde{\mathbf{e}}_1 \in \mathcal{B}$ is given by $P(\tilde{\mathbf{e}}_1) = \mathbf{Pr}[\tilde{\mathbf{e}}_1 \in \mathcal{L}_1 + \mathcal{L}_2]$. This probability is only defined over the internal randomness of FindCandidates. In particular, $P(\tilde{\mathbf{e}}_1)$ is the same for all $\tilde{\mathbf{e}}_1$ and it is independent of the initial randomisation phase. Thus, instantiating the generalised ISD framework with FindCandidates (approximately) increases the number of iterations by a factor $P(\tilde{\mathbf{e}}_1)^{-1}$. Conditioned on the event of having picked a good permutation, i.e. the wanted solution $\tilde{\mathbf{e}}_1$ is balanced, FindCandidates succeeds in finding it with probability $P(\tilde{\mathbf{e}}_1)$. This motivates the next definition.

Definition 5.1.7. We call FindCandidates **τ-solution-preserving** if

$$0 \leq \limsup_{n\to\infty} \frac{1}{n} \log P(\tilde{\mathbf{e}}_1)^{-1} = \tau \ .$$

Remark 5.1.8. Asymptotically, we are only interested in sequences with $\{P(\tilde{\mathbf{e}}_1)^{-1}\}_{n\in\mathbb{N}} \in 2^{\mathcal{O}(n)}$. Thus $\frac{1}{n} \log P(\tilde{\mathbf{e}}_1)^{-1}$ will be upper bounded and the above expression is well-defined.

Being τ-solution-preserving is thus an asymptotic property of FindCandidates. Instantiating the ISD framework with FindCandidates means that the overall coefficient for the number of iterations is increased by τ. For many concrete instantiations of FindCandidates, e.g. all sampling-based variants of Section 5.2, we have $\tau = 0$ due to a constant failure probability. For technical reasons, the improved algorithms of Chapter 6 will have $\tau > 0$ but arbitrarily small. In any case, τ will be independent of the parameters R, L, P, α and Δ.

Moreover, let $c(k, \ell, p, \alpha, \delta)$ denote the running time of FindCandidates (for almost all codes) and set $C(R, L, P, \alpha, \Delta) := \lim_{n\to\infty} \frac{1}{n} \log c(\lfloor Rn \rfloor, \lfloor Ln \rfloor, \lfloor Pn \rfloor, \alpha, \lfloor \Delta n \rfloor)$. The next lemma immediately follows.

Lemma 5.1.9. *Let $0 < R < 1$, $0 < W \leq \mathrm{D}_{\mathsf{GV}}(R)$ and $\alpha \in [0, \frac{1}{2}]$. Let* FindCandidates *be τ-solution preserving. For almost all binary $[n, \lfloor Rn \rfloor]$-codes it holds: Generalised ISD instantiated with* FindCandidates *successfully terminates in time $2^{F_{\mathsf{genISD}}(R,W)n+o(n)}$ for every error vector $\mathbf{e} \in W_{n,\lfloor Wn \rfloor}$ with overwhelming probability where*

$$F_{\mathsf{genISD}}(R, W, \alpha) = \min_{L,P,\Delta} \{N(R, W, L, P) + C(R, L, P, \alpha, \Delta) + \tau\}$$

where the minimum is taken over all valid *parameters L, P and Δ.*

Clearly, this lemma merely tells how the different building blocks of the generalised ISD framework contribute to the complexity coefficient, but remains completely meaningless without further specifying FindCandidates (which determines C, τ and the notion of being a *valid* parameter set). In the next section, we will present a purely sampling-based instantiation of FindCandidates.

5.2 Relations Between Recent Algorithms

We will now provide a first concrete class of instantiations for the generalised ISD framework serving the following purpose:

> **Goal: Relate the generalised ISD framework to Ball-collision Decoding and lay the foundation for improved algorithms.**

This will essentially be achieved by instantiating FindCandidates in the most naïve way: Simply sample two lists $\mathcal{L}_1 \in_R (W_1)^\lambda$ and $\mathcal{L}_2 \in_R (W_2)^\lambda$ by independently picking λ many random elements from W_1 and W_2, respectively (in particular we might pick some elements repeatedly and there will be $< \lambda$ distinct elements in the $\mathcal{L}_i$). The crucial point is to choose the list size $\lambda(k, \ell, p, \alpha, \delta)$ appropriately as a function of the other algorithm parameters in order to make FindCandidates 0-*solution-preserving*. Once $\mathcal{L}_1$ and $\mathcal{L}_2$ have been sampled, we need to find all collisions $(\tilde{\mathbf{e}}_{1,1}, \tilde{\mathbf{e}}_{1,2}) \in \mathcal{L}_1 \times \mathcal{L}_2$ with $\mathbf{Q}(\tilde{\mathbf{e}}_{1,1} + \tilde{\mathbf{e}}_{1,2}) = \tilde{\mathbf{s}}'$ and Hamming weight $\mathrm{wt}(\tilde{\mathbf{e}}_{1,1} + \tilde{\mathbf{e}}_{1,2}) = p$. A collision that fulfils the extra weight condition is called **weight-consistent**. In the case $\alpha = 0$ (no overlap) a collision will always be weight-consistent. For $\alpha > 0$ and $\delta = 0$ it might happen that some errors cancel out

which gives a resulting collision of weight $< p$. For the case $\delta > 0$ it is even possible to obtain vectors of weight $> p$. To obtain only weight-consistent collisions, we can still employ the MERGE-JOIN procedure as defined in Chapter 6 which simply filters out all weight-inconsistent pairs (besides the lists $\mathcal{L}_1$ and $\mathcal{L}_2$ we also provide MERGE-JOIN with the target weight p as well as $\mathbf{Q}'$ and $\tilde{\mathbf{s}}'$ which will be needed to appropriately label and further process the lists). As a consequence, the additional filtering might result in a running time T_{Merge} larger than the size of the merged output list $\mathcal{L}$. This fact will be considered in our analysis, see the next proposition.

Proposition 5.2.1. *It holds* $T_{\mathsf{Merge}} = \mathcal{O}\left(\max\{|\mathcal{L}_i|\log|\mathcal{L}_i|, M\}\right)$ *where* M *denotes the number of (not necessarily weight-consistent) matchings between* $\mathcal{L}_1$ *and* $\mathcal{L}_2$.

Remark 5.2.2. One could in principle also define the generalised framework to work for weight-inconsistent matchings. One can extend a weight-inconsistent matching $\tilde{\mathbf{e}}_1 = \tilde{\mathbf{e}}_{1,1} + \tilde{\mathbf{e}}_{1,2}$ of weight $p' \neq p$ to all $n-k$ coordinates whenever the Hamming distance of $\mathbf{Q}\tilde{\mathbf{e}}_1$ to $\tilde{\mathbf{s}}$ equals $\omega - p'$. In fact, this would increase the number of suitable error patterns that can be recovered within one iteration, nevertheless, we stick with weight-consistent matchings for the following reasons. First the beneficial effect of allowing weight-inconsistent matchings is asymptotically negligible. Second it is compatible with the Finiasz-Sendrier framework. Third, allowing for weight-inconsistent matchings would unnecessarily complicate the subsequent analysis.

For the sake of completeness, we give a pseudo-code description in Algorithm 4 and denote it by SAMPLELISTS. By APPEND we denote a function that appends an element x to a list $\mathcal{L}$ (possibly multiple times).

Algorithm 4: SAMPLELISTS

input : Parameters p, α and δ. Matrix $\mathbf{Q}' \in \mathbb{F}_2^{\ell \times k+\ell}$ and syndrome $\tilde{\mathbf{s}}' \in \mathbb{F}_2^{\ell}$.
output : List $\mathcal{L}$ of candidate solutions to Eq.(5.2).
params: Number of samples $\lambda := \lambda(k, \ell, p, \alpha, \delta)$.

$\mathcal{L}_1 \leftarrow \varnothing$; $\mathcal{L}_2 \leftarrow \varnothing$;
for $i = 1$ **to** λ **do**
  $\tilde{\mathbf{e}}_{1,1} \leftarrow_R W_1$; $\mathcal{L}_1 \leftarrow \text{APPEND}(\mathcal{L}_1, \tilde{\mathbf{e}}_{1,1})$;
  $\tilde{\mathbf{e}}_{1,2} \leftarrow_R W_2$; $\mathcal{L}_2 \leftarrow \text{APPEND}(\mathcal{L}_2, \tilde{\mathbf{e}}_{1,2})$;
$\mathcal{L} \leftarrow \text{MERGE-JOIN}(\mathcal{L}_1, \mathcal{L}_2, p, \mathbf{Q}', \tilde{\mathbf{s}}')$;
return $\mathcal{L}$;

In Section 5.2.1 we will formally prove, that setting

$$\lambda(k, \ell, p, \alpha, \delta) := \frac{|W_1|}{\sqrt{\rho(k, \ell, p, \alpha, \delta)}} \tag{5.10}$$

is sufficient to make SAMPLELISTS 0-solution-preserving by showing that a fixed balanced solution will have at least one representation in $\mathcal{L}_1 \times \mathcal{L}_2$ with *constant* probability.

Intuitively, one might justify this choice following the birthday paradox as follows: Write $\mathcal{L}_1 := \{\tilde{\mathbf{e}}_{1,1}^i : i = 1, \dots, \lambda\}$ and $\mathcal{L}_2 := \{\tilde{\mathbf{e}}_{1,2}^j : j = 1, \dots, \lambda\}$ and set $\rho := \rho(k, \ell, p, \alpha, \delta)$ and $\sigma := |W_1| = |W_2|$. Let $\tilde{\mathbf{e}}_1$ be a fixed, balanced solution. We define λ^2 random variables

$$X_{i,j} := \begin{cases} 1 & \text{if } \tilde{\mathbf{e}}_1 = \tilde{\mathbf{e}}_{1,1}^i + \tilde{\mathbf{e}}_{1,2}^j \\ 0 & \text{otherwise} \end{cases}$$

and observe that $\mathbf{Pr}[X_{i,j} = 1] = \frac{\rho}{\sigma^2}$ since there are exactly ρ choices for $\tilde{\mathbf{e}}_{1,1}^i$ and $\tilde{\mathbf{e}}_{1,2}^j$ is uniquely determined by $\tilde{\mathbf{e}}_1 + \tilde{\mathbf{e}}_{1,1}^i$ once $\tilde{\mathbf{e}}_{1,1}^i$ is fixed. Our final goal is to upper bound the probability of the event $X = 0$ where $X := \sum X_{i,j}$. If the $X_{i,j}$ were independent, this probability would be easy to compute as

$$\mathbf{Pr}[X = 0] = \mathbf{Pr}[X_{i,j} = 0 \ \ \forall 1 \le i, j \le \lambda] = \left(1 - \frac{\rho}{\sigma^2}\right)^{\lambda^2} \le e^{-1}$$

which yields the desired upper bound (where we used $1 - x \le e^{-x}$ for all $x \in \mathbb{R}$). Unfortunately, the $X_{i,j}$ are *not* independent as we will discuss next.

Definition 5.2.3. Let $\tilde{\mathbf{e}}_1$ be balanced. We call an element $\tilde{\mathbf{e}}_{1,1}$ **consistent** with $\tilde{\mathbf{e}}_1$, if $\tilde{\mathbf{e}}_1 + \tilde{\mathbf{e}}_{1,1} = \mathbf{0}$ on the first $(\frac{1}{2} - \alpha)(k + \ell)$ coordinates (and thus $\mathrm{wt}(\tilde{\mathbf{e}}_1 + \tilde{\mathbf{e}}_{1,1}) = \alpha p + \delta$ on the overlapping $2\alpha(k + \ell)$ middle coordinates), i.e.

$$\tilde{\mathbf{e}}_1 + \tilde{\mathbf{e}}_{1,1} \in W_2 \ .$$

Thus only $\tilde{\mathbf{e}}_{1,1}^i$ that are consistent to $\tilde{\mathbf{e}}_1$ can yield (the balanced) $\tilde{\mathbf{e}}_1$ by adding elements from $\mathcal{L}_2$. For $j \ne k$ it holds

$$\begin{aligned} \mathbf{Pr}[X_{i,j} = 1, X_{i,k} = 1] &= \mathbf{Pr}[X_{i,j} = 1 | X_{i,k} = 1] \, \mathbf{Pr}[X_{i,k} = 1] \\ &= \mathbf{Pr}\left[\tilde{\mathbf{e}}_{1,2}^j = \tilde{\mathbf{e}}_{1,2}^k\right] \frac{\rho}{\sigma^2} = \frac{\rho}{\sigma^3} > \frac{\rho^2}{\sigma^4} = \mathbf{Pr}[X_{i,j} = 1] \, \mathbf{Pr}[X_{i,k} = 1] \end{aligned}$$

where the second equality holds since $X_{i,k} = 1$ implies that $\tilde{\mathbf{e}}_{1,1}^i$ must be consistent and thus $X_{i,j} = 1$ iff $\tilde{\mathbf{e}}_{1,2}^j = \tilde{\mathbf{e}}_{1,2}^k$. We circumvent the obstacle of dependencies by first proving that $\mathcal{L}_1$ contains a sufficient number of *distinct, consistent* elements for every fixed $\tilde{\mathbf{e}}_1$ with some constant, non-zero probability. In this case, there will also be at least one suitable element in $\mathcal{L}_2$ that matches at least one of the distinct, consistent elements in $\mathcal{L}_1$ with constant, non-zero probability.

Remark 5.2.4. To the best of our knowledge, the dependency issue has not been properly treated in the literature so far. In [FS09], which covers the special case $\alpha = \frac{1}{2}$ and $\delta = 0$, the authors analyse the choice of λ under the assumption that "all sums $\tilde{\mathbf{e}}_{1,1}^i + \tilde{\mathbf{e}}_{1,2}^j$ are uniformly and independently distributed in $W_{k+\ell,p}$", cf. [FS09, Asmp.(I1)]. This assumption is false because for fixed i and varying $j = 1, \dots, \lambda$, all the pairs $(\tilde{\mathbf{e}}_{1,1}^i, \tilde{\mathbf{e}}_{1,2}^j)$ share a lot of non-zero coordinates. Similarly, Peters provides a proof sketch in [Pet11, Section5.3.2] which implicitly assumes independence of the random variables $X_{i,j}$.

For the moment let us heuristically assume that the above analysis is valid. We will employ SampleLists with λ as defined in Eq.(5.10) for different cases:

- Section 5.2.2 covers $\alpha = 0$ and $\delta = 0$. In this case every solution has a unique representation, i.e. $\rho = 1$ which yields $\lambda = |W_1|$. Thus, sampling $\mathcal{L}_1$ might be replaced by *deterministically* setting $\mathcal{L}_1 = W_1$. We will show that this instantiation is asymptotically as good as BCD and PSSD, as introduced in Section 4.2.

- Section 5.2.4 covers $\alpha > 0$ and $\delta = 0$. For the extreme case $\alpha = \frac{1}{2}$ this resembles the algorithm of [FS09]. We will show that $\alpha > 0$ is always suboptimal for $\delta = 0$, see Theorem 5.2.24.

- Section 5.2.5 extends to $\delta > 0$. The extreme case $\alpha = \frac{1}{2}$ will serve as a starting point for our new algorithm in Section 6.2. The main technical result is a parameter transformation that maps parameters (p, ℓ) for $\alpha = 0$ and $\delta = 0$ to parameters (p', ℓ', δ') for $\alpha = \frac{1}{2}$ while preserving the asymptotic complexity of the algorithm.

The above relations are summarized in Figure 5.6. Thus, we can start from optimal parameters of SampleLists with $\alpha = \delta = 0$ and the resulting running time will always be at least as good as the optimal running time for BCD due to Theorem 5.2.13. We can then apply Theorem 5.2.26 yielding a sampling-based algorithm with parameters (p', ℓ', δ') for $\alpha = \frac{1}{2}$ that achieves the same performance as SampleLists for $\alpha = \delta = 0$ and is thus at least as efficient as BCD. The same parameters can finally be used in our new algorithm while preserving the running time. To prove the superiority of our algorithm compared to BCD, it is then sufficient to slightly adapt these parameters.

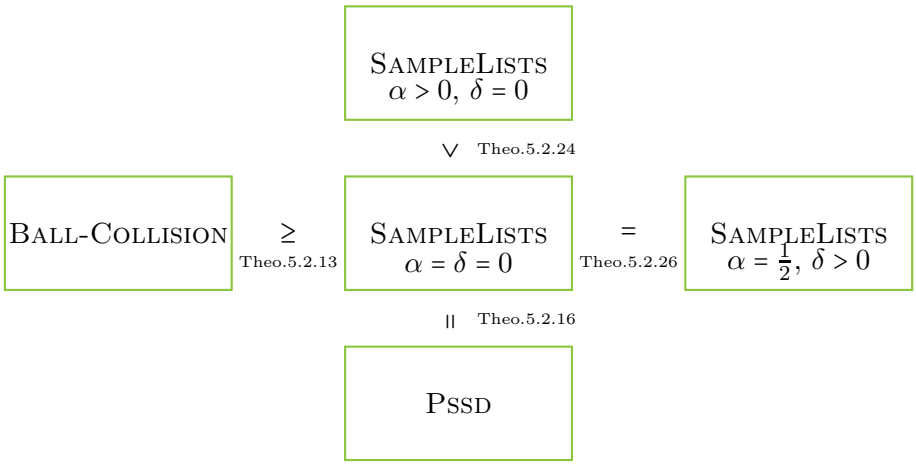


Figure 5.6: Relations between known algorithms and different ISD instantiations using SampleLists. The $\geq$ or = symbols together with the referenced theorems indicate the asymptotic relationships.

5.2.1 Choosing λ

Our aim is to prove that the heuristic choice of λ in Eq.(5.10) does indeed give a solution-preserving instantiation of SampleLists. In the following we write $\rho := \rho(k, \ell, p, \alpha, \delta)$ for the number of representations as defined in Lemma 5.1.5 and

$$\sigma := |W_1| = |W_2| = \binom{\left(\frac{1}{2} + \alpha\right)(k + \ell)}{\frac{p}{2} + \delta} . \tag{5.11}$$

Theorem 5.2.10 (Informal). *For sufficiently large ρ and σ and*

$$\lambda := \frac{\sigma}{\sqrt{\rho}}$$

SampleLists *is 0-solution-preserving.*

The outline of the proof is as follows: Fix a balanced solution $\tilde{\mathbf{e}}_1$ and first show that there will be a sufficiently large number of *distinct, consistent* elements (for $\tilde{\mathbf{e}}_1$) in $\mathcal{L}_1$ with good probability. Then upper bound the probability that none of the elements in $\mathcal{L}_2$ matches any of those "good" elements in $\mathcal{L}_1$. The first part of the proof requires some additional lemmata: We first provide a lower bound on the probability that $\mathcal{L}_1$ contains at least $\frac{\lambda}{2}$ many *distinct* elements. This follows from Theorem 5.2.6 which gives a concentration result for the expected size of the image of a random function $f : [n] \to [m]$. We define a random variable $X_{n,m} := |\mathrm{im}(f)|$ which is a function on the product space $[m]^n$ (equipped with the uniform distribution), i.e. we identify a random function f with its table of values $(f(1), \ldots, f(n))$. The study of $X_{n,m}$ is a typical application of the *Probabilistic Method* [AS08].

Lemma 5.2.5. *It holds*

$$\mathbb{E}[X_{n,m}] = m\left(1 - \left(1 - \frac{1}{m}\right)^n\right) \tag{5.12}$$

which is monotone increasing in n and m.

Proof. $X_{n,m}$ can be written as the sum of m indicator variables I_j where $I_j = 1$ iff $j \in \mathrm{im}(f)$. It holds

$$\mathbb{E}[I_j] = \mathbf{Pr}[I_j = 1] = 1 - \left(1 - \frac{1}{m}\right)^n$$

which proves Eq.(5.12) by linearity of expectation. Clearly, $\mathbb{E}[X_{n,m}]$ is monotone increasing in n. For m, consider the first partial derivative

$$\begin{aligned}\frac{\partial \mathbb{E}[X_{n,m}]}{\partial m} &= 1 - \left(1 - \frac{1}{m}\right)^n - \frac{n}{m}\left(1 - \frac{1}{m}\right)^{n-1} \\ &= 1 - \left(1 - \frac{1}{m}\right)^n \left(1 - \frac{n}{m-1}\right) .\end{aligned}$$

Since $(1-\frac{1}{m})^n \geq 0$ for all $n, m \in \mathbb{N}$ we can assume wlog that $n < m-1$ ($n \geq m-1$ implies $1-\frac{n}{m-1} \leq 0$ and the right-hand side will always be > 0). Thus, it is sufficient to show

$$\left(1-\frac{1}{m}\right)^n < \left(1-\frac{n}{m-1}\right)^{-1}$$

for all m and $n < m-1$: Note that $(1-\frac{1}{m})^n \leq 1$. Furthermore, $0 < n < m-1$ implies $0 < 1-\frac{n}{m-1} < 1$ which yields $(1-\frac{n}{m-1})^{-1} > 1$. □

Theorem 5.2.6.

$$\mathbf{Pr}\left[X_{n,m} \leq \mathbb{E}[X_{n,m}] - \sqrt{n}\ \right] \leq e^{-1/2} \tag{5.13}$$

Proof. Follows from general concentrations results used in the *Probabilistic Method*, cf. [AS08, Chapter7] or more concretely [MV08, Theorem 8.1.1]. □

To apply Theorem 5.2.6 to our setting, observe that uniformly independently picking λ out of σ elements with replacement is equivalent to picking a random function $f : [\lambda] \to [\sigma]$, i.e. the number of distinct elements in $\mathcal{L}_1$ is exactly $X_{\lambda,\sigma}$ and we obtain the following result.

Lemma 5.2.7. *For $\lambda \geq 58$ it holds*

$$\mathbf{Pr}\left[|\tilde{\mathcal{L}}_1| \geq \frac{\lambda}{2}\right] \geq 1 - e^{-1/2} \tag{5.14}$$

where $\tilde{\mathcal{L}}_1$ is the set that results from filtering out duplicate elements in $\mathcal{L}_1$.

Proof. For $n = \lambda$ and $m = \sigma$, Theorem 5.2.6 gives

$$\mathbf{Pr}\left[X_{\lambda,\sigma} \geq \mathbb{E}[X_{\lambda,\sigma}] - \sqrt{\lambda}\right] \geq 1 - e^{-1/2}\ .$$

By definition of λ it always holds $\lambda \leq \sigma$ and by monotonicity of $\mathbb{E}[X_{\lambda,\sigma}]$ it follows, that with probability at least $1 - e^{-1/2}$

$$\begin{aligned} |\tilde{\mathcal{L}}_1| = X_{\lambda,\sigma} &\geq \mathbb{E}[X_{\lambda,\sigma}] - \sqrt{\lambda} \geq \mathbb{E}[X_{\lambda,\lambda}] - \sqrt{\lambda} \\ &= \lambda\left(1 - \left(1-\frac{1}{\lambda}\right)^{\lambda}\right) - \sqrt{\lambda} \geq \lambda(1-e^{-1}) - \sqrt{\lambda} \geq \frac{\lambda}{2} \end{aligned}$$

where the last inequality holds for $\lambda \geq \left(\frac{1}{2} - e^{-1}\right)^{-2} = 57.29$. □

For a fixed, balanced solution $\tilde{\mathbf{e}}_1$ we will now consider the corresponding set $\mathcal{C}$ of distinct, consistent elements in $\mathcal{L}_1$, i.e.

$$\mathcal{C}(\tilde{\mathbf{e}}_1) := \{\tilde{\mathbf{e}}_{1,1} \in \tilde{\mathcal{L}}_1\ :\ \tilde{\mathbf{e}}_{1,1} + \tilde{\mathbf{e}}_1 \in W_2\}\ ,$$

and study its cardinality $C := |\mathcal{C}|$. Since the elements of $\tilde{\mathcal{L}}_1$ are still uniformly distributed in W_1, we expect $C \approx \frac{\rho}{\sigma}|\tilde{\mathcal{L}}_1|$ since a single element in $\tilde{\mathcal{L}}_1$ is consistent with $\tilde{\mathbf{e}}_1$ with probability $\frac{\rho}{\sigma}$. Since $\tilde{\mathcal{L}}_1$ contains only *distinct* elements, the probability of two elements being consistent to $\tilde{\mathbf{e}}_1$ is not independent. Fortunately, their covariance will be negative (if one element is consistent, the chance of being consistent for another, distinct element decreases and vice versa). This will allow to prove that there are "enough" consistent elements in $\tilde{\mathcal{L}}_1$ which eventually allows to show that there will be at least one matching element in $\mathcal{L}_2$ with good probability. The formal statement and its proof is given next.

Lemma 5.2.8. *Let $\tilde{\mathbf{e}}_1$ be a balanced. For $\rho \geq 256$, $\lambda \geq 58$ and $\lambda = \frac{\sigma}{\sqrt{\rho}}$ it holds*

$$\mathbf{Pr}\left[\tilde{\mathbf{e}}_1 \in \mathcal{L}_1 + \mathcal{L}_2\right] \geq \frac{1}{2}(1 - e^{-1/4})(1 - e^{-1/2}) \geq 0.044 \ . \tag{5.15}$$

Proof. Note that $\tilde{\mathbf{e}}_1 \in \mathcal{L}_1 + \mathcal{L}_2$ iff $\tilde{\mathbf{e}}_1 \in \mathcal{C}(\tilde{\mathbf{e}}_1) + \mathcal{L}_2$. Equivalently, there must be at least one $\tilde{\mathbf{e}}_{1,2} \in \mathcal{L}_2$ that is contained in the set $\tilde{\mathbf{e}}_1 + \mathcal{C}(\tilde{\mathbf{e}}_1)$. Note that $\tilde{\mathbf{e}}_1 + \mathcal{C}(\tilde{\mathbf{e}}_1)$ is a subset of W_2 with cardinality C. Now, all the $\tilde{\mathbf{e}}_{1,2}^j$ are uniformly and independently drawn from W_2. This allows to bound the target probability as the product of the λ probabilities $\mathbf{Pr}\left[\tilde{\mathbf{e}}_{1,2}^j \notin \tilde{\mathbf{e}}_1 + \mathcal{C}(\tilde{\mathbf{e}}_1)\right]$ where we write $\mathcal{L}_2 := \{\tilde{\mathbf{e}}_{1,2}^j \ : \ j = 1, \dots \lambda\}$. To lower bound these probabilities we need to lower bound C.

By Lemma 5.2.7, we know that $|\tilde{\mathcal{L}}_1| \geq \frac{\lambda}{2}$ for $\lambda \geq 58$ with probability $\geq 1 - e^{-1/2}$. Let us assume the worst-case $|\tilde{\mathcal{L}}_1| = \frac{\lambda}{2}$. In this case, we write $\tilde{\mathcal{L}}_1 := \{\tilde{\mathbf{e}}_{1,1}^i \ : \ i = 1, \dots, \frac{\lambda}{2}\}$ and $C = \sum C_i$ for the $\frac{\lambda}{2}$ indicator variables

$$C_i = \begin{cases} 1 & \text{if } \tilde{\mathbf{e}}_{1,1}^i \in \mathcal{C}(\tilde{\mathbf{e}}_1) \\ 0 & \text{otherwise} \ . \end{cases}$$

Note that $\mathbb{E}[C_i] = \mathbf{Pr}[C_i = 1] = \frac{\rho}{\sigma}$ and thus $\mathbb{E}[C] = \frac{\lambda\rho}{2\sigma} = \frac{\sqrt{\rho}}{2}$ by definition of λ. Furthermore, for $i \neq j$ we have

$$\begin{aligned} \mathrm{Cov}[C_i, C_j] &= \mathbb{E}[C_i C_j] - \mathbb{E}[C_i]\mathbb{E}[C_j] \\ &= \mathbf{Pr}[C_i = 1 | C_j = 1]\mathbf{Pr}[C_j = 1] - \mathbf{Pr}[C_i = 1]^2 = \left(\frac{\rho - 1}{\sigma}\right)\frac{\rho}{\sigma} - \frac{\rho^2}{\sigma^2} < 0 \end{aligned}$$

which implies

$$\mathrm{Var}[C] = \sum_{i=1}^{\lambda/2} \mathrm{Var}[C_i] + 2 \sum_{1 \leq i < j \leq \frac{\lambda}{2}} \mathrm{Cov}[C_i, C_j] < \sum_{i=1}^{\lambda/2} \mathrm{Var}[C_i] \leq \mathbb{E}[C] \tag{5.16}$$

where the last inequality is due to the fact $\mathrm{Var}[C_i] = \frac{\rho}{\sigma}(1 - \frac{\rho}{\sigma}) \leq \mathbb{E}[C_i]$. Using Chebychev's inequality, see Eq.(2.9), we obtain

$$\begin{aligned} \mathbf{Pr}\left[C \geq \frac{1}{2}\mathbb{E}[C]\right] &\geq 1 - \mathbf{Pr}\left[|C - \mathbb{E}[C]| \geq \frac{1}{2}\mathbb{E}[C]\right] \\ &\geq 1 - \frac{4\,\mathrm{Var}[C]}{\mathbb{E}[C]^2} \geq 1 - \frac{4}{\mathbb{E}[C]} = 1 - \frac{8}{\sqrt{\rho}} \geq \frac{1}{2} \end{aligned} \tag{5.17}$$

for $\rho \geq 256$. Here, the second-last inequality holds due to Eq.(5.16) and the last equality holds by definition of λ. Define the failure event $\mathsf{F} := \{\mathcal{L}_2 \cap (\tilde{\mathbf{e}}_1 + \mathcal{C}(\tilde{\mathbf{e}}_1)) = \varnothing\}$ and the success event $\mathsf{S} = \neg\mathsf{F}$. It holds

$$\mathbf{Pr}[\mathsf{F}] = \left(1 - \frac{C}{\sigma}\right)^{\lambda} \leq \exp\left(-\frac{\lambda C}{\sigma}\right)$$

and combining this with Lemma 5.2.7 and Eq.(5.17) finally yields

$$\begin{aligned}\mathbf{Pr}[\mathsf{S}] &\geq \mathbf{Pr}\left[\mathsf{S} \mid C \geq \frac{\lambda\rho}{4\sigma}\right]\left(\frac{1 - e^{-1/2}}{2}\right) \\ &= \left(1 - \mathbf{Pr}\left[\mathsf{F} \mid C \geq \frac{\lambda\rho}{4\sigma}\right]\right)\left(\frac{1 - e^{-1/2}}{2}\right) \geq \frac{(1 - e^{-1/4})(1 - e^{-1/2})}{2}\end{aligned}$$

since $\exp\left(-\frac{\lambda^2\rho}{4\sigma^2}\right) \leq e^{-1/4}$ by definition of λ. □

Remark 5.2.9. The bound provided by Lemma 5.2.8 is rather loose and the conditions on ρ and λ are quite restrictive yet sufficient for our purposes.

Being 0-*solution-preserving* is an asymptotic property of SampleLists and hence we must translate the above lemma into the asymptotic setting. Define

$$\begin{aligned}\Sigma(R, L, P, \alpha, \Delta) &:= \lim_{n\to\infty} \frac{1}{n} \log \sigma(\lfloor Rn \rfloor, \lfloor Ln \rfloor, \lfloor Pn \rfloor, \alpha, \lfloor \Delta n \rfloor) \\ &= \left(\frac{1}{2} + \alpha\right)(R + L)\,\mathrm{H}\left(\frac{\frac{P}{2} + \Delta}{(\frac{1}{2} + \alpha)(R + L)}\right)\end{aligned} \tag{5.18}$$

and the coefficient for the size of the sampled lists as

$$\begin{aligned}\Lambda(R, L, P, \alpha, \Delta) &:= \lim_{n\to\infty} \frac{1}{n} \log \lambda(\lfloor Rn \rfloor, \lfloor Ln \rfloor, \lfloor Pn \rfloor, \alpha, \lfloor \Delta n \rfloor) \\ &= \Sigma(R, L, P, \alpha, \Delta) - \frac{1}{2}\mathfrak{P}(R, L, P, \alpha, \Delta).\end{aligned} \tag{5.19}$$

where $\mathfrak{P}(R, L, P, \alpha, \Delta) = \lim_{n\to\infty} \frac{1}{n} \log \rho$ denotes the coefficient for the number of representations as defined in Corollary 5.1.6. Note that $\frac{1}{2}\mathfrak{P} < \Sigma$ always holds and thus the restrictions on λ and ρ in Lemma 5.2.8 can be simultaneously fulfilled for sufficiently large n. Furthermore, *any* balanced $\tilde{\mathbf{e}}_1 \in W_{k+\ell,p}$ (and particularly any balanced solution $\tilde{\mathbf{e}}_1 \in \mathcal{S}$) is contained in $\mathcal{L}_1 + \mathcal{L}_2$ even with *constant* probability. This allows to prove the next theorem.

Theorem 5.2.10. *Let* $0 < R < 1$ *and* $\alpha \in [0, \frac{1}{2}]$. *For* $0 \leq L \leq 1 - R$, $0 \leq P \leq R + L$, $0 \leq \Delta \leq 2\alpha(R + L - P)$ *and* $\mathfrak{P}$, Σ *and* Λ *as defined in Eq.(5.9), (5.18) and (5.19),* SampleLists *is* 0-*solution-preserving. Moreover, for almost all binary* $[n, \lfloor Rn \rfloor]$-*codes,* SampleLists *runs in time* $2^{C(R,L,P,\alpha,\Delta)n + o(n)}$ *with*

$$C(R, L, P, \alpha, \Delta) = \max\{\Lambda(R, L, P, \alpha, \Delta)\ ,\ 2\Lambda(R, L, P, \alpha, \Delta) - L\}\ .$$

As we will shortly see, the proof of Theorem 5.2.10 upper bounds the probability of obtaining too many collisions between $\mathcal{L}_1$ and $\mathcal{L}_2$ by a standard argument about the concentration of a particular random variable. As similarly done in the proof of Theorem 4.3.1, we can simply abort SampleList whenever too many collisions occur. We refrain from a detailed description. Altogether, combining Theorem 5.2.10 with Lemma 5.1.9 allows to prove the following main result.

Main Theorem 5.2.1 (SampleLists). *Let $0 < R < 1$, $0 < W \leq \mathrm{D}_{\mathsf{GV}}(R)$ and $\alpha \in [0, \frac{1}{2}]$. For almost all binary $[n, \lfloor Rn \rfloor]$-codes it holds: Generalised ISD instantiated with* SampleLists *successfully terminates in time $2^{F_{\mathsf{Sampling}}(R,W,\alpha,\Delta)n+\mathrm{o}(n)}$ for every error vector $\mathbf{e} \in W_{n,\lfloor Wn \rfloor}$ with overwhelming probability where*

$$F_{\mathsf{Sampling}}(R, W, \alpha) = \min_{P,L,\Delta} \{N(R, W, L, P) + \max\{\Lambda(R, L, P, \alpha, \Delta), 2\Lambda(R, L, P, \alpha, \Delta) - L\}\} \tag{5.20}$$

with $0 \leq L \leq 1 - R$, $\max\{0, R + L + W - 1\} \leq P \leq \min\{R + L, W\}$ and $0 \leq \Delta \leq 2\alpha(R + L - P)$.

Proof of Theorem 5.2.10. It only remains to prove the statement about the running time C. By Proposition 5.2.1, we need to estimate the number of matchings M that occur when merging $\mathcal{L}_1$ and $\mathcal{L}_2$, i.e. we need to estimate the number of pairs $(\tilde{\mathbf{e}}_{1,1}, \tilde{\mathbf{e}}_{1,2}) \in \mathcal{L}_1 \times \mathcal{L}_2$ with $\mathbf{Q}'(\tilde{\mathbf{e}}_{1,1} + \tilde{\mathbf{e}}_{1,2}) = \tilde{\mathbf{s}}'$. We proceed similarly to the proof of Theorem 4.3.1 for BCD, i.e. we apply Lemma 2.4.6 to obtain

$$\mathbb{E}[M] = 2^{(2\Lambda - L)n + \mathrm{o}(n)}$$

and $\mathrm{Var}[M] \leq \mathbb{E}[M]$. Suppose $L < 2\Lambda$. Using Chebychev's inequality, see Eq.(2.9), we obtain

$$\mathbf{Pr}[M \geq 2\mathbb{E}[M]] \leq \mathbf{Pr}[|M - \mathbb{E}[M]| \geq \mathbb{E}[M]] \leq \frac{\mathrm{Var}[M]}{\mathbb{E}[M]^2} \leq \frac{1}{\mathbb{E}[M]} = 2^{-(2\Lambda - L)n + \mathrm{o}(n)} \ ,$$

which is exponentially decreasing in n, i.e. $M \leq 2\mathbb{E}[M] = 2^{(2\Lambda - L)n + \mathrm{o}(n)}$ with probability $1 - 2^{-\Omega(n)}$. Conversely, for $L \geq 2\Lambda$ we have $\mathbb{E}[M] \leq 2^{\mathrm{o}(n)}$ and the following very vague estimation is sufficient: Let $\epsilon > 0$ be arbitrarily small, it holds

$$\mathbf{Pr}[M \geq 2^{\epsilon n} + \mathbb{E}[M]] \leq \mathbf{Pr}[|M - \mathbb{E}[M]| \geq 2^{\epsilon n}] \leq 2^{-2\epsilon n + \mathrm{o}(n)} \ ,$$

i.e. $M \leq 2^{\epsilon n + \mathrm{o}(n)}$ for $\epsilon > 0$ arbitrarily small with probability $1 - 2^{-\Omega(n)}$. By Proposition 5.2.1 we finally obtain $C = \max\{\Lambda, 2\Lambda - L\}$ for ϵ sufficiently small (in the case $\Lambda = 0$ there is nothing to prove).

□

5.2.2 A Variant of Ball-collision Decoding

We now study the instantiation of the generalised ISD framework for $\alpha = \delta = 0$. We call this variant FS-ISD since the only difference to Stern's algorithm is to use the quasi-standard form as first proposed by Finiasz and Sendrier in [FS09]. The main goal is to prove that

FS-ISD is at least as efficient as BCD.

As mentioned before, for $\alpha = \delta = 0$ there are *no* representations involved. This gives $\rho = 1$ and we obtain $\lambda = |W_1|$. It is thus reasonable to replace SampleLists by a *deterministic* procedure that simply computes $\mathcal{L}_1 = W_1$ and $\mathcal{L}_2 = W_2$. In this case, *every* balanced vector $\tilde{\mathbf{e}}_1 \in W_{k+\ell,p}$ is by definition in $\mathcal{L}_1 + \mathcal{L}_2 = W_1 + W_2$ and SampleLists is trivially 0-solution-preserving (with probability 1). In particular, Theorem 5.2.1 still holds for the deterministic variant of SampleLists and we obtain an algorithm with complexity coefficient

$$F_{\mathsf{FS-ISD}}(R, W) = \min_{P,L}\{N_{\mathsf{FS-ISD}} + \max\{\Lambda_{\mathsf{FS-ISD}} \ , \ 2\Lambda_{\mathsf{FS-ISD}} - L\}\} \tag{5.21}$$

where $N_{\mathsf{FS-ISD}} := N(R, W, L, P)$ denotes the number of iterations as defined in Eq.(5.6) and

$$\Lambda_{\mathsf{FS-ISD}}(R, L, P) = \frac{R+L}{2}\,\mathrm{H}\!\left(\frac{P}{R+L}\right) \tag{5.22}$$

denotes the space complexity given by the size of the lists $\mathcal{L}_1$ and $\mathcal{L}_2$. The parameter space is restricted to

$$0 \le \ L \le 1 - R \tag{5.23}$$

$$\max\{0, R + L + W - 1\} \le \ P \le \min\{R + L, W\} \ . \tag{5.24}$$

Remark 5.2.11. The relation

$$N(R, W, L, P) = \mathrm{H}(W) - (1 - R - L)\,\mathrm{H}\!\left(\frac{W - P}{1 - R - L}\right) - 2\Lambda_{\mathsf{FS-ISD}}(R, L, P) \tag{5.25}$$

sometimes allows to simplify calculations, i.e. we may write

$$T(R, W, L, P) = \mathrm{H}(W) - (1 - R - L)\,\mathrm{H}\!\left(\frac{W - P}{1 - R - L}\right) + \max\{-L, -\Lambda_{\mathsf{FS-ISD}}(R, L, P)\}$$

for the running time of FS-ISD for arbitrary valid parameters (L, P).

Remark 5.2.12. Note that $F_{\mathsf{FS-ISD}} > 0$ for all $0 < W \le \mathrm{D}_{\mathsf{GV}}(R)$ and $0 < R < 1$: For valid parameters (L, P) one has either $N(R, W, L, P) > 0$ (and thus $F > 0$) or it holds $N(R, W, L, P) = 0$. In this case, P must be proportional to $R + L$, i.e. $P = (R + L)W$. But this implies $\Lambda(L, P) = \frac{R+L}{2}\,\mathrm{H}(W) > 0$.

The following theorem, first presented in [MMT11] at Asiacrypt 2011, proves how to transform (optimal) BCD parameters (P, Q, L) into valid parameters (P', L') for FS-ISD. In the remainder of this section, we will denote the running time coefficient of BCD and FS-ISD by $T_{\mathsf{BCD}}(P, Q, L)$ and $T_{\mathsf{FS-ISD}}(P', L')$, respectively. Here, (P, Q, L) and (P', L') denote some arbitrary valid (but not necessarily optimal) parameters for BCD and FS-ISD. We drop the parameters R and W for ease of presentation.

Theorem 5.2.13. *Let (P, Q, L) be a parameter set for the BCD algorithm. Then $(P + Q, L)$ is a parameter set for FS-ISD and it holds*

$$T_{\mathsf{BCD}}(P, Q, L) \geq T_{\mathsf{FS-ISD}}(P + Q, L) \ ,$$

i.e. FS-ISD is asymptotically at least as efficient as BCD.

Remark 5.2.14. In [BLP11a], the authors state *"Our own assessment is that if parameters had been chosen more carefully then the algorithm of [32] would have led to an exponential improvement over collision decoding, contrary to the conclusions in [32]."* where [32] is [FS09] and collision decoding refers to Stern's algorithm. However, the authors do not provide any further (formal) justification of this claim comparable to Theorem 5.2.13.

Proof of Theorem 5.2.13. Let (P, Q, L) be a valid parameter set for BCD, i.e. $0 \leq L \leq 1 - R$, $0 \leq Q \leq \min\{L, W\}$ and $\max\{0, R + L + W - Q - 1\} \leq P \leq \min\{R, W - Q\}$. Then $(P', L') := (P + Q, L)$ is a valid parameter set for FS-ISD since $0 \leq L' \leq 1 - R$ and

$$\begin{aligned} P' &= P + Q \leq \min\{R + L, W\} = \min\{R + L', W\} \ , \\ P' &= P + Q \geq \max\{0, R + W + L - 1\} = \max\{0, R + W + L' - 1\} \end{aligned}$$

as required. First observe that the statement trivially holds (with equality) for $L = 0$ (since $Q = 0$ follows and both algorithms are the same). Recall from Section 4.3 that

$$T_{\mathsf{BCD}}(P, Q, L) = N_{\mathsf{BCD}}(P, Q, L) + \max\{\Lambda_{\mathsf{BCD}}(P, Q, L), 2\Lambda_{\mathsf{BCD}}(P, Q, L) - L\} \qquad (5.26)$$

where

$$N_{\mathsf{BCD}}(P, Q, L) = \ \mathrm{H}(W) - R\,\mathrm{H}\left(\frac{P}{R}\right) - L\,\mathrm{H}\left(\frac{Q}{L}\right) - (1 - R - L)\,\mathrm{H}\left(\frac{W - P - Q}{1 - R - L}\right)$$

and

$$\Lambda_{\mathsf{BCD}}(P, Q, L) = \frac{R}{2}\,\mathrm{H}\left(\frac{P}{R}\right) + \frac{L}{2}\,\mathrm{H}\left(\frac{Q}{L}\right) \ .$$

Let us first assume that the maximum in Eq.(5.26) is $2\Lambda_{\mathsf{BCD}} - L$, i.e. $L \leq \Lambda_{\mathsf{BCD}}$ must hold. Using Lemma 2.3.2 we obtain

$$\Lambda_{\mathsf{BCD}}(P, Q, L) \leq \frac{R + L}{2}\,\mathrm{H}\left(\frac{P + Q}{R + L}\right) = \Lambda_{\mathsf{FS-ISD}}(P + Q, L)$$

by definition of $\Lambda_{\mathsf{FS-ISD}}$, see Eq.(5.22), and thus $L \leq \Lambda_{\mathsf{FS-ISD}}$ follows. Consequently, the maximum in $T_{\mathsf{FS-ISD}}$, see Eq.(5.21), is also reached in the right component $2\Lambda_{\mathsf{FS-ISD}} - L$. By definition of N_{BCD} and $N_{\mathsf{FS-ISD}}$ we can now deduce

$$\begin{aligned} T_{\mathsf{FS-ISD}}(P+Q,L) &= N_{\mathsf{FS-ISD}}(P+Q,L) + 2\Lambda_{\mathsf{FS-ISD}}(P+Q,L) - L \\ &= \mathrm{H}(W) - L - (1-R-L)\,\mathrm{H}\left(\frac{W-(P+Q)}{1-R-L}\right) \\ &= N_{\mathsf{BCD}}(P,Q,L) + 2\Lambda_{\mathsf{BCD}}(P,Q,L) - L = T_{\mathsf{BCD}}(P,Q,L) \ . \end{aligned}$$

It remains to consider the case where $L > \Lambda_{\mathsf{BCD}}$. Let us first assume $L > \Lambda_{\mathsf{FS-ISD}}$ also holds, i.e. the maximum in $T_{\mathsf{FS-ISD}}$ is $\Lambda_{\mathsf{FS-ISD}}$. It follows

$$\begin{aligned} T_{\mathsf{FS-ISD}}(P+Q,L) &= N_{\mathsf{FS-ISD}}(P+Q,L) + \Lambda_{\mathsf{FS-ISD}}(P+Q,L) \\ &= \mathrm{H}(W) - (1-R-L)\,\mathrm{H}\left(\frac{W-(P+Q)}{1-R-L}\right) - \frac{R+L}{2}\,\mathrm{H}\left(\frac{P+Q}{R+L}\right) \\ &\leq \mathrm{H}(W) - (1-R-L)\,\mathrm{H}\left(\frac{W-P-Q}{1-R-L}\right) - \frac{R}{2}\,\mathrm{H}\left(\frac{P}{R}\right) - \frac{L}{2}\,\mathrm{H}\left(\frac{Q}{L}\right) \\ &= N_{\mathsf{BCD}}(P,Q,L) + \Lambda_{\mathsf{BCD}}(P,Q,L) = T_{\mathsf{BCD}}(P,Q,L) \end{aligned}$$

where we used Lemma 2.3.2 for the inequality. Contrary, let $L < \Lambda_{\mathsf{FS-ISD}}$, i.e. the maximum in $T_{\mathsf{FS-ISD}}$ is $2\Lambda_{\mathsf{FS-ISD}} - L$. Since $L > \Lambda_{\mathsf{BCD}}$, we obtain

$$\begin{aligned} T_{\mathsf{FS-ISD}}(P+Q,L) &= N_{\mathsf{FS-ISD}}(P+Q,L) + 2\Lambda_{\mathsf{FS-ISD}}(P+Q,L) - L \\ &= \mathrm{H}(W) - (1-R-L)\,\mathrm{H}\left(\frac{W-(P+Q)}{1-R-L}\right) - L \\ &< \mathrm{H}(W) - (1-R-L)\,\mathrm{H}\left(\frac{W-(P+Q)}{1-R-L}\right) - \Lambda_{\mathsf{BCD}}(P,Q,L) \\ &= N_{\mathsf{BCD}}(P,Q,L) + \Lambda_{\mathsf{BCD}}(P,Q,L) = T_{\mathsf{BCD}}(P,Q,L) \ . \end{aligned}$$

Thus, in all cases we have $T_{\mathsf{FS-ISD}}(P+Q,L) \leq T_{\mathsf{BCD}}(P,Q,L)$ as claimed. □

Remark 5.2.15. We point out that the proof of Lemma C1 in the full version of [BLP11a] implicitly shows that $L^* \leq \Lambda_{\mathsf{BCD}}(L^*,P^*,Q^*)$ holds for optimal BCD parameters L^*,P^* and Q^*. In this case, equality always holds in Theorem 5.2.13.

Relation to PSSD

As a side note, we will now relate FS-ISD to PSDD as defined in Section 4.2. First, we give a more detailed description of PSSD and generalize its complexity coefficient to arbitrary target weight $W \leq \mathrm{D}_{\mathsf{GV}}(R)$. Recall the main idea of PSSD: Given a code $\mathcal{C}$ of dimension n and rate R and vector $\mathbf{x} = \mathbf{m}^\top\mathbf{G} + \mathbf{e}$ where $\mathbf{e}$ has Hamming weight Wn, choose two parameters $R \leq \beta \leq 1$ and $\max\{0, W+\beta-1\} \leq \alpha \leq \min\{W,\beta\}$ and consider the *punctured* code $\mathcal{C}'$ given by the first βn coordinates. If $\mathcal{C}$ has generator matrix $\mathbf{G} = \begin{pmatrix}\mathbf{G}' & \mathbf{G}''\end{pmatrix}$ where $\mathbf{G}'$ consists of the first βn columns of $\mathbf{G}$, then $\mathcal{C}'$ is generated by $\mathbf{G}'$. Clearly, $\mathcal{C}'$ has length $n' = \beta n$ and rate $R' = \frac{k}{n'} = \frac{R}{\beta}$. Now, PSSD works as follows:

- Find all codewords $\mathbf{c}' \in \mathcal{C}'$ within distance αn from $\mathbf{x}'$. Therefore compute all solutions $\mathbf{e}'$ of weight αn to the equation

$$\mathbf{H}'\mathbf{e}' = \mathbf{s}'$$

  where $\mathbf{s}' = \mathbf{H}'\mathbf{x}'$ is the syndrome of the punctured received word $\mathbf{x}'$ and $\mathbf{H}'$ denotes a parity check matrix of $\mathcal{C}'$. This is done by splitting $\mathbf{H}'$ and $\mathbf{e}'$ into two disjoint parts as in the basic SSD, see Section 4.2.

- If the extension $\mathbf{c}$ to all n coordinates of one of those codewords $\mathbf{c}'$ is within distance Wn from $\mathbf{x}$, output it. Here, the extension can easily be computed by solving $\mathbf{m}^\top\mathbf{G}' = \mathbf{c}'$ for $\mathbf{m}$ and computing $\mathbf{c} = \mathbf{m}^\top\mathbf{G}$ afterwards.

If the above procedure fails, one repeats it with permuted coordinates of $\mathcal{C}$. We continue with a rather informal analysis of PSSD, which is sufficient for our purposes, see [Bar98] for a rigorous version. Clearly, a single iteration succeeds if $\mathbf{e}$ has Hamming weight αn on the first βn and $(W-\alpha)n$ on the last $(1-\beta)n$ coordinates, i.e. the coeffcient for the number of iteration is

$$N_{\mathsf{PSSD}}(\alpha,\beta) = \mathrm{H}(W) - \beta\,\mathrm{H}\left(\frac{\alpha}{\beta}\right) - (1-\beta)\,\mathrm{H}\left(\frac{W-\alpha}{1-\beta}\right) . \tag{5.27}$$

The complexity of one iteration is the maximum between the number of vectors of length $\frac{\beta}{2}n$ and weight $\frac{\alpha}{2}n$ *and* the number of solutions to the equation $\mathbf{H}'\mathbf{e}' = \mathbf{s}'$. Since $\mathcal{C}'$ has rate R/β, the syndrome $\mathbf{s}'$ has length $(1-\frac{R}{\beta})\beta n$ and the number of solutions can be estimated as

$$\binom{\beta n}{\alpha n} 2^{-(\beta-R)n}$$

which yields

$$T_{\mathsf{PSSD}}(\alpha,\beta) = N_{\mathsf{PSSD}}(\alpha,\beta) + \max\left\{\Lambda_{\mathsf{PSSD}}(\alpha,\beta), 2\Lambda_{\mathsf{PSSD}}(\alpha,\beta) - (\beta-R)\right\} \tag{5.28}$$

where

$$\Lambda_{\mathsf{PSSD}}(\alpha,\beta) = \frac{\beta}{2}\,\mathrm{H}\left(\frac{\alpha}{\beta}\right) . \tag{5.29}$$

Given these formulas, the next theorem is trivial to prove.

Theorem 5.2.16. *Let (P, L) be a parameter set for the FS-ISD algorithm. Then $(P, R + L)$ is a parameter set for PSSD and it holds*

$$T_{\mathsf{FS-ISD}}(P,L) = T_{\mathsf{PSSD}}(P,R+L) ,$$

i.e. FS-ISD is asymptotically equivalent to PSSD.

Combining this with Theorem 5.2.13 gives the following

Corollary 5.2.17. *PSSD is asymptotically at least as efficient as BCD.*

Proof of Theorem 5.2.16. Let (P, L) be a valid parameter set for FS-ISD, i.e. $0 \leq L \leq 1 - R$ and $\max\{0, R + L + W - 1\} \leq P \leq \min\{R + L, W\}$. Then $(\alpha, \beta) := (P, R + L)$ is a valid parameter set for PSSD, i.e. $\beta = R + L \geq R$ and $\beta = R + L \leq 1$ holds and $\max\{0, \beta + W - 1\} \leq \alpha \leq \min\{\beta, W\}$ is obvious. It immediately follows $N_{\mathsf{FS-ISD}}(P, L) = N_{\mathsf{PSSD}}(P, R+L)$ and $\Lambda_{\mathsf{FS-ISD}}(P, L) = \Lambda_{\mathsf{PSSD}}(P, R+L)$. Moreover, it holds $L = \beta - R$ which completes the proof. □

5.2.3 Properties of Optimal FS-ISD Parameters (L^*, P^*)

For ease of presentation, we write $T(L, P) := T_{\mathsf{FS-ISD}}(L, P)$ for the running time coefficient of FS-ISD and $\Lambda(L, P) := \Lambda_{\mathsf{FS-ISD}}(L, P)$ for the space complexity coefficient. Also recall the useful Eq.(5.25), i.e.

$$\begin{aligned} T(L,P) &= \mathrm{H}(W) - (1-R-L)\,\mathrm{H}\left(\frac{W-P}{1-R-L}\right) + \max\{-L \ , \ -\Lambda(L,P)\} \\ &= \mathrm{H}(W) - (1-R-L)\,\mathrm{H}\left(\frac{W-P}{1-R-L}\right) + \max\left\{-L \ , \ -\frac{R+L}{2}\,\mathrm{H}\left(\frac{P}{R+L}\right)\right\} \ . \end{aligned}$$

In preparation for Section 6.3, where we prove the superiority of our new algorithm, we will now establish some useful facts about *optimal parameters* (L^*, P^*) for the FS-ISD algorithm, namely it always holds

- $0 < L^* < 1 - R$ (see Lemma 5.2.18 and Lemma 5.2.19),
- $0 < P^* < W$ (see Corollary 5.2.20),
- $W - P^* < \frac{1-R-L^*}{2}$ and $P^* < \frac{R+L^*}{2}$ (see Lemma 5.2.21 and Lemma 5.2.22).

Recall that the complete parameter space of FS-ISD is defined by $0 \leq L \leq 1 - R$ and $\max\{0, R + L + W - 1\} \leq P \leq \min\{R + L, W\}$. Essentially, we will show that optimal parameters (L^*, P^*) are not located at the *corners of the parameter space*. Almost all of the proofs will be constructive and we will show how to improve on parameters that do not fulfil the respective conditions. The proofs are elementary (basic calculus) but rather technical and can safely be skipped if one is only interested in the high-level idea of our proof of superiority in Section 6.3. We will frequently compute the Taylor series expansion of H where a term like $(x + \epsilon)\log(x + \epsilon)$ has series expansion $x\log(x) + \epsilon\log(ex) + \mathcal{O}(\epsilon^2)$ around $\epsilon = 0$ (where one has to take care of $x > 0$ which is necessary for log being differentiable).

Besides the above points, it will also follow that optimal parameters (L^*, P^*) always fulfill $\Lambda(L^*, P^*) \leq L^*$, or equivalently, $2\Lambda(L^*, P^*) - L^* \leq \Lambda(L^*, P^*)$. Put simply, merging the lists $\mathcal{L}_1$ and $\mathcal{L}_2$ is at most as expensive as constructing (or sampling) them, see Corollary 5.2.23.

Lemma 5.2.18. *Let (L^*, P^*) be optimal parameters for FS-ISD, then $0 < L^*$ holds.*

Proof. Fix $0 < R < 1$ and $0 < W \leq \mathrm{D}_{\mathsf{GV}}(R)$. Assume $L^* = 0$. We obtain $\max\{-L^*, -\Lambda(L^*, P^*)\} = 0$ and deduce

$$T(0, P^*) = \mathrm{H}(W) - (1-R)\,\mathrm{H}\left(\frac{W - P^*}{1-R}\right)$$

which is monotone increasing in P^* since $W \leq \mathrm{D}_{\mathsf{GV}}(R) \leq \frac{1-R}{2}$. Thus $P^* = 0$ must hold for optimal parameters (L^*, P^*) with $L^* = 0$ and the FS-ISD algorithm degenerates to Plain ISD. According to Theorem 5.2.13, we have $F_{\mathsf{Ball}}(R, W) \geq F_{\mathsf{FS-ISD}}(R, W) = T(0,0)$. Since Plain ISD can also be instantiated in the BCD framework (with $P = Q = L = 0$), it follows $F_{\mathsf{Ball}}(R, W) = T(0,0)$ and we have found optimal BCD parameters with $Q = L = 0$. This contradicts Theorem 4.3.4 and thus $L^* = 0$ must be false. □

Lemma 5.2.19. *Let (L^*, P^*) be optimal parameters for FS-ISD, then $L^* < 1 - R$ holds.*

Proof. Fix $0 < R < 1$ and $0 < W \leq \mathrm{D}_{\mathsf{GV}}(R)$. Assume $L^* = 1 - R$. The restriction (5.24) implies $P^* = W$. In this case, FS-ISD degenerates to Split Syndrome Decoding whose complexity is given by $T(1-R, W) = \frac{\mathrm{H}(W)}{2}$. Note that $W > 0$ and $R < 1$ implies $L^* > 0$ and $P^* > 0$, thus there is space to decrease both parameters by some small amount. More precisely, we define

$$L' := L^* - \epsilon \quad \text{and} \quad P' := (1-\epsilon)P^* = (1-\epsilon)W$$

which are valid parameters for some $\epsilon > 0$: Restriction (5.23) on L^* is trivially fulfilled and restriction (5.24) holds since

$$R + L' + W - 1 = W - \epsilon \leq (1-\epsilon)W \leq \min\{1-\epsilon, W\} = \min\{R + L', W\}$$

is valid (we have $W \leq \frac{1}{2} < 1$). Note that our choice of P' is proportional to $R + L' = 1 - \epsilon$. Intuitively, this reduces the number of iteration to a polynomial which can be verified by computing

$$\begin{aligned} N(L', P') &= \mathrm{H}(W) - (1 - R - L')\,\mathrm{H}\left(\frac{W - P'}{1 - R - L'}\right) - (R + L')\,\mathrm{H}\left(\frac{P'}{R + L'}\right) \\ &= \mathrm{H}(W) - \epsilon\,\mathrm{H}\left(\frac{\epsilon W}{\epsilon}\right) - (1-\epsilon)\,\mathrm{H}\left(\frac{(1-\epsilon)W}{1-\epsilon}\right) = 0 \ . \end{aligned}$$

Furthermore we obtain $\Lambda(L', P') = \frac{1-\epsilon}{2}\,\mathrm{H}(W)$ which yields

$$\begin{aligned} T(L', P') &= \max\{2\Lambda(L', P') - L', \Lambda(L', P')\} \\ &= \max\left\{(1-\epsilon)H(W) - (1 - R - \epsilon), \frac{1-\epsilon}{2}H(W)\right\} \ . \end{aligned}$$

Clearly, the second argument of the maximum is smaller than $T(1-R, W) = \frac{\mathrm{H}(W)}{2}$ for $\epsilon > 0$ and it remains to show

$$(1-\epsilon)\,\mathrm{H}(W) - (1 - R - \epsilon) < \frac{1}{2}\,\mathrm{H}(W) \tag{5.30}$$

for all $W \leq \mathrm{D}_{\mathsf{GV}}(R)$ and some $\epsilon > 0$. Rearranging Eq.(5.30) yields the sufficient condition

$$\epsilon < \frac{1 - R - \frac{1}{2}\mathrm{H}(W)}{1 - \mathrm{H}(W)} \quad .$$

Since $W \leq \mathrm{D}_{\mathsf{GV}}(R) = \mathrm{H}^{(-1)}(1-R)$ it follows $1 - R - \frac{\mathrm{H}(W)}{2} \geq \frac{1-R}{2} > 0$ for all $0 < R < 1$, i.e. the right-hand side is strictly positive. This finally proves the existence of some suitable $\epsilon > 0$ with $T(L', P') < T(1-R, W)$ which contradicts the optimality of $L^* = 1 - R$. □

Corollary 5.2.20. *For optimal FS-ISD parameters (L^*, P^*) it holds*

$$\max\{0, R + L^* + W - 1\} < P^* < W \quad .$$

Proof. Let (L^*, P^*) be optimal parameters, i.e. by Lemma 5.2.18 and Lemma 5.2.19 we have $0 < L^* < 1 - R$ and we write $L^* = 1 - R - \epsilon$ for some $\epsilon > 0$. Note $P^* = 0$ yields

$$T(L^*, P^*) = T(L^*, 0) = \mathrm{H}(W) - (1 - R - L^*)\,\mathrm{H}\left(\frac{W}{1 - R - L^*}\right)$$

which is clearly minimal for $L^* = 0$ in contradiction to $L^* > 0$. Now consider the case $P^* = R + L^* + W - 1 = W - \epsilon$ (i.e. $\epsilon < W$) and define $P' := (1-\epsilon)W$. As in the preceding proof, P' is valid and minimizes $N(L^*, P') = 0$. We obtain

$$T(L^*, P^*) = \mathrm{H}(W) + \max\left\{-\frac{1-\epsilon}{2}\,\mathrm{H}\left(\frac{W-\epsilon}{1-\epsilon}\right)\ ,\ -(1 - R - \epsilon)\right\}$$

and

$$T(L^*, P') = \max\left\{\frac{1-\epsilon}{2}\,\mathrm{H}(W)\ ,\ (1-\epsilon)\,\mathrm{H}(W) - (1 - R - \epsilon)\right\} \quad .$$

If the maximum in $T(L^*, P^*)$ is $-(1 - R - \epsilon)$, then the maximum in $T(L^*, P')$ is $(1 - \epsilon)\,\mathrm{H}(W) - (1 - R - \epsilon)$ since

$$\begin{aligned}(1-\epsilon)\,\mathrm{H}(W) - (1 - R - \epsilon) &\geq (1-\epsilon)\,\mathrm{H}(W) - \frac{1-\epsilon}{2}\,\mathrm{H}\left(\frac{W-\epsilon}{1-\epsilon}\right) \\ &\geq \frac{1-\epsilon}{2}\,\mathrm{H}(W) + \frac{1-\epsilon}{2}\underbrace{\left[\mathrm{H}(W) - \mathrm{H}\left(\frac{W-\epsilon}{1-\epsilon}\right)\right]}_{>0} > \frac{1-\epsilon}{2}\,\mathrm{H}(W)\end{aligned}$$

since $\frac{W-\epsilon}{1-\epsilon} < W < \frac{1}{2}$. Thus, it follows $T(L^*, P^*) - T(L^*, P') = \epsilon\,\mathrm{H}(W) > 0$. Now, let the maximum in $T(L^*, P^*)$ be given by the left argument. Let us first assume that the maximum in $T(L^*, P')$ is also given by the left argument. As before, it follows

$$\begin{aligned}T(L^*, P^*) - T(L^*, P') &= \mathrm{H}(W) - \frac{1-\epsilon}{2}\,\mathrm{H}\left(\frac{W-\epsilon}{1-\epsilon}\right) - \frac{1-\epsilon}{2}\,\mathrm{H}(W) \\ &= \epsilon\,\mathrm{H}(W) + \frac{1-\epsilon}{2}\left[\mathrm{H}(W) - \mathrm{H}\left(\frac{W-\epsilon}{1-\epsilon}\right)\right] > 0 \quad .\end{aligned}$$

If the maximum in $T(L^*, P')$ is given by the right argument, we need to show that

$$h(\epsilon) := T(L^*, P^*) - T(L^*, P') = \epsilon\,\mathrm{H}(W) - \frac{1-\epsilon}{2}\,\mathrm{H}\left(\frac{W-\epsilon}{1-\epsilon}\right) + (1-R-\epsilon) > 0$$

for all $0 < \epsilon < W$ and all $0 < W \leq \mathrm{D}_{\mathsf{GV}}(R) = \mathrm{H}^{(-1)}(1-R)$. Note that $h(0) = 1 - R - \frac{\mathrm{H}(W)}{2} \geq \frac{1-R}{2} > 0$ for all $0 < R < 1$. Moreover,

$$\frac{dh}{d\epsilon} = \mathrm{H}(W) - 1 - \frac{1}{2}\log\left(\frac{W-\epsilon}{1-\epsilon}\right) \geq \mathrm{H}(W) - 1 - \frac{1}{2}\log W > 0$$

for all $0 < W < \frac{1}{2}$, i.e. h is monotone increasing for all W and thus $h(\epsilon) > 0$ for all $0 < \epsilon < W$.

Let us finally assume $P^* = W$. It follows

$$T(L^*, P^*) = T(L^*, W) = \mathrm{H}(W) + \max\left\{-L^* \;,\; -\frac{R+L^*}{2}\,\mathrm{H}\left(\frac{W}{R+L^*}\right)\right\}$$

which is clearly minimised for the maximal choice $L^* = 1 - R$ (contradicting $L^* < 1 - R$). □

The next lemma gives a stronger lower bound on P^* than the preceding corollary. We show that $P^* > W - \frac{1-R-L^*}{2}$. This observation will play a crucial role in the proof of superiority in Section 6.3.

Lemma 5.2.21. *For optimal parameters (L^*, P^*) it holds*

$$W - P^* < \frac{1-R-L^*}{2} \;.$$

Proof. Let (L^*, P^*) be optimal parameters, i.e. $0 < L^* < 1-R$ and $\max\{0, R+L^*+W-1\} < P^* < W$. Suppose that $W - P^* \geq \frac{1-R-L^*}{2}$. Then, $P^* < \frac{R+L^*}{2}$ must hold (since $P^* \geq \frac{R+L^*}{2}$ would imply $W = W - P^* + P^* \geq \frac{1}{2}$ which is a contradiction because $W \leq \mathrm{D}_{\mathsf{GV}}(R) < \frac{1}{2}$ for all $0 < R < 1$). We distinguish two cases: First, let $W - P^* = \frac{1-R-L^*}{2} + \epsilon$ for some $\epsilon > 0$. Then define $P' := P^* + \delta$ for some $0 < \delta < \epsilon$ (note that P' remains valid, i.e. $P' \leq \min\{R + L^*, W\}$, for small enough δ since $P^* < \frac{R+L^*}{2}$ and $P^* < W$). Furthermore, $\Lambda(L^*, P)$ is a continuous function in P which implies $\Lambda(L^*, P^*) < L^*$ iff $\Lambda(L^*, P') < L^*$ for small enough δ, i.e. we obtain

$$\begin{aligned} T(L^*, P^*) - T(L^*, P') = \; & (1-R-L^*)\underbrace{\left[\mathrm{H}\left(\frac{W-P'}{1-R-L^*}\right) - \mathrm{H}\left(\frac{W-P^*}{1-R-L^*}\right)\right]}_{=:A} \\ & + \max\{0 \;,\; \frac{R+L^*}{2}\underbrace{\left[\mathrm{H}\left(\frac{P'}{R+L^*}\right) - \mathrm{H}\left(\frac{P^*}{R+L^*}\right)\right]}_{=:B}\} > 0 \end{aligned}$$

where the last inequality holds due to the following reasons: Note that

$$\frac{W-P'}{1-R-L^*} = \frac{W-P^*-\delta}{1-R-L^*} = \frac{1}{2} + \frac{\epsilon-\delta}{1-R-L^*} < \frac{1}{2} + \frac{\epsilon}{1-R-L^*} = \frac{W-P^*}{1-R-L^*}$$

and thus $A > 0$ (clearly $\mathrm{H}(\frac{1}{2}+y) - \mathrm{H}(\frac{1}{2}+x) > 0$ for all $0 \leq y < x \leq \frac{1}{2}$). Similarly, $P^* < \frac{R+L^*}{2}$ implies

$$B = \mathrm{H}\left(\frac{P^*+\delta}{R+L^*}\right) - \mathrm{H}\left(\frac{P^*}{R+L^*}\right) > 0$$

for δ small enough. This implies $T(L^*, P^*) > T(L^*, P')$ in contradiction to the optimality of (L^*, P^*). Consequently, $W - P^* > \frac{1-R-L^*}{2}$ must be false. Thus, the case $W - P^* = \frac{1-R-L^*}{2}$ remains. Note, that $\mathrm{H}\left(\frac{W-P^*}{1-R-L^*}\right) = 1$ which yields

$$T(L^*, P^*) = \mathrm{H}(W) - (1-R-L^*) + \max\{-L^* \ , \ -\frac{R+L^*}{2}\,\mathrm{H}\left(\frac{P^*}{R+L^*}\right)\} \ .$$

Now, we can exclude the case where the maximum is given by $-L^*$: If the maximum were $-L^*$ then $T(L^*, P^*) = \mathrm{H}(W) - (1-R)$ would follow. But $\mathrm{H}(W) - (1-R) \leq 0$ for all $0 < W \leq \mathrm{D}_{\mathsf{GV}}(R)$ and $0 < R < 1$ which is a contradiction to Remark 5.2.12. Consequently, we have $\Lambda(L^*, P^*) < L^*$ which motivates to slightly decrease L^* (note that there is space to decrease L^* for optimal (L^*, P^*) due to Corollary 5.2.20). Consider

$$T(L, P) = \mathrm{H}(W) - (1-R-L)\,\mathrm{H}\left(\frac{W-P}{1-R-L}\right) - \frac{R+L}{2}\,\mathrm{H}\left(\frac{P}{R+L}\right)$$

and compute the Taylor series expansion of $T(L-\epsilon, P)$ around $\epsilon = 0$ to obtain

$$T(L-\epsilon, P) - T(L, P) = \left[\frac{-\log(1-\frac{P}{L+R}) + 2\log(\frac{1-L-R-W+P}{1-L-R})}{2}\right]\epsilon + \mathcal{O}(\epsilon^2)$$

which is negative if

$$\frac{(L+R)(1-L-R-W+P)^2}{(1-L-R)^2(L+R-P)} < 1 \ .$$

Substituting $W = P^* + \frac{1-R-L^*}{2}$ yields the condition

$$\frac{L^*+R}{4(L^*+R-P^*)} < 1$$

which is always fulfilled because $P^* < \frac{R+L^*}{2}$. Thus changing L^* to $L^*-\epsilon$ will decrease T in contradiction to the optimality of L^* and we eventually excluded the remaining case $W - P^* = \frac{1-R-L^*}{2}$. □

The next lemma gives a refined upper bound on P^* which can be proven analogously to Lemma 5.2.21 as before.

Lemma 5.2.22. *For optimal* (L^*, P^*) *it holds*

$$P^* < \frac{R + L^*}{2} \quad .$$

Proof of Lemma 5.2.22. Let (L^*, P^*) be optimal parameters. Suppose $P^* = \frac{R+L^*}{2} + \epsilon$ for some $\epsilon > 0$. Define $P' := P^* - \delta$ for some $0 < \delta < \epsilon$ (note that P' remains valid since there is always space to decrease optimal P^* due to Corollary 5.2.20). As in Lemma 5.2.21 we obtain

$$\begin{aligned} T(L^*, P^*) - T(L^*, P') = \; & (1 - R - L^*) \underbrace{\left[\mathrm{H}\left(\frac{W - P'}{1 - R - L^*} \right) - \mathrm{H}\left(\frac{W - P^*}{1 - R - L^*} \right) \right]}_{=:A} \\ & + \max\{0 \;,\; \frac{R + L^*}{2} \underbrace{\left[\mathrm{H}\left(\frac{P'}{R + L^*} \right) - \mathrm{H}\left(\frac{P^*}{R + L^*} \right) \right]}_{=:B} \} > 0 \end{aligned}$$

since

$$\frac{P'}{R + L^*} = \frac{P^*}{R + L^*} - \frac{\delta}{R + L^*} = \frac{1}{2} + \frac{\epsilon - \delta}{R + L^*} < \frac{1}{2} + \frac{\epsilon}{R + L^*} = \frac{P^*}{R + L^*}$$

which implies $B > 0$ and

$$W - P' = W - P^* + \delta > W - P^*$$

which implies $A > 0$ (recall that $W - P^* < \frac{1-R-L^*}{2}$ must hold). Altogether, $T(L^*, P^*) - T(L^*, P') > 0$ in contradiction to the optimality of (L^*, P^*). It remains to consider the case $P^* = \frac{R+L^*}{2}$. First, consider $T(L^*, P) = \mathrm{H}(W) - (1 - R - L^*)\,\mathrm{H}\left(\frac{W-P}{1-R-L^*}\right) - L^*$ as a function in P and compute

$$\frac{dT}{dP} = -\log\left(\frac{W - P}{1 - L^* - R} \right) + \log\left(\frac{1 - L^* - R - W + P}{1 - L^* - R} \right) \quad .$$

Now, for $P^* = \frac{R+L^*}{2}$, the necessary condition $\frac{dT}{dP} = 0$ yields

$$\frac{L^* + R + 2\,W - 2}{L^* + R - 2\,W} = 1$$

which is equivalent to $W = \frac{1}{2}$ (but this is never true for $0 < W \leq \mathrm{D}_{\mathsf{GV}}(R)$ and $0 < R < 1$). Analogously, consider $T(L^*, P) = \mathrm{H}(W) - (1 - R - L^*)\,\mathrm{H}\left(\frac{W-P}{1-R-L^*}\right) - \frac{R+L^*}{2}\,\mathrm{H}\left(\frac{P}{R+L^*}\right)$ which gives

$$\frac{dT}{dP} = \frac{1}{2}\left[\log\left(\frac{P}{L^* + R} \right) - \log\left(1 - \frac{P}{L^* + R} \right) \right] + \log\left(\frac{W - P}{1 - L^* - R} \right) - \log\left(\frac{1 - L^* - R - W + P}{1 - L^* - R} \right).$$

Thus, for $P^* = \frac{R+L^*}{2}$ we obtain the same unachievable condition as before. Consequently, optimal parameters can not have $P^* = \frac{R+L^*}{2}$. □

We conclude with the following corollary which will later be used in the main proof of Section 6.3.

Corollary 5.2.23. *For optimal* (L^*, P^*) *it holds*

$$\Lambda(L^*, P^*) \leq L^* \ .$$

Proof. Let (L^*, P^*) be optimal, i.e. in particular we have $0 < L^* < 1 - R$ and $W - P^* < \frac{1-R-L^*}{2}$. Suppose $L^* < \Lambda(L^*, P^*)$, i.e. consider

$$T(L, P) = \mathrm{H}(W) - (1 - R - L)\,\mathrm{H}\left(\frac{W - P}{1 - R - L}\right) - L$$

and compute the series expansion of $T(L + \epsilon, P)$ around $\epsilon = 0$ to obtain

$$T(L + \epsilon, P) - T(L, P) = \left[-1 + \log\left(\frac{1 - R - L}{1 - R - L - W + P}\right)\right]\epsilon + \mathcal{O}(\epsilon^2)$$

which is negative for (L^*, P^*) since $W - P^* < \frac{1-R-L^*}{2}$ implies $1 - R - L^* - W + P > \frac{1-R-L^*}{2}$ and thus

$$\log\left(\frac{1 - R - L^*}{1 - R - L^* - W + P^*}\right) < \log 2 < 1 \ .$$

Consequently, slightly increasing L^* yields a better running time, contradicting the optimality of L^*.

□

5.2.4 Sampling with $\alpha > 0$ and $\Delta = 0$

Recall that the parameter α controls the size of the range of overlapping coordinates and Δ controls the amount of intersecting coordinates within this range. Thus, $\alpha > 0$ implies the existence of representations. We will now study whether a sampling based algorithm is capable of exploiting those representations effectively. The intuitive message of this section is that

Sampling does *not* allow to exploit representations effectively.

More formally, we consider SampleLists for varying $\alpha \in [0, \frac{1}{2}]$ and $\Delta = 0$ and show that the resulting algorithm will never beat the *deterministic* variant for $\alpha = 0$ where *no* representation exist at all. Recall that the running time of SampleLists for fixed parameters (L, P) (and varying $0 \leq \alpha \leq \frac{1}{2}$ and $\Delta = 0$) is given by

$$T(R, W, L, P, \alpha, 0) = N(R, W, L, P) + \max\{\Lambda(R, L, P, \alpha, 0) \ , \ 2\Lambda(R, L, P, \alpha, 0) - L\} \ ,$$

which we will write as $T(L, P, \alpha)$ and $\Lambda(L, P, \alpha)$. Note that N is *independent* of α which allows us to focus on the coefficient

$$\begin{aligned}\Lambda(L, P, \alpha) &= \Sigma(L, P, \alpha) - \frac{1}{2}\mathfrak{P}(L, P, \alpha) \\ &= \left(\frac{1}{2} + \alpha\right)(R + L)\,\mathrm{H}\left(\frac{P}{(1 + 2\alpha)(R + L)}\right) - \alpha P \ .\end{aligned}$$

For fixed P and L we will show that $\alpha = 0$ minimizes $\Lambda(L, P, \alpha)$.

Theorem 5.2.24 ($\alpha > 0$ is suboptimal)**.** *For fixed, valid parameters (L, P) and for all $0 < \alpha \leq \frac{1}{2}$ it holds*

$$T(L, P, 0, 0) \leq T(L, P, \alpha, 0)$$

where equality holds iff $P = 0$, i.e. $\alpha > 0$ is suboptimal.

The proof is elementary but requires the following fact about the binary entropy function.

Lemma 5.2.25. *For all $x \in (0, 1]$ it holds $\frac{1}{2}\,\mathrm{H}(x) < \mathrm{H}\left(\frac{x}{2}\right) - \frac{x}{2}$.*

Proof. Define $h(x) := \mathrm{H}\left(\frac{x}{2}\right) - \frac{x}{2} - \frac{1}{2}\,\mathrm{H}(x) = \frac{1-x}{2}\log(1-x) - (1-\frac{x}{2})\log(1-\frac{x}{2})$ and observe that

$$\frac{dh}{dx} = \frac{\log(1-\frac{x}{2}) - \log(1-x)}{2} > 0$$

for all $x \in (0, 1)$ since $\log(x)$ is monotone increasing. Thus $h(x)$ is also monotone increasing and $\lim_{x\to 0} h(x) = 0$, consequently $h(x) > 0$ for all $x \in (0, 1]$. □

Proof of Theorem 5.2.24. It suffices to consider $\Lambda(L, P, \alpha)$. There is nothing to prove for $P = 0$ and we focus on $P > 0$. Note that

$$\frac{\partial \Lambda}{\partial \alpha} = -\frac{2(R+L)P}{\ln(2)(1+2\alpha)[(1+2\alpha)(R+L) - P]} < 0$$

since $(1+2\alpha)(R+L) - P > R + L - P \geq 0$ for valid P and any $\alpha \in (0, \frac{1}{2})$. This implies that there can not be a local minimum in the interior of $[0, \frac{1}{2}]$ and it suffices to compare the boundaries (note that Λ is a continuous function in α defined over the compact interval $[0, \frac{1}{2}]$, thus there must be a minimum in $[0, \frac{1}{2}]$). We obtain

$$\Lambda(L, P, 0) = \frac{R+L}{2}\,\mathrm{H}\left(\frac{P}{R+L}\right) < (R+L)\,\mathrm{H}\left(\frac{P}{2(R+L)}\right) - \frac{P}{2} = \Lambda(L, P, \frac{1}{2})$$

where we applied Lemma 5.2.25 for $x = \frac{P}{R+L}$. Note that optimal parameters (L^*, P^*) for $\alpha = 0$ always fulfil $P^* > 0$ and in this case we obtain a strict inequality $T(L^*, P^*, 0, 0) < T(L^*, P^*, \alpha, 0)$, i.e. $\alpha > 0$ is indeed suboptimal. □

In the next section, we will also allow for $\Delta > 0$ which further increases the number of representations. In this case, there exists a sampling based algorithm that achieves the same performance as its deterministic variant.

5.2.5 Sampling with $\alpha > 0$ and $\Delta > 0$

In this section we achieve an important goal, namely we

Lay the foundations for improved ISD algorithms (using representations).

This will be done by proving that sampling for $\alpha = \frac{1}{2}$ and appropriate $\Delta > 0$ is equivalent to FS-ISD. That is, we can use sampling for $\alpha = \frac{1}{2}$ and some $\Delta > 0$ (where we can exploit representations in a maximal way) as a starting point for the improved algorithm of the next section. We will now extend the FS-ISD algorithm to the setting $\alpha = \frac{1}{2}$ and $\Delta > 0$: In contrary to the preceding section (where $\Delta = 0$) we give a transformation ϕ that maps valid parameters (L, P) (for $\alpha = \Delta = 0$) to valid parameters (L, P) and $\Delta = \phi(L, P)$ (for $\alpha = \frac{1}{2}$) while preserving the running time of the algorithm.

Theorem 5.2.26. *Let (L, P) be valid FS-ISD parameters ($\alpha = \Delta = 0$) with $P \leq \frac{R+L}{2}$. Then (L, P) and $\Delta := \phi(L, P)$ with*

$$\phi(L, P) = \frac{1}{2}\left[R + L - P - \sqrt{(R + L)(R + L - 2P)}\right] \tag{5.31}$$

are valid parameters for $\alpha = \frac{1}{2}$ and it holds

$$T(L, P, 0, 0) = T(L, P, \frac{1}{2}, \Delta) \ .$$

Remark 5.2.27. Note that the condition $P \leq \frac{R+L}{2}$ is required to guarantee that $\phi(L, P)$ is well-defined ($\phi(L, P) \in \mathbb{R}$). Due to Lemma 5.2.22 this condition will always be fulfilled for *optimal* FS-ISD parameters. In the proof of superiority in Section 6.3 we will start from optimal FS-ISD parameters and extend them to $\alpha = \frac{1}{2}$. Eventually, these parameters can be modified to further improve the running time of our new algorithm by an exponential factor.

Proof of Theorem 5.2.26. Let (L, P) be valid parameters for FS-ISD with $P \leq \frac{R+L}{2}$. Since L and P remain unchanged, it is sufficient to show $0 \leq \Delta \leq R + L - P$ in order to prove the validity w.r.t. $\alpha = \frac{1}{2}$. This follows immediately since $\Delta = \frac{1}{2}(R + L - P - \sqrt{(R + L)(R + L - 2P)}) \leq \frac{R+L-P}{2}$. Recall that

$$T(L, P, \alpha, \Delta) = N(L, P) + \max\{2\Lambda(L, P, \alpha, \Delta) - L \ , \ \Lambda(L, P, \alpha, \Delta)\} \ ,$$

i.e. $N(L, P)$ is independent of α and Δ and it suffices to show $\Lambda(L, P, 0, 0) = \Lambda(L, P, \frac{1}{2}, \Delta)$ in order to prove $T(L, P, 0, 0) = T(L, P, \frac{1}{2}, \Delta)$. For ease of presentation we define $x :=$

$R + L$ and consider two functions

$$f(x,P) \coloneqq \frac{x}{2}\,\mathrm{H}\!\left(\frac{P}{x}\right) = \Lambda(L,P,0,0) \ ,$$
$$g(x,P,\Delta) \coloneqq x\,\mathrm{H}\!\left(\frac{\frac{P}{2}+\Delta}{x}\right) - \frac{1}{2}\left[P + (x-P)\,\mathrm{H}\!\left(\frac{\Delta}{x-P}\right)\right] = \Lambda(L,P,\frac{1}{2},\Delta) \ .$$

Thus, we aim to prove $f(x,P) - g(x,P,\Delta) = 0$ for $\Delta = \frac{1}{2}(x - P - \sqrt{x(x-2P)})$. Now, consider the change of variables

$$\tilde{P} \coloneqq \frac{P}{x} \quad \text{and} \quad \tilde{\Delta} \coloneqq \frac{\frac{1}{2}(x-P)-\Delta}{x}$$

and note that $\frac{f(x,P)-g(x,P,\Delta)}{x} = 0$ iff $\tilde{f}(\tilde{P}) - \tilde{g}(\tilde{P},\tilde{\Delta}) = 0$ where

$$\tilde{f}(\tilde{P}) = \frac{1}{2}\,\mathrm{H}(\tilde{P}) \ ,$$
$$\tilde{g}(\tilde{P},\tilde{\Delta}) = \mathrm{H}\!\left(\frac{1}{2}+\tilde{\Delta}\right) - \frac{1}{2}\left[\tilde{P} + (1-\tilde{P})\,\mathrm{H}\!\left(\frac{1}{2}+\frac{\tilde{\Delta}}{1-\tilde{P}}\right)\right] \ .$$

Moreover, $\Delta = \frac{1}{2}(x - P - \sqrt{x(x-2P)})$ yields the relation $\tilde{P} = \frac{1}{2} - 2\tilde{\Delta}^2$. As a reward for this transformation, we now obtain a much simpler equation in *one* single variable $\tilde{\Delta}$. We compute

$$\begin{aligned}
\tilde{f}(\tilde{P}) - \tilde{g}(\tilde{P},\tilde{\Delta}) &= \tilde{f}(\tfrac{1}{2} - 2\tilde{\Delta}^2) - \tilde{g}(\tfrac{1}{2} - 2\tilde{\Delta}^2, \tilde{\Delta}) \\
&= \frac{1}{2}\,\mathrm{H}\!\left(\frac{1}{2}+2\tilde{\Delta}^2\right) - \mathrm{H}\!\left(\tilde{\Delta}+\frac{1}{2}\right) + \frac{1}{2}\left(\frac{1}{2}-2\tilde{\Delta}^2\right) + \left(\frac{1}{4}+\tilde{\Delta}^2\right)\mathrm{H}\!\left(\frac{(\tilde{\Delta}+\frac{1}{2})^2}{\frac{1}{2}+2\tilde{\Delta}^2}\right) \\
&= -\left(\frac{1}{4}+\tilde{\Delta}^2\right)\log\left(\frac{1}{2}+2\tilde{\Delta}^2\right) - \left(\frac{1}{4}-\tilde{\Delta}^2\right)\log\left(\frac{1}{2}-2\tilde{\Delta}^2\right) \\
&\quad + \left(\frac{1}{2}+\tilde{\Delta}\right)\log\left(\frac{1}{2}+\tilde{\Delta}\right) + \left(\frac{1}{2}-\tilde{\Delta}\right)\log\left(\frac{1}{2}-\tilde{\Delta}\right) + \frac{1}{2}\left(\frac{1}{2}-2\tilde{\Delta}^2\right) \\
&\quad - \cancel{\left(\frac{1}{4}+\tilde{\Delta}^2\right)\frac{(\frac{1}{2}+\tilde{\Delta})^2}{\frac{1}{2}+2\tilde{\Delta}^2}}\,2\log\left(\frac{1}{2}+\tilde{\Delta}\right) + \frac{1}{2}\left(\frac{1}{2}+\tilde{\Delta}\right)^2\log\left(\frac{1}{2}+2\tilde{\Delta}^2\right) \\
&\quad - \left(\frac{1}{2}-\tilde{\Delta}\right)^2\log\left(\frac{1}{2}-d\right) + \frac{1}{2}\left(\frac{1}{2}-\tilde{\Delta}\right)^2\log\left(\frac{1}{2}+2\tilde{\Delta}^2\right)
\end{aligned}$$

and grouping together the last three lines according to the different logarithms yields

$$\begin{aligned}
&\tilde{f}(\tilde{P}) - \tilde{g}(\tilde{P}, \tilde{\Delta}) = \\
&\quad \cancel{-\left(\frac{1}{4} + \tilde{\Delta}^2\right) \log\left(\frac{1}{2} + 2\tilde{\Delta}^2\right)} - \left(\frac{1}{4} - \tilde{\Delta}^2\right) \log\left(\frac{1}{2} - 2\tilde{\Delta}^2\right) + \frac{1}{2}\left(\frac{1}{2} - 2\tilde{\Delta}^2\right) \\
&\quad + \underbrace{\left(\left(\frac{1}{2} + \tilde{\Delta}\right) - \left(\frac{1}{2} + \tilde{\Delta}\right)^2\right)}_{=\frac{1}{4} - \tilde{\Delta}^2} \log\left(\frac{1}{2} + \tilde{\Delta}\right) + \underbrace{\left(\left(\frac{1}{2} - \tilde{\Delta}\right) - \left(\frac{1}{2} - \tilde{\Delta}\right)^2\right)}_{=\frac{1}{4} - \tilde{\Delta}^2} \log\left(\frac{1}{2} - \tilde{\Delta}\right) \\
&\quad \cancel{+\left(\frac{1}{4} + \tilde{\Delta}^2\right) \log\left(\frac{1}{2} + 2\tilde{\Delta}^2\right)} \\
&= \frac{1}{2}\left(\frac{1}{2} - 2\tilde{\Delta}^2\right) - \left(\frac{1}{4} - \tilde{\Delta}^2\right) \log\left(\frac{1}{2} - 2\tilde{\Delta}^2\right) + \left(\frac{1}{4} - \tilde{\Delta}^2\right) \log\left(\left(\frac{1}{2} + \tilde{\Delta}\right)\left(\frac{1}{2} - \tilde{\Delta}\right)\right) \\
&= \left(\frac{1}{4} - \tilde{\Delta}^2\right)\left[1 + \log\left(\frac{(\frac{1}{2} + \tilde{\Delta})(\frac{1}{2} - \tilde{\Delta})}{\frac{1}{2} - 2\tilde{\Delta}^2}\right)\right] = 0
\end{aligned}$$

since the argument of the last logarithm simplifies to $\frac{1}{2}$. □

We conclude this chapter with some final remarks.

Remark 5.2.28.

1. The above proof implicitly shows that ϕ *preserves the size of the lists* $\mathcal{L}_1$ *and* $\mathcal{L}_2$ and we will exploit this property in the superiority proof of Section 6.3.

2. The formula for $\phi(L, P)$ was obtained by computing local extrema of $\Lambda(L, P, \Delta)$ when viewed as function of Δ for fixed L and P. In doing so, one obtains two (symmetric) local minima at $\Delta = \frac{1}{2}(R + L - P) \pm \sqrt{(R + L)(R + L - 2P)}$. In particular, this proves that $\phi(L, P)$ picks the best possible Δ for any fixed L, P, i.e. there is no way to outperform the deterministic variant with $\alpha = \Delta = 0$.

6 Improved ISD Algorithms

> "A person who never made a mistake never tried anything new."
>
> *Albert Einstein*

This chapter contains the main algorithmic contribution of this thesis, namely

The design of asymptotically fastest ISD algorithms.

All subsequent algorithms are based on the so-called *Representation Technique* that was originally introduced by Howgrave-Graham and Joux in the context of generic algorithms for the subset-sum problem [HGJ10]. It was subsequently improved by Becker, Coron and Joux in [BCJ11]. In contrary to the sampling-based algorithms of Chapter 5, the goal is to exploit the exponentially many representations $\tilde{\mathbf{e}}_1 + \tilde{\mathbf{e}}_2 = \tilde{\mathbf{e}}$ in a *randomised, constructive* way. Recall that the parameters α and δ in the generalised framework control the amount of representations and yield different algorithms:

- The first algorithm was published in [MMT11] and deals with $\alpha = \frac{1}{2}$ and $\delta = 0$.
- The second algorithm generalises to $\delta > 0$ and was published in [BJMM12].

In this thesis, we will focus on the latter (as the former can be seen as a special case).

Roadmap

First, we formally introduce and analyse Merge-Join, which has already been used as an important building block in Chapter 4 and 5, and will further be used in all improved algorithms. We will then proceed to a simple variant of our improved ISD algorithm in Section 6.2. The full algorithm, as presented in [BJMM12] can be found in Section 6.4. In contrary to [BJMM12], where the superiority of the algorithm was solely supported by numerical optimisation, we will give a *formal proof of superiority* for the simple variant in Section 6.3 based on the results of the preceding chapter. We also present extensive numerical data for optimal parameter choices and compare the different algorithms also with respect to their time and space complexity, see Section 6.4.2 and 6.4.3. We finally conclude with some interesting open problems.

6.1 The Merge-Join Building Block

Given a matrix $\mathbf{Q} \in \mathbb{F}_2^{r\times(k+\ell)}$ and two lists $\mathcal{L}_1$ and $\mathcal{L}_2$ containing binary vectors $\mathbf{x}_1, \ldots, \mathbf{x}_{|\mathcal{L}_1|}$ and $\mathbf{y}_1, \ldots, \mathbf{y}_{|\mathcal{L}_2|}$ of length $k+\ell$, we aim to join those elements $\mathbf{x}_i$ and $\mathbf{y}_j$ into a new list $\mathcal{L} = \mathcal{L}_1 \bowtie \mathcal{L}_2$ whose sum has weight p, i.e. $\mathrm{wt}(\mathbf{x}_i + \mathbf{y}_j) = p$. Furthermore, we require that the corresponding column-sum of $\mathbf{Q}$ already matches a given target $\mathbf{t} \in \mathbb{F}_2^r$, i.e. $\mathbf{Q}(\mathbf{x}_i + \mathbf{y}_j) = \mathbf{t}$.

Searching for matching vectors $\mathbf{Q}\mathbf{y}_j + \mathbf{t}$ and $\mathbf{Q}\mathbf{x}_i$ accomplishes this task. We already discussed the issue of weight-inconsistent matchings, i.e. matchings with $\mathrm{wt}(\mathbf{x}_i + \mathbf{y}_j) \neq p$. Notice that we may also obtain the same vector sum from two different pairs of vectors from $\mathcal{L}_1, \mathcal{L}_2$. In this case we obtain a matched vector that we already have, which we call a **duplicate**. During our matching process we filter out all inconsistent solutions and duplicates.

The matching process is illustrated in Figure 6.1. The elementary algorithm is given as Algorithm 5 and realizes the collision search as follows. Sort the first list lexicographically according to the r-bit labels $L_1(\mathbf{x}_i) := \mathbf{Q}\mathbf{x}_i$ and the second list according to the labels $L_2(\mathbf{y}_j) := \mathbf{Q}\mathbf{y}_j + \mathbf{t}$. We add $\mathbf{t}$ to the labels of the second list to guarantee $\mathbf{Q}(\mathbf{x}_i + \mathbf{y}_j) = \mathbf{t}$.

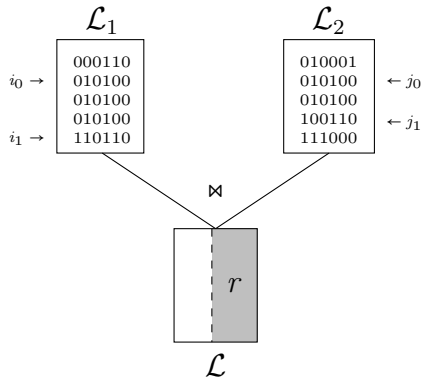


Figure 6.1: Illustration of the Merge-Join algorithm.

To detect all collisions, one now initializes two counters i and j starting at the beginning of the lists $\mathcal{L}_1$ and $\mathcal{L}_2$ and pointing at elements $\mathbf{x}_i$ and $\mathbf{y}_j$. As long as those elements do not yield a collision, either i or j is increased depending on the relative order of the labels $L_1(\mathbf{x}_i)$ and $L_2(\mathbf{y}_j)$. Once a collision $L_1(\mathbf{x}_i) = L_2(\mathbf{y}_j)$ occurs, four auxiliary counters i_0, i_1 and j_0, j_1 are initialised with i and j, respectively. Then i_1 and j_1 can further be incremented as long as the list elements retain the same labels, while i_0 and j_0 mark the first collision (i_0, j_0) between labels $L_1(\mathbf{x}_{i_0})$ and $L_2(\mathbf{y}_{j_0})$. Obviously, this procedure defines two sets $C_1 = \{\mathbf{x}_{i_0}, \ldots, \mathbf{x}_{i_1}\}$ and $C_2 = \{\mathbf{y}_{j_0}, \ldots, \mathbf{y}_{j_1}\}$ such that all possible combinations yield a collision, i.e. the set $C_1 \times C_2$ can be added to the output list $\mathcal{L}$.

This procedure is then continued with $i \leftarrow i_1$ and $j \leftarrow j_1$ until one of the counters i, j arrives at the end of a list. As mentioned before, we remove inconsistent solutions with incorrect weight $\mathrm{wt}(\mathbf{x}_i + \mathbf{y}_j) \neq p$ and duplicate elements $\mathbf{x}_i + \mathbf{y}_j = \mathbf{x}_k + \mathbf{y}_\ell$ on the fly.

Algorithm 5: MERGE-JOIN

input : Matrix $\mathbf{Q} \in \mathbb{F}_2^{r \times (k+\ell)}$, lists $\mathcal{L}_1, \mathcal{L}_2$, weight p and target vector $\mathbf{t} \in \mathbb{F}_2^r$.
output : List $\mathcal{L} = \mathcal{L}_1 \bowtie \mathcal{L}_2$.

```
Lexicographically sort L_1 and L_2 according to the labels L_1(x_i) and L_2(y_j);
Set matching counter M ← 0 and L = ∅. Let i ← 0 and j ← 0;
while i < |L_1| and j < |L_2| do
    if L_1(x_i) <_lex L_2(y_j) then i++;
    if L_1(x_i) >_lex L_2(y_j) then j++;
    if L_1(x_i) = L_2(y_j) then
        Let i_0, i_1 ← i and j_0, j_1 ← j;
        while i_1 < |L_1| and L_1(x_{i_1}) = L_1(x_{i_0}) do i_1++;
        while j_1 < |L_2| and L_2(y_{j_1}) = L_2(y_{j_0}) do j_1++;
        for i ← i_0 to i_1 − 1 do
            for j ← j_0 to j_1 − 1 do
                M++;
                Insert x_i + y_j into L (unless filtered out);
        Set i ← i_1 and j ← j_1;
return L, M;
```

Note that we introduced a matching counter M which allows us to take into account the time that is spent for removing weight-inconsistent and duplicate matchings. Neglecting the costs for computing and comparing the labels $L_1(\mathbf{x}_i)$ and $L_2(\mathbf{y}_j)$ (which is reasonable in the asymptotic setting), the total running time of MERGE-JOIN is easy to compute.

Lemma 6.1.1. MERGE-JOIN *runs in time*

$$\mathcal{O}\left(\max\left\{|\mathcal{L}_1|\log|\mathcal{L}_1|, |\mathcal{L}_2|\log|\mathcal{L}_2|, M\log M\right\}\right) \quad .$$

Proof. Sorting both lists can be done in time $\mathcal{O}(|\mathcal{L}_i|\log|\mathcal{L}_i|)$ and the while-loop obviously terminates after at most $\mathcal{O}(\max\{|\mathcal{L}_i|\})$ steps. The search for duplicates can also be done in time $\mathcal{O}(M\log M)$ in the worst-case. Moreover, the whole execution of the algorithm yields exactly M matchings. □

6.2 Beating Ball-Collision Decoding - A New ISD Algorithm

In the following we fix $\alpha = \frac{1}{2}$ and $\Delta > 0$ in our generalised ISD framework (for $\Delta = 0$ one can recover a simple variant of the improved algorithms presented in [MMT11]). In particular we have $W_1 = W_2 = W_{k+\ell,p_1}$ where we define $p_1 := \frac{p}{2} + \delta$. Recall that $W_{k+\ell,p_1}$ is the Hamming sphere of dimension $k+\ell$ and radius p_1 centered around $\mathbf{0}$, that is

W_1 and W_1 contain all $k+\ell$-dimensional binary vectors of Hamming weight exactly p_1. Moreover, note that *every* element in $W = W_{k+\ell,p}$ is balanced since $\alpha = \frac{1}{2}$. As before, we set $\sigma := |W_1| = |W_2|$ where

$$|W_1| = |W_2| = \binom{k+\ell}{p_1}$$

and ρ as the number of representations, i.e.

$$\rho = \binom{p}{\frac{p}{2}}\binom{k+\ell-p}{\delta} \ .$$

Recall that we aim to solve the equation

$$\mathbf{Qe} = \mathbf{s} \tag{6.1}$$

where $\mathbf{Q}$ is a $\ell \times (k+\ell)$ matrix, $\mathbf{s}$ has length ℓ and $\mathbf{e}$ should have Hamming weight p. For ease of presentation, we omit all $\tilde{}$ and $'$ symbols from here on in. In contrast to the sampling-based algorithms we have presented thus far, the main idea underlying the improved ISD algorithm of this section is to

Construct structured, size-reduced lists $\mathcal{L}_1$ and $\mathcal{L}_2$ that contain at least one representation with good probability.

This will be achieved by defining some appropriate constraint that has to be fulfilled by all elements in $\mathcal{L}_1$ and $\mathcal{L}_2$ with the following properties:

- The constraint shrinks the lists by a significant amount.
- At least one representation fulfils the constraint with good probability.
- It is possible to efficiently compute all elements that fulfil the constraint.

That is, we aim to define a small, structured subset of $W_1 \times W_2$ that is likely to contain a representation and that can be efficiently computed. Moreover, *both* components $\mathbf{e}_i$ of a pair $(\mathbf{e}_1, \mathbf{e}_2)$ that is *not* a representation must independently fulfil the respective constraint. Such a pair is thus eliminated with much higher probability than a representation. A natural way to implement this idea, which goes back to Wagner's algorithm for the generalised birthday problem [Wag02], is as follows: Choose a random target vector $\mathbf{t}$ of length $r \leq \ell$ (where r has to be specified) and define

$$\mathcal{L}_1 := \{\mathbf{e}_1 \in W_1 : (\mathbf{Qe}_1)_{[r]} = \mathbf{t}\} \ . \tag{6.2}$$

Simply speaking, $\mathcal{L}_1$ contains all vectors of length $k+\ell$ and weight $p_1 = \frac{p}{2} + \delta$ such that their corresponding column-sum $(\mathbf{Qe}_1)_{[r]}$ (projected to the first r coordiantes) matches the random target $\mathbf{t}$. Similarly, define

$$\mathcal{L}_2 := \{\mathbf{e}_2 \in W_2 : (\mathbf{Qe}_2)_{[r]} = \mathbf{t} + \mathbf{s}_{[r]}\} \tag{6.3}$$

and note that by definition every element in $\mathcal{L}_1+\mathcal{L}_2$ already matches the target syndrome $\mathbf{s}$ on the first r coordinates by construction, i.e. $(\mathbf{Q}(\mathbf{e}_1+\mathbf{e}_2))_{[r]} = \mathbf{s}_{[r]}$. Since $\mathbf{Q}$ is a random matrix, $(\mathbf{Q}\mathbf{e}_1)_{[r]}$ and $(\mathbf{Q}\mathbf{e}_2)_{[r]}$ are random vectors of length r. Thus, we expect $\mathcal{L}_1$ and $\mathcal{L}_2$ to have size $\approx \frac{\sigma}{2^r}$ and for large enough r the first condition, *shrinking the lists*, can be achieved. Moreover, let $\mathbf{e} = \mathbf{e}_1 + \mathbf{e}_2$ be a solution to Eq.(6.1). If $(\mathbf{Q}\mathbf{e}_1)_{[r]} = \mathbf{t}$ then $(\mathbf{Q}\mathbf{e}_2)_{[r]} = \mathbf{s}+\mathbf{t}$ also holds. This means that a single representation fulfils the constraint with probability $\frac{1}{2^r}$. Since there are exactly ρ different representations $(\mathbf{e}_1, \mathbf{e}_2)$ of $\mathbf{e}$, one expects $\frac{\rho}{2^r}$ surviving representations in $\mathcal{L}_1 \times \mathcal{L}_2$. Consequently, for $r = \lfloor \log \rho \rfloor$ the expected number of surviving representations is ≥ 1 for every solution $\mathbf{e}$. This gives us confidence in achieving the second condition, *keeping at least one representation with good probability*, although it will take some effort to turn this argument into a rigorous proof. Also note that this choice of r yields an expected list size $\approx \frac{\sigma}{\rho}$ which might be considerably smaller than $\frac{\sigma}{\sqrt{\rho}}$ (which represents the list size for FS-ISD). Once $\mathcal{L}_1$ and $\mathcal{L}_2$ have been created, the (weight-consistent) solutions to Eq.(6.1) can be computed by invoking Merge-Join (with target weight p). The expected number of (possibly weight-inconsistent) matchings is $\approx (\frac{\sigma}{\rho})^2 \frac{\rho}{2^\ell}$ since every pair already matches $\mathbf{s}$ on the first r coordinates. It remains to provide a way to efficiently compute $\mathcal{L}_1$ and $\mathcal{L}_2$ which will be done next.

Creating $\mathcal{L}_1$ and $\mathcal{L}_2$.

We explain how to create $\mathcal{L}_1$. The other list can be constructed analogously. We apply a classical Meet-in-the-middle collision search by decomposing $\mathbf{e}_1$ into $\mathbf{e}_1 = \mathbf{y} + \mathbf{z}$ by two non-overlapping vectors $\mathbf{y}$ and $\mathbf{z}$ of length $k + \ell$. To be more precise, we first choose a random partition of $[k+\ell]$ into two equal sized sets P_1 and P_2, i.e. $[k+\ell] = P_1 \cup P_2$ with $|P_1| = |P_2| = \frac{k+\ell}{2}$, and force $\mathbf{y}$ to have its $\frac{p_1}{2}$ 1-entries in P_1 and $\mathbf{z}$ to have its $\frac{p_1}{2}$ 1-entries in P_2. That is we construct two base lists

$$\mathcal{B}_1(P_1) := \{\mathbf{y} \in \mathbb{F}_2^{k+\ell} \mid \mathrm{wt}(\mathbf{y}) = \frac{p_1}{2} \text{ and } \mathrm{supp}(\mathbf{y}) \subset P_1\}$$

and

$$\mathcal{B}_2(P_2) := \{\mathbf{z} \in \mathbb{F}_2^{k+\ell} \mid \mathrm{wt}(\mathbf{z}) = \frac{p_1}{2} \text{ and } \mathrm{supp}(\mathbf{z}) \subset P_2\}$$

where $\mathrm{supp}(\mathbf{x})$ denotes the support, i.e. the set of all non-zero coordinates, of a vector $\mathbf{x}$. We then invoke Merge-Join to compute

$$\mathcal{L}_1 = \text{Merge-Join}(\mathcal{B}_1(P_1), \mathcal{B}_2(P_2), p_1, \mathbf{Q}_{[r]}, \mathbf{t}) \ .$$

Recall that Merge-Join gets as input the base lists $\mathcal{B}_1$ and $\mathcal{B}_2$ as well as the target vector $\mathbf{t}$ and computes all matchings $(\mathbf{y}, \mathbf{z})$ with $\mathbf{Q}_{[r]}(\mathbf{y}+\mathbf{z}) = \mathbf{t}$ and $\mathrm{wt}(\mathbf{y}+\mathbf{z}) = p_1$ as desired. The complete algorithm, which will be called BasicReps, is given in Algorithm 6.

Remark 6.2.1. Note that decomposing $\mathbf{e}_1$ into $\mathbf{y}$ and $\mathbf{z}$ from fixed, disjoint sets P_1 and P_2 introduces a probability of loosing the vector $\mathbf{e}_1$ and hence the representation $\mathbf{e} = \mathbf{e}_1 + \mathbf{e}_2$.

For a randomly chosen partition P_1, P_2, the probability that $\mathbf{e}_1$ has the same number of 1-entries in P_1 and P_2 is given by

$$P_{\text{split}} = \frac{\binom{(k+\ell)/2}{p_1/2}^2}{\binom{k+\ell}{p_1}}$$

which is asymptotically inverse-polynomial in n. Choosing independent partitions $P_{i,1}$ and $P_{i,2}$ and appropriate base lists $\mathcal{B}_{i,1}$ and $\mathcal{B}_{i,2}$ for the two lists $\mathcal{L}_i$, we can guarantee *independent* splitting conditions for both $\mathbf{e}_1$ and $\mathbf{e}_2$ yielding a total splitting probability of $P_{\text{Split}} = (P_{\text{split}})^2$ (which is still inverse-polynomial in n). We acknowledge Dan Bernstein for proposing to chose independent splitting conditions.

Algorithm 6: BasicReps

input : Parameters p and δ. Matrix $\mathbf{Q} \in \mathbb{F}_2^{\ell \times (k+\ell)}$ and syndrome $\mathbf{s} \in \mathbb{F}_2^{\ell}$.
output : List $\mathcal{L}$ of candidate solutions for Eq.(6.1).
params: Number of representations ρ according to Eq.(5.8) and $r = \lfloor \log \rho \rfloor \leq \ell$.
Weight $p_1 = \frac{p}{2} + \delta$.

Choose random target $\mathbf{t}_1 \in_R \mathbb{F}_2^r$ and set $\mathbf{t}_2 = \mathbf{s}_{[r]} + \mathbf{t}_1$;
for $i = 1$ **to** 2 **do**
  Choose random partition $P_{i,1} \cup P_{i,2} = [k + \ell]$;
  Compute base lists $\mathcal{B}_{i,1}$ and $\mathcal{B}_{i,2}$;
  $\mathcal{L}_i \leftarrow$ Merge-Join$(\mathcal{B}_{i,1}(P_{i,1}), \mathcal{B}_{i,2}(P_{i,2}), p_1, \mathbf{Q}_{[r]}, \mathbf{t}_i)$;
$\mathcal{L} \leftarrow$ Merge-Join$(\mathcal{L}_1, \mathcal{L}_2, p, \mathbf{Q}, \mathbf{s})$;
return $\mathcal{L}$;

The crucial point in the analysis of BasicReps is to deal with a certain class of malformed input matrices $\mathbf{Q}$: For example, imagine $\mathbf{Q}$ being the all-zero matrix. Clearly, no target vector $\mathbf{t} \neq 0$ will be hit by any $\mathbf{e}_1 \in W_1$ and thus $\mathcal{L}_1 = \varnothing$ which implies $\mathcal{L} = \varnothing$, i.e. no candidate solution will be found. Of course, for long codes, $\mathbf{Q} = \mathbf{0}$ is very unlikely. Simply speaking, we will now prove that almost all matrices $\mathbf{Q}$ are good. For a fixed solution $\mathbf{e}$ there is a "good" chance that at least one consistent $\mathbf{e}_1$ hits a randomly chosen target vector $\mathbf{t}$. The proof follows the basic strategy proposed in [BCJ11] for the subset-sum scenario and a proof sketch for the case of decoding already appeared in [BJMM12].

Treating Bad Matrices Q.

The following theorem is a generalisation of [NSS01, Theorem3.2] which only covers the case $m = 1$. We give an entirely different, self-contained combinatorial proof which is completely missing in [BJMM12].

Theorem 6.2.2. *For a fixed matrix $\mathbf{Q} \in \mathbb{F}_2^{m\times n}$, a target vector $\mathbf{t} \in \mathbb{F}_2^m$ and an arbitrary set $\mathcal{B} \subset \mathbb{F}_2^n$, $\varnothing \neq \mathcal{B}$ we define*

$$P_{\mathbf{Q}}(\mathcal{B},\mathbf{t}) := \frac{1}{|\mathcal{B}|}\left|\{\mathbf{x}\in\mathcal{B} \;:\; \mathbf{Q}\mathbf{x}=\mathbf{t}\}\right| \;.$$

It holds

$$\sum_{\mathbf{Q}\in\mathbb{F}_2^{m\times n}}\sum_{\mathbf{t}\in\mathbb{F}_2^m}(P_{\mathbf{Q}}(\mathcal{B},\mathbf{t})-\frac{1}{2^m})^2 = 2^{mn}\frac{2^m-1}{2^m|\mathcal{B}|} \;. \tag{6.4}$$

Note that for fixed $\mathbf{Q}$, one can view $P_{\mathbf{Q}}(\mathcal{B},\mathbf{t})$ as a probability distribution over the $\mathbf{t}$'s, in particular it holds

$$\sum_{\mathbf{t}\in\mathbb{F}_2^m} P_{\mathbf{Q}}(\mathcal{B},\mathbf{t}) = 1 \tag{6.5}$$

In this light, Eq.(6.4) can be seen as the expected deviation of $P_{\mathbf{Q}}(\mathcal{B},\mathbf{t})$ from the uniform distribution over $\mathbb{F}_2^m$ averaged over all $\mathbf{Q}$. Intuitively, for almost all $\mathbf{Q}$, the distribution $P_{\mathbf{Q}}(\mathcal{B},\mathbf{t})$ is close to uniform and thus only a small fraction of target values $\mathbf{t}$ can be bad, i.e. not hit by *any* element in $\mathcal{B}$. This is formalised in Lemma 6.2.3.

Proof of Theorem 6.2.2. We write $N_{\mathbf{Q}}(\mathcal{B},\mathbf{t}) := |\{\mathbf{x}\in\mathcal{B} : \mathbf{Q}\mathbf{x}=\mathbf{t}\}|$. Thus $P_{\mathbf{Q}}(\mathcal{B},\mathbf{t}) = \frac{N_{\mathbf{Q}}(\mathcal{B},\mathbf{t})}{|\mathcal{B}|}$ and $\sum_{\mathbf{t}} N_{\mathbf{Q}}(\mathcal{B},\mathbf{t}) = |\mathcal{B}|$ holds. One computes

$$\sum_{\mathbf{Q}}\sum_{\mathbf{t}}\left(P_{\mathbf{Q}}(\mathcal{B},\mathbf{t})-\frac{1}{2^m}\right)^2 = \sum_{\mathbf{Q}}\sum_{\mathbf{t}}\left[P_{\mathbf{Q}}(\mathcal{B},\mathbf{t})^2 - \frac{2P_{\mathbf{Q}}(\mathcal{B},\mathbf{t})}{2^m}+\frac{1}{2^{2m}}\right]$$

and Eq.(6.5) gives $\sum_{\mathbf{Q}}\sum_{\mathbf{t}} P_{\mathbf{Q}}(\mathcal{B},\mathbf{t}) = 2^{mn}$, i.e. it remains to prove

$$\sum_{\mathbf{Q}}\sum_{\mathbf{t}} P_{\mathbf{Q}}(\mathcal{B},\mathbf{t})^2 = 2^{mn}\left(\frac{2^m-1}{2^m|\mathcal{B}|}+\frac{1}{2^m}\right) \;.$$

We rewrite the left-hand side as

$$\begin{aligned}\sum_{\mathbf{Q}}\sum_{\mathbf{t}} P_{\mathbf{Q}}(\mathcal{B},\mathbf{t})^2 &= \sum_{\mathbf{Q}}\left[\left(\sum_{\mathbf{t}} P_{\mathbf{Q}}(\mathcal{B},\mathbf{t})\right)^2 - \sum_{\mathbf{t}}\sum_{\mathbf{t}'\neq\mathbf{t}} P_{\mathbf{Q}}(\mathcal{B},\mathbf{t})P_{\mathbf{Q}}(\mathcal{B},\mathbf{t}')\right]\\ &= 2^{mn} - \sum_{\mathbf{Q}}\sum_{\mathbf{t}}\sum_{\mathbf{t}'\neq\mathbf{t}} P_{\mathbf{Q}}(\mathcal{B},\mathbf{t})P_{\mathbf{Q}}(\mathcal{B},\mathbf{t}')\end{aligned}$$

where we used Eq.(6.5) again. We now switch from $P_{\mathbf{Q}}(\mathcal{B},\mathbf{t})$ to $N_{\mathbf{Q}}(\mathcal{B},\mathbf{t})$ and it remains to prove

$$\begin{aligned}\sum_{\mathbf{Q}}\sum_{\mathbf{t}}\sum_{\mathbf{t}'\neq\mathbf{t}} N_{\mathbf{Q}}(\mathcal{B},\mathbf{t})N_{\mathbf{Q}}(\mathcal{B},\mathbf{t}') &= 2^{mn}|\mathcal{B}|^2\left(1-\frac{1}{2^m}-\frac{2^m-1}{2^m|\mathcal{B}|}\right)\\ &= 2^{mn}\frac{(2^m-1)|\mathcal{B}|(|\mathcal{B}|-1)}{2^m} \;.\end{aligned}$$

This can be done by induction over the size of $\mathcal{B}$.

For $|\mathcal{B}| = 1$, it holds $N_{\mathbf{Q}}(\mathcal{B}, \mathbf{t}) \cdot N_{\mathbf{Q}}(\mathcal{B}, \mathbf{t}') = 0$ for $\mathbf{t} \neq \mathbf{t}'$ which proves the initial step. Define $\tilde{\mathcal{B}} := \{\mathbf{x}_1, \ldots, \mathbf{x}_{k+1}\}$ and $\mathcal{B} := \tilde{\mathcal{B}} \setminus \{\mathbf{x}\}$ where $\mathbf{x} := \mathbf{x}_{k+1}$. Our goal is to prove

$$\sum_{\mathbf{Q}} \sum_{\mathbf{t}} \sum_{\mathbf{t}' \neq \mathbf{t}} N_{\mathbf{Q}}(\tilde{\mathcal{B}}, \mathbf{t}) N_{\mathbf{Q}}(\tilde{\mathcal{B}}, \mathbf{t}') = 2^{mn} \frac{(2^m - 1)|\tilde{\mathcal{B}}|(|\tilde{\mathcal{B}}| - 1)}{2^m} \tag{6.6}$$

and we start by manipulating the left-hand side as follows: Note that for fixed $\mathbf{t}$, either $\mathbf{Q}\mathbf{x} = \mathbf{t}$ (which gives $N_{\mathbf{Q}}(\tilde{\mathcal{B}}, \mathbf{t}) = N_{\mathbf{Q}}(\mathcal{B}, \mathbf{t}) + 1$) or $\mathbf{Q}\mathbf{x} \neq \mathbf{t}$ (which gives $N_{\mathbf{Q}}(\tilde{\mathcal{B}}, \mathbf{t}') = N_{\mathbf{Q}}(\mathcal{B}, \mathbf{t}') + 1$ for exactly one $\mathbf{t}' \neq \mathbf{t}$). This yields

$$\begin{aligned}
&\sum_{\mathbf{Q}} \sum_{\mathbf{t}} N_{\mathbf{Q}}(\tilde{\mathcal{B}}, \mathbf{t}) \sum_{\mathbf{t}' \neq \mathbf{t}} N_{\mathbf{Q}}(\tilde{\mathcal{B}}, \mathbf{t}') \\
&= \sum_{\mathbf{t}} \sum_{\substack{\mathbf{Q} \\ \mathbf{Q}\mathbf{x} = \mathbf{t}}} (N_{\mathbf{Q}}(\mathcal{B}, \mathbf{t}) + 1) \sum_{\mathbf{t}' \neq \mathbf{t}} N_{\mathbf{Q}}(\mathcal{B}, \mathbf{t}') + \sum_{\mathbf{t}} \sum_{\substack{\mathbf{Q} \\ \mathbf{Q}\mathbf{x} \neq \mathbf{t}}} N_{\mathbf{Q}}(\mathcal{B}, \mathbf{t}) \left[\left(\sum_{\mathbf{t}' \neq \mathbf{t}} N_{\mathbf{Q}}(\mathcal{B}, \mathbf{t}') \right) + 1 \right] \\
&= \underbrace{\sum_{\mathbf{t}} \sum_{\mathbf{Q}} \sum_{\mathbf{t}' \neq \mathbf{t}} N_{\mathbf{Q}}(\mathcal{B}, \mathbf{t}) N_{\mathbf{Q}}(\mathcal{B}, \mathbf{t}')}_{= 2^{mn} \frac{(2^m - 1)|\mathcal{B}|(|\mathcal{B}| - 1)}{2^m}} + \sum_{\mathbf{t}} \sum_{\substack{\mathbf{Q} \\ \mathbf{Q}\mathbf{x} = \mathbf{t}}} \underbrace{\sum_{\mathbf{t}' \neq \mathbf{t}} N_{\mathbf{Q}}(\mathcal{B}, \mathbf{t}')}_{= |\mathcal{B}| - N_{\mathbf{Q}}(\mathcal{B}, \mathbf{t})} + \sum_{\mathbf{t}} \sum_{\substack{\mathbf{Q} \\ \mathbf{Q}\mathbf{x} \neq \mathbf{t}}} N_{\mathbf{Q}}(\mathcal{B}, \mathbf{t}) \\
&= 2^{mn} \left(|\mathcal{B}| + \frac{(2^m - 1)|\mathcal{B}|(|\mathcal{B}| - 1)}{2^m} \right) + \sum_{\mathbf{t}} \Big(\sum_{\substack{\mathbf{Q} \\ \mathbf{Q}\mathbf{x} \neq \mathbf{t}}} N_{\mathbf{Q}}(\mathcal{B}, \mathbf{t}) - \sum_{\substack{\mathbf{Q} \\ \mathbf{Q}\mathbf{x} = \mathbf{t}}} N_{\mathbf{Q}}(\mathcal{B}, \mathbf{t}) \Big)
\end{aligned}$$

by induction hypothesis on $\mathcal{B}$. By substituting $|\mathcal{B}| + 1 = |\tilde{\mathcal{B}}|$ in Eq.(6.6) we must finally show

$$\sum_{\mathbf{t}} \Big(\sum_{\substack{\mathbf{Q} \\ \mathbf{Q}\mathbf{x} \neq \mathbf{t}}} N_{\mathbf{Q}}(\mathcal{B}, \mathbf{t}) - \sum_{\substack{\mathbf{Q} \\ \mathbf{Q}\mathbf{x} = \mathbf{t}}} N_{\mathbf{Q}}(\mathcal{B}, \mathbf{t}) \Big) = 2^{mn} \frac{|\mathcal{B}|(2^m - 2)}{2^m} \quad . \tag{6.7}$$

The left-hand side can be written as

$$\sum_{\mathbf{t}} \sum_{\mathbf{Q}} N_{\mathbf{Q}}(\mathcal{B}, \mathbf{t}) - 2 \sum_{\mathbf{t}} \sum_{\substack{\mathbf{Q} \\ \mathbf{Q}\mathbf{x} = \mathbf{t}}} N_{\mathbf{Q}}(\mathcal{B}, \mathbf{t}) = 2^{mn} |\mathcal{B}| - 2 \sum_{\mathbf{t}} \sum_{\substack{\mathbf{Q} \\ \mathbf{Q}\mathbf{x} = \mathbf{t}}} N_{\mathbf{Q}}(\mathcal{B}, \mathbf{t}) \quad . \tag{6.8}$$

Moreover,

$$\sum_{\mathbf{t}} \sum_{\substack{\mathbf{Q} \\ \mathbf{Q}\mathbf{x} = \mathbf{t}}} N_{\mathbf{Q}}(\mathcal{B}, \mathbf{t}) = \sum_{\tilde{\mathbf{x}} \in \mathcal{B}} \sum_{\mathbf{t}} \sum_{\substack{\mathbf{Q} \\ \mathbf{Q}\mathbf{x} = \mathbf{t} \\ \mathbf{Q}\tilde{\mathbf{x}} = \mathbf{t}}} 1 = |\mathcal{B}| \frac{2^{mn}}{2^m} \tag{6.9}$$

and using this in Eq.(6.8) yields Eq.(6.7) which finishes the proof. The last equality in Eq.(6.9) holds due to the following reasoning: For fixed $\mathbf{x} \neq \tilde{\mathbf{x}}$ consider the equivalence relation

$$\mathbf{Q} \equiv \mathbf{Q}' :\Leftrightarrow \mathbf{Q}\mathbf{x} = \mathbf{Q}'\mathbf{x} \text{ and } \mathbf{Q}\tilde{\mathbf{x}} = \mathbf{Q}'\tilde{\mathbf{x}} \quad ,$$

i.e. every equivalence class $[\mathbf{Q}]$ is characterised by $\mathbf{Qx}$ and $\mathbf{Q\tilde{x}}$ for some representative $\mathbf{Q}$. Consequently, there are 2^{2m} distinct equivalence classes each containing $\frac{2^{mn}}{2^{2m}}$ elements. Thus, the expression

$$\sum_{\mathbf{t}} \sum_{\substack{\mathbf{Q} \\ \mathbf{Qx}=\mathbf{t} \\ \mathbf{Q\tilde{x}}=\mathbf{t}}} 1$$

gives the cardinality of exactly 2^m different equivalence classes which is $\frac{2^{mn}}{2^m}$. □

Lemma 6.2.3. *Let $m \leq n$ and $|\mathcal{B}| \geq 2^m$. Let $\Gamma \in \mathbb{N}$. For all but a $\frac{1}{\Gamma-1}$ fraction of all $\mathbf{Q} \in \mathbb{F}_2^{m \times n}$ it holds*

$$\mathbf{Pr}\left[P_{\mathbf{Q}}(\mathcal{B}, \mathbf{t}) = 0\right] \leq 1 - \frac{1}{\Gamma}$$

where the probability is over the random choice of $\mathbf{t} \in \mathbb{F}_2^m$.

Proof. Let $N_{\mathbf{Q}}(\mathcal{B}, \mathbf{t})$ be defined as in the preceding proof, i.e. for a fixed matrix $\mathbf{Q}$, a target $\mathbf{t}$ is *not* hit iff $N_{\mathbf{Q}}(\mathcal{B}, \mathbf{t}) = 0$. We call $\mathbf{Q}$ *bad* if $N_{\mathbf{Q}}(\mathcal{B}, \mathbf{t}) = 0$ for at least $2^m(1 - \frac{1}{\Gamma})$ many $\mathbf{t}$'s, i.e. at most $\frac{2^m}{\Gamma}$ targets are hit for bad $\mathbf{Q}$. Note that $P_{\mathbf{Q}}(\mathcal{B}, \mathbf{t}) = 0$ iff $N_{\mathbf{Q}}(\mathcal{B}, \mathbf{t}) = 0$. Consequently, if $\mathbf{Q}$ is not bad, there are at least $\frac{2^m}{\Gamma}$ targets $\mathbf{t}$ with $N_{\mathbf{Q}}(\mathcal{B}, \mathbf{t}) > 0$ and thus

$$\mathbf{Pr}\left[P_{\mathbf{Q}}(\mathcal{B}, \mathbf{t}) > 0\right] \geq \frac{1}{\Gamma}$$

for every good $\mathbf{Q}$, i.e. $\mathbf{Pr}\left[P_{\mathbf{Q}}(\mathcal{B}, \mathbf{t}) = 0\right] \leq 1 - \frac{1}{\Gamma}$. Denote by $\chi(\Gamma)$ the number of bad $\mathbf{Q}$'s, i.e. it remains to show $\chi(\Gamma) \leq \frac{2^{mn}}{\Gamma-1}$. Note that the more $P_{\mathbf{Q}}(\mathcal{B}, \mathbf{t})$ deviates from the uniform distribution, the larger $\sum_{\mathbf{t} \in \mathbb{F}_2^m}(P_{\mathbf{Q}}(\mathcal{B}, \mathbf{t}) - \frac{1}{m})^2$ becomes which yields the lower bound

$$\sum_{\mathbf{t} \in \mathbb{F}_2^m} (P_{\mathbf{Q}}(\mathcal{B}, \mathbf{t}) - \frac{1}{2^m})^2 \geq 2^m \left(1 - \frac{1}{\Gamma}\right) \frac{1}{2^{2m}} + \frac{2^m}{\Gamma} \left(\frac{\Gamma - 1}{2^m}\right)^2 = \frac{\Gamma - 1}{2^m} \tag{6.10}$$

for bad $\mathbf{Q}$: At least $2^m(1 - \frac{1}{\Gamma})$ many $\mathbf{t}$ are bad, i.e. $P_{\mathbf{Q}}(\mathbf{B}, \mathbf{t}) = 0$, and we may assume that *exactly* $2^m(1 - \frac{1}{\Gamma})$ many $\mathbf{t}$ are bad (the more $\mathbf{t}$ are bad, the larger becomes the sum). This explains the first summand. For the remaining $\frac{2^m}{\Gamma}$ good $\mathbf{t}$, the sum is clearly minimised if they are uniformly distributed, i.e. $P_{\mathbf{Q}}(\mathcal{B}, \mathbf{t}) = \frac{\Gamma}{2^m}$. This explains the second summand. Combining Eq.(6.10) with Eq.(6.4) eventually gives

$$\frac{2^{mn}(2^m - 1)}{2^m |\mathcal{B}|} \geq \chi(\Gamma) \frac{\Gamma - 1}{2^m}$$

and $|\mathcal{B}| \geq 2^m$ yields $\chi(\Gamma) \leq \frac{2^{mn}}{\Gamma-1}$ as desired. □

We can now use Lemma 6.2.3 to prove that BasicReps is τ-solution-preserving for some fixed, arbitrarily small $\tau > 0$.

Lemma 6.2.4. *For almost all input matrices $\mathbf{Q}$,* BasicReps *is τ-solution-preserving for arbitrary $\tau > 0$.*

Proof. Let $\mathbf{e}$ be a fixed solution to Eq.(6.1). Define $\mathcal{C}(\mathbf{e})$ as the set of all $\mathbf{e}_1 \in W_{k+\ell,p_1}$ consistent to $\mathbf{e}$, i.e. $|\mathcal{C}(\mathbf{e})| = \rho$. Note that a single representation $(\mathbf{e}_1, \mathbf{e}_2)$ is contained in $\mathcal{L}_1 \times \mathcal{L}_2$ if the following three independent events occur:

E_1: $(\mathbf{Q}\mathbf{e}_1)_{[r]} = \mathbf{t}_1$ (where $\mathbf{t}_1$ is the random target vector corresponding to list $\mathcal{L}_1$).

E_2: $\mathbf{e}_1$ splits according to $P_{1,1}$ and $P_{1,2}$, i.e. $\mathbf{e}_1 = \mathbf{y} + \mathbf{z}$ such that $\mathrm{supp}(\mathbf{y}) \subset P_{1,1}$ and $\mathrm{supp}(\mathbf{z}) \subset P_{1,2}$.

E_3: $\mathbf{e}_2$ splits according to $P_{2,1}$ and $P_{2,2}$.

Here, E_1 depends on the random choice of $\mathbf{t}_1$ and the E_i for $i = 2, 3$ depend on the random choice of the partition $P_{i,1} \cup P_{i,2}$, respectively (thus the E_i are independent). First consider E_1 and apply Lemma 6.2.3 with $\mathcal{B} := \mathcal{C}(\mathbf{e})$, $m := r = \lfloor \log \rho \rfloor$ and $n := k + \ell$ (note that $|\mathcal{C}(\mathbf{e})| = \rho \geq 2^r$). It follows

$$\mathbf{Pr}\left[E_1 = 1\right] = \mathbf{Pr}\left[P_{\mathbf{Q}}(\mathcal{C}(\mathbf{e}), \mathbf{t}) > 0\right] \geq \frac{1}{\Gamma}$$

for all but a $\frac{1}{\Gamma^{-1}}$ fraction of all $\mathbf{Q}$'s (and some $\Gamma \in \mathbb{N}$). According to Remark 6.2.1 it further holds

$$\mathbf{Pr}\left[E_2 = 1\right] = \mathbf{Pr}\left[E_3 = 1\right] = P_{\text{split}} = \frac{\binom{(k+\ell)/2}{p_1/2}^2}{\binom{k+\ell}{p_1}} \ .$$

Consequently, $\mathbf{e}$ will be found by BasicReps if *at least* one representation $(\mathbf{e}_1, \mathbf{e}_2)$ fulfils all E_i. This probability can easily be lower bounded by combining the preceding steps, i.e.

$$P(\mathbf{e}) := \mathbf{Pr}\left[\mathbf{e} \in \mathcal{L}_1 + \mathcal{L}_2\right] \geq \mathbf{Pr}\left[E_1 = E_2 = E_3 = 1\right] \geq \left(\frac{\binom{(k+\ell)/2}{p_1/2}^2}{\binom{k+\ell}{p_1}}\right)^2 \Gamma^{-1} \ .$$

For the asymptotics, simply set $\Gamma := 2^{\tau n}$ for some arbitrary $\tau > 0$. By using the standard approximation Eq.(2.15), it is easy to see that

$$\lim_{n\to\infty} \frac{1}{n} \log\left(\frac{\binom{(R+L)n/2}{P_1 n/2}^2}{\binom{(R+L)n}{P_1 n}}\right) = 0 \ .$$

We eventually obtain

$$\lim_{n\to\infty} \frac{1}{n} \log(P(\mathbf{e})^{-1}) \geq \tau$$

for all but a $\frac{1}{2^{\tau n}-1} = 2^{-\tau n + o(n)}$ fraction of input matrices $\mathbf{Q}$, i.e. BasicReps is τ-solution preserving for almost all $\mathbf{Q}$. □

Remark 6.2.5 (Choice of r). Note that the proof of Lemma 6.2.4 requires $|\mathcal{C}(\mathbf{e})| = \rho \geq 2^r$ in order to apply Lemma 6.2.3. Since the running time of a single iteration is clearly decreasing in r and it is superfluous to let more than one representation survive, one should always choose $r = \lfloor \log \rho \rfloor$ maximal. However, choosing $r \gg \log \rho$ further decreases the workload per iteration and might allow for better running times. To adapt the proof of Lemma 6.2.4, one has to modify the choice of Γ to $\Gamma = 2^{\tau n + r - \log \rho}$ (note that Lemma 6.2.3 can also be applied for $|\mathcal{B}| = 2^{cm}$ for some $c < 1$ yielding a similar statement that holds for a $\frac{2^{(1-c)m}}{\Gamma - 1}$ fraction of input matrices). With $\mathfrak{P} = \lim \frac{1}{n} \log \rho$ and $r := \hat{\mathfrak{P}} n$ for some $\hat{\mathfrak{P}} > \mathfrak{P}$ the resulting variant of BasicReps becomes $(\tau + \hat{\mathfrak{P}} - \mathfrak{P})$-solution preserving. As further discussed in Remark 6.2.7, choosing $r \gg \log \rho$ does *not* improve the overall running time (yet might allow for a better time-memory trade-off).

Runtime Analysis and Main Theorem

It remains to analyse the runtime of BasicReps which can be done in a straightforward way, similarly to the proof of Theorem 5.2.10 and Theorem 4.3.1 for BCD. Recall that for $\alpha = \frac{1}{2}$

$$\Sigma(R, L, P, \Delta) = (R + L)\, \mathrm{H}\left(\frac{\frac{P}{2} + \Delta}{R + L} \right) \tag{6.11}$$

can be used to define the size of the base lists as $\frac{\Sigma}{2}$ and the number of representation is given by

$$\mathfrak{P}(R, L, P, \Delta) = P + (R + L - P)\, \mathrm{H}\left(\frac{\Delta}{R + L - P} \right) . \tag{6.12}$$

Since BasicReps is only well-defined for $\mathfrak{P} \leq L$ and both $\mathfrak{P}$ and Σ are symmetric in Δ (around $\frac{R+L-P}{2}$), we may always assume

$$\Delta < \frac{R + L - P}{2} . \tag{6.13}$$

Note that the strict inequality holds since $\Delta = \frac{R+L-P}{2}$ implies $\mathfrak{P} = R + L$ (in contradiction to $\mathfrak{P} \leq L$ and $0 < R < 1$). Also recall from Section 5.2 that the size of the merged list is given by

$$\Lambda(R, L, P, \Delta) = \Sigma(R, L, P, \Delta) - \frac{1}{2} \mathfrak{P}(R, L, P, \Delta) . \tag{6.14}$$

Lemma 6.2.6. *Let $0 < R < 1$. For $0 \leq L \leq 1 - R$, $0 \leq P \leq R + L$, $0 \leq \Delta < \frac{R+L-P}{2}$ and $\mathfrak{P}(L, P, \Delta) \leq L$ with Σ, $\mathfrak{P}$ and Λ as defined in Eq.(6.11), (6.12) and (6.14) the following holds: For almost all $\mathbf{Q}$,* BasicReps *runs in time $2^{C(R,L,P,\Delta)n + o(n)}$ where*

$$C(R, L, P, \Delta) = \max \left\{ \frac{\Sigma}{2}, \Sigma - \mathfrak{P}, 2\Lambda - L \right\} .$$

Proof. According to Proposition 5.2.1 we need to bound the size of the lists $\mathcal{B}_{i,j}$ and $\mathcal{L}_i$ and the number of collisions between $\mathcal{L}_1$ and $\mathcal{L}_2$, denoted by M (note that the number of matchings between the $\mathcal{B}_{i,j}$ does not differ from the size of the merged lists $\mathcal{L}_i$ due to the support-disjoint splitting). Obviously it holds

$$\lim_{n\to\infty}\frac{1}{n}\log|\mathcal{B}_{i,j}| = \frac{R+L}{2}\,\mathrm{H}\left(\frac{\frac{P}{2}+\Delta}{R+L}\right) = \frac{\Sigma}{2}\ .$$

Moreover, similarly to Lemma 2.4.6, one computes

$$\mathbb{E}\left[|\mathcal{L}_i|\right] = \frac{|\mathcal{B}_{i,1}\times\mathcal{B}_{i,2}|}{2^r}$$

and $\mathrm{Var}\left[|\mathcal{L}_i|\right] \le \mathbb{E}\left[|\mathcal{L}_i|\right]$. Asymptotically, this gives $\mathbb{E}\left[|\mathcal{L}_i|\right] = 2^{(\Sigma-\mathfrak{P})n+o(n)}$. Apply Vandermonde's identity, see Lemma 2.3.2, which yields

$$\Sigma = (R+L)\,\mathrm{H}\left(\frac{\frac{P}{2}+\Delta}{R+L}\right) > P-(R+L-P)\,\mathrm{H}\left(\frac{\Delta}{R+L-P}\right) = \mathfrak{P} \tag{6.15}$$

where the strict inequality holds due to Eq.(6.13). Thus, $\mathbb{E}\left[|\mathcal{L}_i|\right]$ is exponentially increasing in n and we obtain

$$\mathbf{Pr}\left[|\mathcal{L}_i| \ge 2\mathbb{E}\left[|\mathcal{L}_i|\right]\right] \le \frac{1}{\mathbb{E}\left[|\mathcal{L}_i|\right]}$$

by Chebychev's inequality. In other words, $\lim_{n\to\infty}\frac{1}{n}\log|\mathcal{L}_i| = \Sigma-\mathfrak{P}$ with probability $1-2^{-\Omega(n)}$. From know on let us assume $|\mathcal{L}_i| = 2^{(\Sigma-\mathfrak{P})n+o(n)}$. Now, by construction of the lists $\mathcal{L}_i$, every pair $(\mathbf{e}_1,\mathbf{e}_2)\in\mathcal{L}_1\times\mathcal{L}_2$ already matches $\mathbf{s}$ on r coordinates. Thus every pair yields a matching on all ℓ coordinates with probability $2^{r-\ell}$ and the expected number of matchings becomes

$$\mathbb{E}\left[M\right] = \frac{|\mathcal{L}_1\times\mathcal{L}_2|}{2^{\ell-r}} = 2^{(2(\Sigma-\mathfrak{P})-L+\mathfrak{P})n+o(n)} = 2^{(2\Lambda-L)n+o(n)}$$

since $\mathfrak{P}\le L$. Analogously to the proof of Theorem 5.2.10 one obtains $\lim_{n\to\infty}\frac{1}{n}\log M = \max\{2\Lambda-L,\epsilon\}$ for $\epsilon>0$ arbitrarily small with probability $1-2^{-\Omega(n)}$ (by distinguishing the cases $L<2\Lambda$ and $L\ge 2\Lambda$). This finishes the proof by Proposition 5.2.1. □

By modifying the overall algorithm in the same way as presented in the proof of Theorem 4.3.1, that is by simply aborting BasicReps in every bad iteration where too many collisions occur, the proof of the next important result immediately follows from Lemma 5.1.9, Lemma 6.2.4 and Lemma 6.2.6.

Main Theorem 6.2.1 (BasicReps). *Let* $0 < R < 1$, $0 < W \leq \mathrm{D}_{\mathsf{GV}}(R)$ *and* $\alpha = \frac{1}{2}$. *For almost all binary* $[n, \lfloor Rn \rfloor]$*-codes it holds: generalised ISD instantiated with* BasicReps *successfully terminates in time* $2^{F_{\mathsf{BReps}}(R,W)n+o(n)}$ *for every error vector* $\mathbf{e} \in W_{n,\lfloor Wn \rfloor}$ *with overwhelming probability where*

$$F_{\mathsf{BReps}}(R,W) = \min_{P,L,\Delta}\{N(R,W,L,P) + \tau + \max\left\{\frac{\Sigma(L,P,\Delta)}{2}, \Sigma(L,P,\Delta) - \mathfrak{P}(L,P,\Delta), 2\Lambda(L,P,\Delta) - L\right\}\}$$

with $0 \leq L \leq 1-R$, $\max\{0, R+L+W-1\} \leq P \leq \min\{R+L, W\}$, $0 \leq \Delta < \frac{R+L-P}{2}$, $0 \leq \mathfrak{P} \leq L$ *and* $\tau > 0$ *arbitrarily small.*

Note that the resulting algorithm has space complexity $S_{\mathsf{BReps}} = \max\left\{\frac{\Sigma}{2}\ ,\ \Sigma - \mathfrak{P}\right\}$ since the final list $\mathcal{L}$ must not be entirely stored within the overall framework, i.e. every single candidate solution can be extended on the fly.

Remark 6.2.7. In Remark 6.2.5 we discussed the possibility of choosing $r \gg \log \rho$ yielding a $(\tau + \hat{\mathfrak{P}} - \mathfrak{P})$-solution preserving variant of BasicReps. Using larger r affects the second and third term in the maximum of F_{BReps}, i.e. $\Sigma - \mathfrak{P}$ becomes $\Sigma - \hat{\mathfrak{P}}$ and $2\Lambda - L = 2\Sigma - \mathfrak{P} - L$ becomes $2\Sigma - \hat{\mathfrak{P}} - L$. Altogether, the workload per iteration can maximally be decreased by a factor $\hat{\mathfrak{P}} - \mathfrak{P}$ which will always be compensated by the increased number of iterations, thus fixing $r = \log \rho$ is reasonable without loss of generality. However, the space complexity might be reduced for different choices of r.

Numerical Optimisation and Comparison

Since the proof of superiority in the following section gives no indication about the quantitative improvement of BasicReps over Ball-Collision, we additionally provide numerically optimised parameters (see Table 6.2) and conclude with the interpolated complexity curve (see Figure 6.2) whose worst-case complexity $\max_R F(R) = 0.1054$ is achieved for $R = 0.4277$.

R	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
L	.0463	.0862	.1099	.1249	.1361	.1346	.1251	.1006	.0603
P	.0148	.0268	.0323	.0347	.0359	.0333	.0287	.0207	.0103
Δ	.0010	.0019	.0026	.0032	.0036	.0038	.0037	.0031	.0019
F_{Breps}	.0545	.0832	.0989	.1051	.1035	.0952	.0807	.0602	.0337
S_{Breps}	.0232	.0432	.0553	.0626	.0681	.0674	.0628	.0505	.0302

Table 6.1: Optimal parameters L, P, Δ for BasicReps for $W = \mathrm{D}_{\mathsf{GV}}(R)$ and resulting time and space complexities.

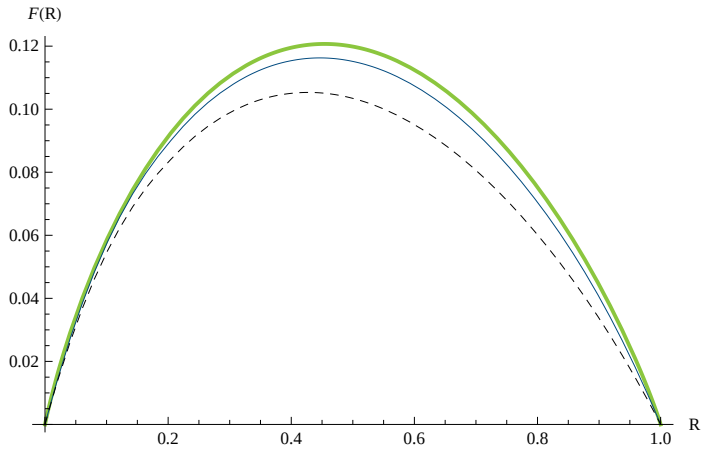


Figure 6.2: F_{Pra} (thick curve), interpolated F_{Ball} (thin curve) and F_{BReps} (dashed curve) for $W = \mathrm{D}_{\mathsf{GV}}(R)$.

6.3 A Formal Proof of Superiority

In this section we finally prove that BasicReps offers an exponential speed-up over BCD for all rates $0 < R < 1$. The outline of the proof is as follows: we start from optimal parameters for FS-ISD which offer at least the same performance as BCD by Theorem 5.2.13. We can then apply the parameter transformation of Theorem 5.2.26 yielding a sampling-based algorithm with the same performance. The same parameters can be used for BasicReps and achieve at least the same performance. In fact, they will predominately achieve the exact same running time, but in this case slightly increasing the parameter L allows to further improve the running time of BasicReps.

Main Theorem 6.3.1. *For all $0 < R < 1$, all $0 < W \leq \mathrm{D}_{\mathsf{GV}}(R)$ and almost all binary $[n, \lfloor Rn \rfloor]$-codes it holds:*

$$F_{\mathsf{BReps}}(R, W) < F_{\mathsf{Ball}}(R, W) \ .$$

Proof. Let $(\tilde{L}, \tilde{P}, \tilde{Q})$ be optimal BCD parameters. By Theorem 5.2.13, there must be optimal FS-ISD parameters (L^*, P^*) with

$$T_{\mathsf{FS-ISD}}(L^*, P^*) \leq T_{\mathsf{Ball}}(\tilde{L}, \tilde{P}, \tilde{Q}) \tag{6.16}$$

and it holds

$$0 < L^* < 1 - R \ , \tag{6.17}$$

$$W - P^* < \frac{1 - R - L^*}{2} \ , \tag{6.18}$$

$$0 < P^* < \frac{R + L^*}{2} \tag{6.19}$$

due to the results from Section 5.2.3. Applying the transformation ϕ from Theorem 5.2.26 yields

$$\Delta^* := \phi(L^*, P^*) = \frac{1}{2}(R + L^* - P^* - \sqrt{(R + L^*)(R + L^* - 2P^*)} > 0$$

which is well-defined due to Eq.(6.19). We denote the running time of the resulting sampling-based algorithm and BasicReps by T_{Sam} and T_{BReps}, respectively, i.e.

$$T_{\mathsf{Sam}}(L, P, \Delta) = N(L, P) + \max\{\Lambda(L, P, \Delta), 2\Lambda(L, P, \Delta) - L\}$$

and

$$\begin{aligned} T_{\mathsf{BReps}}(L, P, \Delta, \tau) =& N(L, P) \\ &+ \max\{\frac{\Sigma(L, P, \Delta)}{2}, \Sigma(L, P, \Delta) - \mathfrak{P}(L, P, \Delta), 2\Lambda(L, P, \Delta) - L\} + \tau \end{aligned}$$

with $\Lambda(L, P, \Delta) = \Sigma(L, P, \Delta) - \frac{1}{2}\mathfrak{P}(L, P, \Delta)$ and $\tau > 0$ arbitrarily small. In the following, we assume $\tau = 0$ for ease of presentation: For every parameter set (L', P', Δ') with $T_{\mathsf{Breps}}(L', P', \Delta', 0) < T_{\mathsf{Sam}}(L^*, P^*, \Delta^*)$ there will be some small enough τ preserving the strict inequality. Since $N(L, P)$ is identical for both algorithms, we focus on the respective maxima in T_{Sam} and T_{BReps}. Also recall that the parameter space for both algorithms is defined by the constraints

$$\begin{aligned} 0 \leq\ & L \leq 1 - R \\ \max\{0, R + L + W - 1\} \leq\ & P \leq \min\{W, R + L\} \\ 0 \leq\ & \Delta < \frac{R + L - P}{2} \end{aligned}$$

and by the additional constraint $0 \leq \mathfrak{P}(L, P, \Delta) \leq L$ for BasicReps. Clearly, since (L^*, P^*) are valid FS-ISD parameters and remain unchanged, the first two constraints are always fulfilled. Moreover, it holds $\Delta^* < \frac{1}{2}(R + L^* - P^*)$ due to Eq.(6.19). The only critical point is that the constraint $\mathfrak{P}(L^*, P^*, \Delta^*) \leq L^*$ might be violated yielding an invalid parameter set for BasicReps. We show how to deal with this problem at the end of this proof. For the moment, suppose $\mathfrak{P}(L^*, P^*, \Delta^*) \leq L^*$. Since ϕ preserves the list size, i.e. $\Lambda_{\mathsf{FS-ISD}}(L^*, P^*) = \Lambda(L^*, P^*, \Delta^*)$, we can apply Corollary 5.2.23 to obtain $\Lambda(L^*, P^*, \Delta^*) \leq L^*$, i.e. the maximum in T_{Sam} becomes $\Lambda(L^*, P^*, \Delta^*)$. Moreover, $0 < P^*$ implies $\mathfrak{P}(L^*, P^*, \Delta^*) > 0$ and thus

$$\Sigma(L^*, P^*, \Delta^*) - \mathfrak{P}(L^*, P^*, \Delta^*) < \Sigma(L^*, P^*, \Delta^*) - \frac{\mathfrak{P}(L^*, P^*, \Delta^*)}{2} = \Lambda(L^*, P^*, \Delta^*) \ .$$

Consequently, we do not need to care about the second argument of the maximum in T_{BReps}. As shown in the proof of Lemma 6.2.6, see Eq.(6.15), it holds $\Sigma(L^*, P^*, \Delta^*) > \mathfrak{P}(L^*, P^*, \Delta^*)$ which implies

$$\frac{\Sigma(L^*, P^*, \Delta^*)}{2} < \Sigma(L^*, P^*, \Delta^*) - \frac{\mathfrak{P}(L^*, P^*, \Delta^*)}{2} = \Lambda(L^*, P^*, \Delta^*) \ .$$

Thus, we must not be concerned about the first argument of the maximum in T_{BReps} either. Let us assume that the maximum in T_{BReps} is actually $2\Lambda(L^*, P^*, \Delta^*) - L^*$. Note that $T_{\mathsf{BReps}}(L^*, P^*, \Delta^*) = N(L^*, P^*) + 2\Lambda(L^*, P^*, \Delta^*) - L^*$ which motivates to define a slightly increased parameter $L' = L^* + \epsilon$ for some $\epsilon > 0$ in order to get an improved running time: By continuity of Σ and $\mathfrak{P}$, the maximum in T_{BReps} is still $2\Lambda - L'$ for small enough ϵ. More formally, one computes

$$\frac{\partial T}{\partial L}(L^*, P^*, \Delta^*) = -1 - 2\log\left(\frac{2(R + L^* - \Delta^*) - P^*}{2(L^* + R)}\right) + \log\left(1 - \frac{P^*}{L^* + R}\right) + \log\left(1 - \frac{\Delta^*}{L^* + R - P^*}\right) - \log\left(\frac{1 - R - L^* - W + P^*}{1 - L^* - R}\right)$$

which is negative if

$$\frac{4(L^* + R - P^* - \Delta^*)(1 - L^* - R)(L^* + R)}{[2(R + L^* - \Delta^*) - P^*]^2(1 - L^* - R + P^* - W)} < 2 \ .$$

By definition of Δ^* this is equivalent to

$$\frac{1 - R - L^*}{1 - R - L^* - W + P^*} < 2$$

which holds for optimal FS-ISD parameters due to Eq.(6.18). Thus (L^*, P^*, Δ^*) can not be optimal and changing L^* to $L' = L^* + \epsilon$ for $\epsilon > 0$ small enough yields a parameter set (L', P^*, Δ^*) with

$$\begin{aligned} F_{\mathsf{BReps}}(R, W) \le T_{\mathsf{BReps}}(L', P^*, \Delta^*) &< T_{\mathsf{Sam}}(L^*, P^*, \Delta^*) \\ &= T_{\mathsf{FS\text{-}ISD}}(L^*, P^*) \le F_{\mathsf{Ball}}(R, W) \ . \end{aligned}$$

It remains to deal with the case where $\mathfrak{P}(L^*, P^*, \Delta^*) > L^*$. Loosely speaking, this means that the parameter L^* does not offer sufficient space for BasicReps to eliminate as much representations (in a constructive way) as possible. Clearly, one could still run BasicReps with an adjusted choice $r = \ell$, i.e. one eliminates as much representations as allowed by L^*, but this would result in a larger complexity for merging the two lists $\mathcal{L}_1$ and $\mathcal{L}_2$, i.e. *every* pair in $\mathcal{L}_1 \times \mathcal{L}_2$ would yield a matching. More formally the maximum in T_{BReps} would become

$$\max\{\frac{\Sigma(L, P, \Delta)}{2}, \Sigma(L, P, \Delta) - L, 2\Sigma(L, P, \Delta) - 2L\}$$

and $2\Lambda - L = 2\Sigma - \mathfrak{P} - L < 2\Sigma - 2L$ implies that the third expression might become larger than the maximum in T_{Sam} which would completely destroy the above proof (slightly increasing L^* might become insufficient). To fix this problem, consider the following variant of BasicReps: Let $\delta > 0$ be the gap between $\mathfrak{P}$ and L^*, i.e. $\mathfrak{P}(L^*, P^*, \Delta^*) + \delta = L^*$.

- Construct the lists $\mathcal{L}_1$ and $\mathcal{L}_2$ with constraints of size $\ell = \lfloor L^* n \rfloor$.
- Randomly discard a $\frac{1}{2^{\lfloor \frac{\delta}{2} \rfloor n}}$-fraction of both lists.

Note that the lists constructed in the first step will have $2^{(\mathfrak{P}-L^*)n+o(n)} = 2^{\delta n+o(n)}$ remaining representations in $\mathcal{L}_1 \times \mathcal{L}_2$ for every candidate solutions (on average), thus randomly eliminating a $2^{\frac{\delta}{2}n+o(n)}$-fraction independently in both lists keeps one representation with good probability (this argument could be turned into a rigorous proof analogously to Section 5.2.1 where we explained how to choose the size of the lists for a purely sampling-based approach). This variant offers the same complexity for constructing the lists $\mathcal{L}_1$ and $\mathcal{L}_2$. Since we eliminate additional elements, the size coefficient of these lists becomes $\Sigma - L^* - \frac{\delta}{2}$ which reduces the number of matchings to $2\Sigma - 2L^* - \delta = 2\Sigma - \mathfrak{P} - L^* = 2\Lambda - L^*$, i.e. the third expression in the maximum of T_{BReps} has the desired form. This finally allows to establish exactly the same proof for the modified BasicReps algorithm as for the regular version. □

Remark 6.3.1. The special case $\mathfrak{P}(L^*, P^*, \Delta^*) > L^*$ might be completely excluded by a refined analysis of the space of optimal parameters (L^*, P^*). In particular, we computed optimal FS-ISD parameters (L^*, P^*) for various $0 < R < 1$ and checked the resulting values for $\Delta^* = \phi(L^*, P^*)$ which never violated the condition $\mathfrak{P}(L^*, P^*, \Delta^*) \leq L^*$.

6.4 Using Representations Iteratively

It is possible to further improve our ISD algorithm by recursively using BasicReps to compute the intermediate lists $\mathcal{L}_1$ and $\mathcal{L}_2$: Recall that these intermediate lists are defined to contain all $\mathbf{e}_1$ and $\mathbf{e}_2$ of weight $p_1 = \frac{p}{2} + \delta$ whose $\mathbf{Q}$-column-sum matches the random target vector $\mathbf{t}_1$ and $\mathbf{t}_2 = \mathbf{s} + \mathbf{t}_1$, respectively, and we computed those lists by merging appropriate support-disjoint base lists. Alternatively, one could simply view the problem of computing $\mathcal{L}_1$ (and similarly $\mathcal{L}_2$) as the initial problem of solving Eq.(6.1) with input matrix $\mathbf{Q} := \mathbf{Q}_{[r]} \in \mathbb{F}_2^{r \times k+\ell}$, "syndrome" $\mathbf{s} := \mathbf{t}_1$ and target weight p_1. Thus, one might invoke BasicReps to solve this problem, i.e. one computes $\mathcal{L}_1 \leftarrow \text{BasicReps}(\mathbf{Q}_{[r]}, \mathbf{t}_1, p_1, \delta_2)$ and similarly $\mathcal{L}_2$ by replacing $\mathbf{t}_1$ with $\mathbf{t}_2$. This implicitly means to introduce another level of representations and there will be

$$\rho_2(k, \ell, p_1, \delta_2) = \binom{p_1}{\frac{p_1}{2}} \binom{k + \ell - p_1}{\delta_2}$$

many representations $(\mathbf{e}_{1,1}, \mathbf{e}_{1,2})$ for every representation $\mathbf{e}_1$ of $\mathbf{e}$. Here, $\delta_2 \leq \frac{k+\ell-p_1}{2}$ is a new parameter that allows to control the number of representations on the second level. Unravelling BasicReps yields an algorithm that can be described as a computation tree of depth three as illustrated in Figure 6.3.

We enumerate the layers from bottom to top, i.e. the third layer identifies the initial computation of disjoint base lists $\mathcal{B}_1$ and $\mathcal{B}_2$ (as implicitly done by BasicReps) and the zero layer identifies the final output list $\mathcal{L}$. Note that BasicReps computes the list $\mathcal{L}_1^{(1)}$ by merging the two intermediate lists $\mathcal{L}_1^{(2)}$ and $\mathcal{L}_2^{(2)}$ whose elements already match randomly chosen target vectors $\mathbf{t}_1^{(2)}$ and $\mathbf{t}_2^{(2)} = \mathbf{t}_1 + \mathbf{t}_1^{(2)}$ on $r_2 = \lfloor \log \rho_2 \rfloor$ coordinates, as illustrated below.

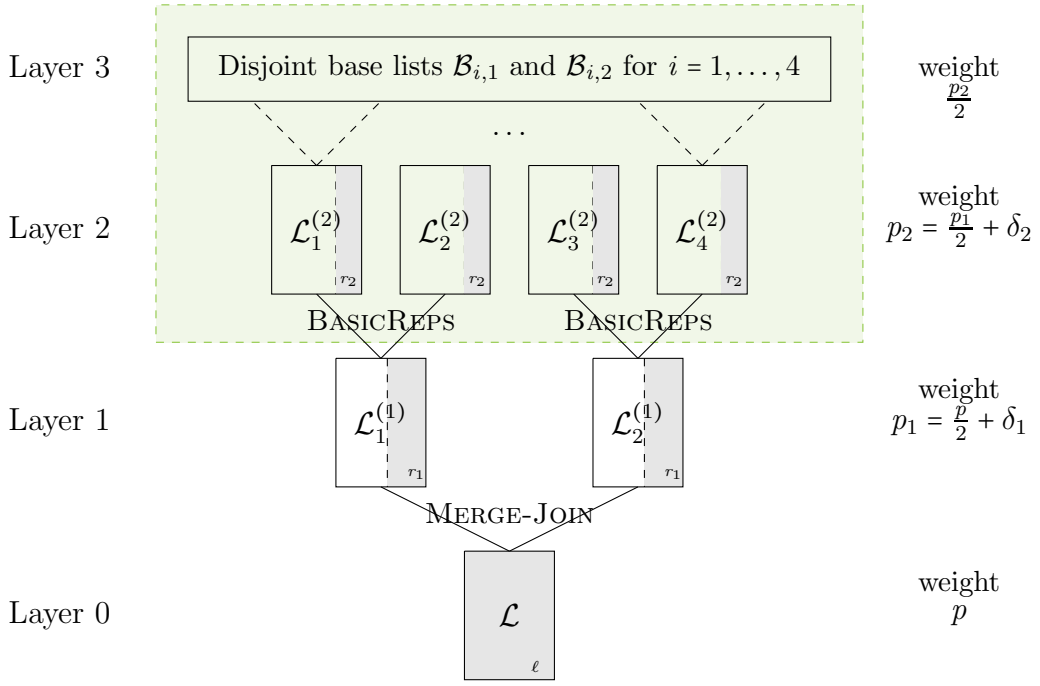


Figure 6.3: Unravelled computation tree of the ColumnMatch algorithm. The gray area and the parameter in the bottom right corner of a list represent the amount of matched coordinates. The framed second and third layer belong to the recursive call of BasicReps.

We call the resulting algorithm Column-Match since it resembles the eponymous algorithm presented in [BJMM12]. Its formal description is given in Algorithm 7. Compared to [BJMM12], where the algorithm is constructed in a direct way without initially defining BasicReps, the approach presented in this thesis offers a simplified analysis (based on the preceding results) and is easier to extend to more levels, see Section 6.5.

Algorithm 7: Column-Match

input : Parameters p, δ_1 and δ_2. Matrix $\mathbf{Q} \in \mathbb{F}_2^{\ell \times (k+\ell)}$ and syndrome $\mathbf{s} \in \mathbb{F}_2^{\ell}$.
output : List $\mathcal{L}$ of candidate solutions for Eq.(6.1).
params: Number of representations $\rho_1 := \rho(k, \ell, p, \delta_1)$ (and $\rho_2 := \rho(k, \ell, p_1, \delta_2)$ implicit in BasicReps) according to Eq.(5.8), $r_i := \lfloor \log \rho_i \rfloor$ such that $0 \le r_2 \le r_1 \le \ell$. Weight $p_1 := \frac{p}{2} + \delta_1$ (and $p_2 = \frac{p_1}{2} + \delta_2$ implicit in BasicReps).

Choose random target $\mathbf{t}_1 \in_R \mathbb{F}_2^{r_1}$ and set $\mathbf{t}_2 = \mathbf{s}_{[r_1]} + \mathbf{t}_1$;
for $i = 1$ **to** 2 **do**
  $\mathcal{L}_i \leftarrow \text{BasicReps}(\mathbf{Q}_{[r_1]}, \mathbf{t}_i, p_1, \delta_2)$
$\mathcal{L} \leftarrow \text{Merge-Join}(\mathcal{L}_1, \mathcal{L}_2, p, \mathbf{Q}, \mathbf{s})$;
return $\mathcal{L}$;

The analysis of Column-Match is similar to the analysis of BasicReps and is based on the following two statements.

Lemma 6.4.1. *For almost all input matrices* $\mathbf{Q}$*,* Column-Match *is* τ*-solution preserving for arbitrary* $\tau > 0$*.*

Proof. Let $\mathbf{e}$ be a fixed solution to Eq.(6.1). Define $\mathcal{C}(\mathbf{e})$ as the set of all $\mathbf{e}_1 \in W_{k+\ell,p_1}$ consistent to $\mathbf{e}$, i.e. $|\mathcal{C}(\mathbf{e})| = \rho_1 = \rho(k,\ell,p,\delta_1)$. Note that a single representation $(\mathbf{e}_1,\mathbf{e}_2)$ is contained in $\mathcal{L}_1 \times \mathcal{L}_2$ if the following three independent events occur:

E_1: $(\mathbf{Q}\mathbf{e}_1)_{[r]} = \mathbf{t}_1$ (where $\mathbf{t}_1$ is the random target vector corresponding to list $\mathcal{L}_1$).

E_2: $\mathbf{e}_1$ is found by BasicReps.

E_3: $\mathbf{e}_2$ is found by BasicReps.

Note, that E_1 only depends on the random choice of $\mathbf{t}_1$ and both E_2 and E_3 depend on the independent random choices of BasicReps (i.e. the choice of additional random target vectors and partitions). Similarly to the proof of Lemma 6.2.4, we can apply Lemma 6.2.3 to obtain

$$\lim_{n\to\infty} \frac{1}{n} \log \mathbf{Pr}\left[E_1 = 1\right]^{-1} \geq \tau'$$

for some arbitrarily small $\tau' > 0$ and almost all $\mathbf{Q}$. Moreover, Lemma 6.2.4 guarantees that BasicReps is τ''-solution preserving, i.e.

$$\lim_{n\to\infty} \frac{1}{n} \log \mathbf{Pr}\left[E_i = 1\right]^{-1} \geq \tau''$$

for $i = 2,3$, some arbitrarily small τ'' and almost all $\mathbf{Q}$. Now, the claim follows by independence of the events E_i for $\tau := \tau' + 2\tau''$ arbitrarily small. □

For the runtime analysis, it is important to estimate the actual size of the intermediate lists $\mathcal{L}_1^{(1)}$ and $\mathcal{L}_2^{(1)}$ on the first layer within the computation tree. Recall that these lists are output by BasicReps which filters out weight-inconsistent and duplicate matchings. Thus, the size of these lists differs exponentially from the number of matchings that occur during their creation. We have

$$|\mathcal{L}_i^{(1)}| \ll \frac{|\mathcal{L}_{2i-1}^{(2)} \times \mathcal{L}_{2i}^{(2)}|}{2^{r_2-r_1}} \quad .$$

Since the lists $\mathcal{L}_i^{(1)}$ contain vectors of length $k+\ell$ and weight p_1 that fulfil a random constraint on r_1 coordinates, we have the following upper bound

$$\mathbb{E}\left[|\mathcal{L}_i^{(1)}|\right] \leq \frac{\binom{k+\ell}{p_1}}{2^{r_1}} \tag{6.20}$$

which will be used to prove the next lemma. Note that this is bound is *asymptotically tight* when r_1 and r_2 are *fixed* to the respective number of representations, see below for a more detailed discussion about how to choose r_1 and r_2. For ease of presentation we define $P_0 := P$, $P_1 := \frac{P}{2} + \Delta_1$ and

$$\begin{aligned}\mathfrak{P}_i &:= \mathfrak{P}(L, P_{i-1}, \Delta_i) \ , \\ \Sigma_i &:= \Sigma(L, P_{i-1}, \Delta_i)\end{aligned}$$

where Σ and $\mathfrak{P}$ are as defined in Eq.(6.11) and (6.12).

Lemma 6.4.2. *Let* $0 < R < 1$. *For* $0 \le L \le 1 - R$, $0 \le P \le R + L$, $0 \le \Delta_i < \frac{R+L-P_i}{2}$ *and* $0 \le \mathfrak{P}_2 \le \mathfrak{P}_1 \le L$ *the following holds: For almost all* $\mathbf{Q}$, Column-Match *runs in time* $2^{C(R,L,P,\Delta_1,\Delta_2)n+o(n)}$ *where*

$$C(R, L, P, \Delta_1, \Delta_2) = \max\left\{\frac{\Sigma_2}{2}, \Sigma_2 - \mathfrak{P}_2, 2\Sigma_2 - \mathfrak{P}_2 - \mathfrak{P}_1, 2\Sigma_1 - \mathfrak{P}_1 - L\right\} \ .$$

Proof. The first three arguments of the maximum follow immediately from Lemma 6.2.6 since we applied BasicReps with $P' = P_1$ and $\Delta' = \Delta_2$. Note that the parameter L in the last expression of the maximum in Lemma 6.2.6 has to be replaced by $\mathfrak{P}_1$ since the input matrix and the target vector consist of r_1 rows, i.e. we obtain $2\Sigma(L, P_1, \Delta_2) - \mathfrak{P}(L, P_1, \Delta_2) - \mathfrak{P}(L, P, \Delta_1)$ as claimed. By approximating Eq.(6.20) (and by the standard argument about the concentration of the actual list size around its expectation as similarly done in the preceding proofs, e.g. Lemma 6.2.6 for BasicReps or Theorem 4.3.1 for BCD) we obtain

$$\Lambda := \lim_{n\to\infty} \frac{1}{n} \log |\mathcal{L}_i^{(1)}| = (R + L)\, \mathrm{H}\left(\frac{P_1}{R + L}\right) - \mathfrak{P}_1 = \Sigma_1 - \mathfrak{P}_1$$

with probability $1 - 2^{-\Omega(n)}$ (note that Λ is smaller than $2\Sigma_2 - \mathfrak{P}_2 - \mathfrak{P}_1$ which represents *all* matchings that occur within BasicReps, i.e. Λ does not need to occur in C). Now, similarly to the proof of Lemma 6.2.6, the coefficient for the number of matchings when merging $\mathcal{L}_1^{(1)}$ and $\mathcal{L}_1^{(2)}$ is given by

$$2\Lambda + \mathfrak{P}_1 - L = 2\Sigma_1 - \mathfrak{P}_1 - L$$

with probability $1 - 2^{\Omega(n)}$. This finishes the proof by Lemma 6.1.1. □

The main result follows as usual by simply aborting the computation of Column-Match in every bad iteration and by combining the above two lemmata together with Lemma 5.1.9. In contrast to the preceding algorithms, there are two abort criteria for every invocation of Column-Match. Namely, either the size of the lists on level one (or equivalently: the number of collisions that occur when merging two different lists on the second level) or the number of collision between $\mathcal{L}_1^{(1)}$ and $\mathcal{L}_2^{(1)}$ becomes too large. As the proof of Lemma 6.4.2 shows, both events occur with negligible probability $2^{-\Omega(n)}$. Consequently, a single iteration (and thus the first good iterations w.r.t. a fixed solution $\mathbf{e}$) succeeds with probability $1 - 2^{-\Omega(n)}$.

Main Theorem 6.4.1 (Column-Match). *Let* $0 < R < 1$, $0 < W \leq \mathrm{D}_{\mathsf{GV}}(R)$ *and* $\alpha = \frac{1}{2}$. *For almost all binary* $[n, \lfloor Rn \rfloor]$*-codes it holds: generalised ISD instantiated with* Column-Match *successfully terminates in time* $2^{F(R,W)n+o(n)}$ *for every error vector* $\mathbf{e} \in W_{n,\lfloor Wn \rfloor}$ *with overwhelming probability where*

$$F(R,W) = \min_{P,L,\Delta_1,\Delta_2} \{N(R,W,L,P) + \tau + \max\left\{\frac{\Sigma_2}{2} \ , \ \Sigma_2 - \mathfrak{P}_2 \ , \ 2\Sigma_2 - \mathfrak{P}_2 - \mathfrak{P}_1 \ , \ 2\Sigma_1 - \mathfrak{P}_1 - L\right\}\}$$

with $0 \leq L \leq 1-R$, $\max\{0, R+L+W-1\} \leq P \leq \min\{R+L, W\}$, $0 \leq \Delta_i < \frac{R+L-P_i}{2}$, $0 \leq \mathfrak{P}_2 \leq \mathfrak{P}_1 \leq L$ *and* $\tau > 0$ *arbitrarily small.*

The size of the intermediate lists $\mathcal{L}_i$ has been estimated in the proof of Lemma 6.4.2 and the overall algorithm has space complexity $S = \max\left\{\frac{\Sigma_2}{2}, \Sigma_2 - \mathfrak{P}_2, \Sigma_1 - \mathfrak{P}_1\right\}$.

6.4.1 Choice of r_1 and r_2

In our description of the improved algorithm, we fixed the parameters r_1 and r_2 naturally according to the respective number of representations ρ_1 and ρ_2 which enabled us to *rigorously* prove that Column-Match is τ-solution preserving.

However, it is in principle possible to phrase the algorithm for any choice of r_1 and r_2 under the natural constraint $0 \leq r_2 \leq r_1 \leq \ell$. Put differently, one can define an extended parameter space which might even allow to find an improved complexity coefficient. Let us first exclude some special cases:

- As explained in Remark 6.2.5 it is always suboptimal to choose $r_2 \ll \log \rho_2$ (keeping one representation on the second layer for every representation contained in the first layer is sufficient).
- Without loss of generality $r_1 \leq \log \rho_1$ according to Remark 6.2.7 (choosing $r_1 = \log(\rho_1) + \varepsilon$ for some $\varepsilon > 0$ increases τ' in the proof of Lemma 6.4.1 and thus the number of iterations by ε whereas the workload per iteration is maximally decreased by the same factor).

It remains to consider the case $\log \rho_2 \ll r_2 \leq r_1 \ll \log \rho_1$. Unfortunately, allowing for such parameters complicates the algorithm's analysis in two aspects:

1. The estimate for the size of the merged lists $\mathcal{L}_i^{(1)}$ on the first layer according to Eq.(6.20) is no longer tight. Recall that these lists are computed by invoking BasicReps which implicitly removes weight-inconsistent solutions. In order to compute the actual size of $\mathcal{L}_1^{(1)}$ (and similarly the size of $\mathcal{L}_2^{(1)}$) one has to consider the probability that a matching $(\mathbf{e}_1, \mathbf{e}_2) \in \mathcal{L}_1^{(2)} \times \mathcal{L}_2^{(2)}$ is weight-consistent, i.e. one has to compute the probability that two random vectors $\mathbf{e}_1$ and $\mathbf{e}_2$ of length $k + \ell$

and weight $p_2 = \frac{p_1}{2} + \delta_2$ have a sum $\mathbf{e}_1 + \mathbf{e}_2$ of weight p_1. This probability is given by

$$\pi_1 := \frac{\binom{k+\ell}{p_1}\binom{p_1}{p_1/2}\binom{k+\ell-p_1}{\delta_2}}{\binom{k+\ell}{p_2}^2} \tag{6.21}$$

since we can first choose p_1 positions for the sum and then choose $\frac{p_1}{2}$ positions for $\tilde{\mathbf{e}}_1$ within the sum and δ_2 positions for $\tilde{\mathbf{e}}_1$ outside the sum (which determines $\tilde{\mathbf{e}}_2$). This gives the alternative estimate

$$\mathbb{E}\left[|\mathcal{L}_1^{(1)}|\right] = \pi_1 \frac{|L_1^{(2)}| \cdot |L_2^{(2)}|}{2^{r_1+r_2}} = \frac{\binom{k+\ell}{p_1}}{2^{r_1+r_2}} \underbrace{\binom{p_1}{p_1/2}\binom{k+\ell-p_1}{\delta_2}}_{=\rho_2} \tag{6.22}$$

which coincides with Eq.(6.20) if $r_2 = \log \rho_2$ and shrinks the list size by an exponential factor if $r_2 \gg \log \rho_2$.

2. The proof of Lemma 6.4.1 collapses, i.e. we do not know how to (rigorously) prove that Column-Match is τ-solution preserving. Heuristically, one might argue as follows: Let $\mathbf{t}_1$ be the random target vector chosen in a fixed iteration. Consider a fixed solution $\mathbf{e}$ and denote its set of *first-layer-representations* as $\mathcal{R}(\mathbf{e})$, i.e. $\mathcal{R}(\mathbf{e})$ contains pairs of vectors $(\mathbf{e}_1, \mathbf{e}_2)$ with $\mathbf{e} = \mathbf{e}_1 + \mathbf{e}_2$, $\mathrm{wt}(\mathbf{e}_i) = p_1$ and $(\mathbf{Q}\mathbf{e}_1)_{[r_1]} = \mathbf{t}_1$. Since $r_1 \ll \log(\rho_1)$, one expects $\mathcal{R}(\mathbf{e})$ to have exponential size $\sigma := \frac{\rho_1}{2^{r_1}}$. In order to find a single pair $(\mathbf{e}_1, \mathbf{e}_2)$, both $\mathbf{e}_1$ and $\mathbf{e}_2$ must be output by Basic-Reps. Since $r_2 \gg \log \rho_2$, this happens with exponentially small probability $\pi_2 := \left(\frac{\rho_2}{2^{r_2}}\right)^2$. If the probabilities π_2 for different pairs $(\mathbf{e}_1, \mathbf{e}_2)$ were *independent*, the probability of finding at least one of them would be

$$1 - (1 - \pi_2)^\sigma \geq 1 - \exp\left(-\frac{\rho_1 \rho_2^2}{2^{r_1+2r_2}}\right) \tag{6.23}$$

which can be made constant, e.g. by requiring $r_1 + 2r_2 = \log \rho_1 + 2 \log \rho_2$. However, the independence property is not fulfilled: Different representations $\mathbf{e}_1 \neq \tilde{\mathbf{e}}_1$ on the first layer may share a large number of representations on the second layer. For example set $k + \ell = 10$, $p = 4$, $\delta_1 = 2$ and $\delta_2 = 1$ and consider a solution $\mathbf{e} = (1,1,1,1,0,0,0,0,0,0)$. Suppose that the representations $\mathbf{e}_1 = (1,1,0,0,0,0,0,0,1,1)$ and $\tilde{\mathbf{e}}_1 = (1,0,0,1,0,0,0,0,1,1)$ (with appropriate $\mathbf{e}_2$ and $\tilde{\mathbf{e}}_2$) are both contained in $\mathcal{R}(\mathbf{e})$. Now, $\mathbf{e}_1$ and $\tilde{\mathbf{e}}_1$ share the same representation $\mathbf{e}_{1,1} = \tilde{\mathbf{e}}_{1,1} = (0,1,0,1,0,0,0,0,1,0)$ (with corresponding distinct counterparts $\mathbf{e}_{1,2}$ and $\tilde{\mathbf{e}}_{1,2}$) on the second layer.

Despite these difficulties, we numerically examined a variant of our algorithm where we fixed $r_1 + 2r_2 = \log \rho_1 + 2 \log \rho_2$ according to Eq.(6.23), i.e. it suffices to additionally optimize either r_1 or r_2. Taking Eq.(6.22) into account, the maximum in the complexity coefficient as given in Theorem 6.4.1 becomes $\max\{\frac{\Sigma_2}{2}, \Sigma_2 - \mathfrak{P}_2 - \varepsilon, 2\Sigma_2 - \mathfrak{P}_2 - \mathfrak{P}_1 + \varepsilon,$

$2\Sigma_1 - \mathfrak{P}_1 - L\}$ where $0 \le \varepsilon \le \mathfrak{P}_2$ is a new parameter representing the difference between r_2 and $\log \rho_2$. We did not obtain any improvement over the original variant with fixed r_1 and r_2 which further justifies our initial choice of parameters.

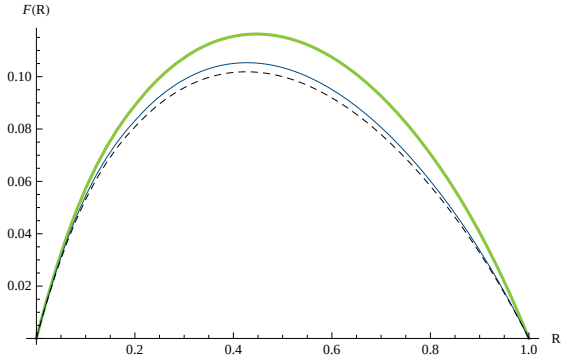


Figure 6.4: Interpolated F_{Ball} (thick curve), F_{BReps} (thin curve) and F (dashed curve).

6.4.2 Numerical Results and Comparison

For the sake of completeness we provide optimal parameter choices for different rates $0 < R < 1$ in Table 6.4.2 and the corresponding interpolated complexity coefficient in Figure 6.4. This yields an interpolated worst-case complexity of $\max_R F(R) = 0.1019$ for $R = 0.4263$.

R	**0.1**	**0.2**	**0.3**	**0.4**	**0.45**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
L	.1023	.1672	.1929	.2085	.2055	.2058	.1971	.1646	.1259	.0704
P	.0351	.0527	.0560	.0564	.0538	.0520	.0461	.0350	.0238	.0112
Δ_1	.0052	.0091	.0101	.0109	.0103	.0103	.0098	.0078	.0057	.003
Δ_2	.0013	.0023	.0020	.0018	.0014	.0013	.0011	.0006	.0003	.0001
F	.0532	.0808	.0958	.1017	.1017	.1000	.0920	.0779	.0583	.0328
S	.0345	.0570	.0690	.0767	.0777	.0786	.0767	.0665	.0524	.0306

Table 6.2: Optimal parameters L, P, Δ_1, Δ_2 for Column-Match for $W = \mathrm{D}_{\mathsf{GV}}(R)$ and resulting time and space complexities.

6.4.3 Time-Memory Trade-offs

Compared to Prange's algorithm, all improved ISD algorithms come at the cost of an exponential space complexity. Clearly, a large space complexity might be considered as a serious drawback, in particular for many practical applications. However, the space complexity issue has not been discussed in the literature at all. The main (and only) goal for designers of generic decoding algorithms is to reduce the running time as much as possible. In this section, we give a short comparison of the presented ISD

algorithms with respect to their time *and* space complexity. For this purpose, let T and S denote the time and space complexity coefficients of some ISD algorithm. Now, a natural measure might be to consider the product of time and space complexity, or in terms of complexity coefficients, to consider $T + S$ and to ask for optimal parameters *minimizing* this expression. Answering this question for various ISD algorithms is rather disillusioning: In general,

Prange's plain ISD yields the best time-space complexity product.

We only justify this statement for Ball-collision decoding: Recall that for BCD

$$T(L, P, Q) = N(L, P, Q) + \max\{\Lambda(L, P, Q), 2\Lambda(L, P, Q) - L\}$$

by Eq.(5.26). This yields

$$T(L, P, Q) + \Lambda(L, P, Q) = N(L, P, Q) + \max\{2\Lambda(L, P, Q), 3\Lambda(L, P, Q) - L\}.$$

Thus we always have

$$\begin{aligned} T(L, P, Q) + \Lambda(L, P, Q) &\geq N(L, P, Q) + 2\Lambda(L, P, Q) \\ &= \mathrm{H}(W) - (1 - R - L)\,\mathrm{H}\left(\frac{W - P - Q}{1 - R - L}\right) \end{aligned}$$

which is clearly minimal for $L = P = Q = 0$ (where BCD degenerates to Plain ISD). Consequently, comparing the improved algorithms with respect to minimal $T+S = T+\Lambda$ is meaningless. Even worse, it is conceivable that the improved algorithms might turn out to be "equivalent" in the following sense: If A and B are two ISD algorithms such that A's optimal running time coefficient is T_A (with corresponding space complexity S_A), do there exist parameters for B with $T_B + S_B < T_A + S_A$? For example, if one compares $T + S$ for the optimal values of Stern's algorithm, BCD and the improved algorithm of this section (which we will call BJMM in reference to [BJMM12]) for rate $R = 0.5$, see Table 4.1, 4.2 and 6.4.2, one obtains

$$\begin{aligned} T_{\mathsf{Stern}} + S_{\mathsf{Stern}} &= 0.14856 \\ T_{\mathsf{BCD}} + S_{\mathsf{BCD}} &= 0.15287 \\ T_{\mathsf{BJMM}} + S_{\mathsf{BJMM}} &= 0.1786\ . \end{aligned}$$

Note that Stern's algorithm seems to outperform all recent algorithms (and is itself clearly dominated by Prange's algorithm with coefficient 0.119865). Fortunately, both BCD and BJMM offer a lot of flexibility with respect to their memory consumption due to their large parameter spaces. Moreover, it is easy to search for optimal parameters while restricting the space complexity of the respective algorithm to a fixed upper bound, e.g. we computed optimal BCD or BJMM parameters that guarantee the same (or smaller) space complexity as Stern's algorithm. The resulting time-space complexity products and their interpolated curves can be found below.

R	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
Prange	.0585	.0914	.1106	.1196	.1199	.1125	.0977	.0753	.0443
Stern	.0688	.1087	.1332	.1458	.1487	.1423	.1269	.1016	.0635
BCD	.0687	.1084	.1329	.1455	.1484	.1421	.1267	.1014	.0634
BJMM	.0674	.1055	.1283	.1395	.1412	.1340	.1183	.0934	.0573

Table 6.3: Time-space complexity products for different ISD algorithms where parameters for Stern's algorithm are optimal w.r.t to the *running time* and BCD and BJMM parameters are optimal w.r.t. the running time with bounded (e.g. Stern's) space complexity.

We would like to stress that the above numerical approach only leads to results of no great significance. Note that optimal parameters for Stern's algorithm are not necessarily unique and there might be different optima with decreased space complexity (although we were unable to find such parameters numerically). In summary, we strongly conjecture that the recent algorithms indeed improve over Stern's algorithm, but a deeper theoretical analysis seems superfluous (in particular with the general superiority of Prange's algorithm in mind).

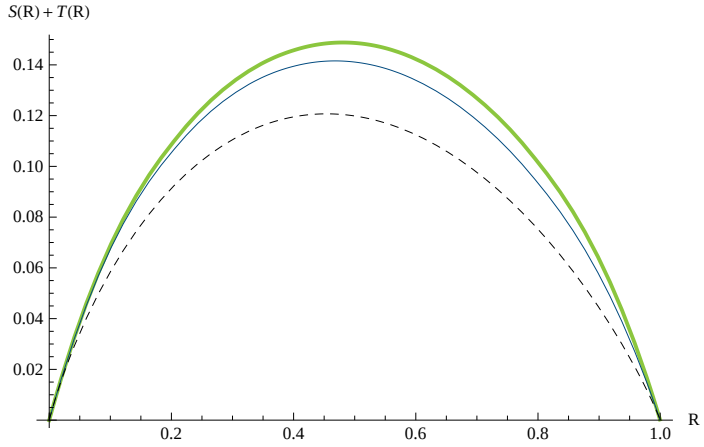


Figure 6.5: Interpolated $T + S$-curves for Stern's algorithm (thick curve), and BJMM (thin curve). Optimal time-space complexity of Prange's algorithm (dashed curve).

6.5 Discussion and Open Problems

We conclude this chapter with some open questions. The first one is not restricted to ISD algorithms and concerns the optimality of the representation technique in general. Note that one could easily generalise the Column-Match algorithm to more layers: On input $\mathbf{Q}$, $\mathbf{s}$ and p, recursively define algorithms Reps$[i]$ with parameters p and $\delta_1, \ldots, \delta_i$ by first computing

$$\mathcal{L}_j \leftarrow \text{Reps}[i-1](\mathbf{Q}_{[r_1]}, \mathbf{t}_j, p_1, \delta_2, \ldots, \delta_i)$$

for $j = 1, 2$ (where $\mathbf{t}_j$ are random target vectors of dimension r_1) and by outputting $\mathcal{L} \leftarrow \text{MERGE-JOIN}(\mathcal{L}_1, \mathcal{L}_2, p, \mathbf{Q}, \mathbf{s})$ where $\text{REPS}[1] \coloneqq \text{BASICREPS}$. The parameters are restricted as usual, i.e. we have $0 \le \delta_{j+1} < \frac{k+\ell-p_j}{2}$ where $p_{j+1} = \frac{p_j}{2} + \delta_{j+1}$ for $j = 0, \ldots, i-1$ with $p_0 \coloneqq p$. This implicitly defines different numbers of representations ρ_j on layers $j = 1, \ldots, i$ and corresponding parameters $r_j = \log \rho_j$. Analogously to Theorem 6.4.1, the resulting complexity coefficient is given by

$$F(R, W, i) = \min_{P, L, \Delta_1, \ldots, \Delta_i} \{ N(R, W, L, P) + \tau + \max_{j=i,\ldots,2} \left\{ \frac{\Sigma_i}{2} \ , \ \Sigma_i - \mathfrak{P}_i \ , \ 2\Sigma_j - \mathfrak{P}_j - \mathfrak{P}_{j-1} \ , \ 2\Sigma_1 - \mathfrak{P}_1 - L \right\} \}$$

with $0 \le L \le 1 - R$, $\max\{0, R + L + W - 1\} \le P \le \min\{R + L, W\}$, $0 \le \Delta_i < \frac{R+L-P_i}{2}$, $0 \le \mathfrak{P}_i \le \mathfrak{P}_{i-1} \le \ldots \mathfrak{P}_1 \le L$ and $\tau > 0$ arbitrarily small. It is thus very natural to ask:

Is the COLUMN-MATCH algorithm optimal - or - is further improvement possible by adding additional recursion layers?

Unfortunately, we were not able to answer this question theoretically. The question of determining the optimal recursion depth is also unsolved for all other applications of the representation technique, e.g. the generic subset-sum algorithm of [BCJ11]. We believe that an entirely different modelling and analysis is needed to answer this question. However, numerical experiments (both for the decoding and the subset-sum scenario) indicate that a depth of two already yields the best results.

Moreover, motivated by the lack of fast space-efficient algorithms as discussed in Section 6.4.3, it might be interesting to

Design space-efficient ISD algorithm's (different from Plain ISD).

For example, it is possible to apply generic techniques for space-efficient Meet-in-the-Middle attacks as presented in [vOW99] to Stern's algorithm: Finding weight-p solutions $\mathbf{e}$ to $\mathbf{Qe} = \mathbf{s}$ can be phrased as finding collisions for two mappings $f_b : D \to R$ with domain $D \coloneqq W_{\frac{k}{2}, \frac{p}{2}}$ and range $R = \mathbb{F}_2^\ell$ defined by

$$f_b(\mathbf{e}) \coloneqq \mathbf{Q}_b \mathbf{e} + b \cdot \mathbf{s}$$

where $b \in \{0, 1\}$ and $\mathbf{Q} = (\mathbf{Q}_0 || \mathbf{Q}_1)$. By construction, a collision $f_0(\mathbf{e}_0) = f_1(\mathbf{e}_1)$ yields the solution $(\mathbf{e}_0 || \mathbf{e}_1)$. Now, our goal is to define another function $f : S \to S$ that behaves like a random function and has a collision that is related to the above collision $(\mathbf{e}_0, \mathbf{e}_1)$. For such f, the techniques of [vOW99] can be used to find collisions in time $\approx |S|^{3/2}$ using only constant memory (this also works when a particular efficiently recognisable "golden collision" is searched). A (heuristic) way to define f is to make use of a constant-weight embedding ϕ. A constant-weight embedding ϕ is simply a function that maps ℓ-bit strings to words of length $\frac{k}{2}$ and weight $\frac{p}{2}$. Using ϕ (see [FS96] for a simple

implementation) together with some function $g : R \to \{0,1\}$, e.g. $g(\mathbf{x}) = \sum x_i \mod 2$, we obtain a function $f : \{0,1\} \times D \to \{0,1\} \times D$ via

$$f(b, \mathbf{e}) := (g(f_b(\mathbf{e})), \phi(f_b(\mathbf{e}))) \quad .$$

Since $\mathbf{Q}$ is a random binary matrix, applying f_b yields uniform vectors of length ℓ, i.e. g also yields a uniform Bit b, and it is reasonable to (heuristically) assume that f behaves like a random function. Thus, since $|S| \approx \binom{k/2}{p/2}$, we obtain a (heuristic) variant of Stern's algorithm with slightly increased time complexity and *constant* space complexity (to be more precise, the running time is increased by a $\sqrt{\binom{k/2}{p/2}}$-factor assuming $\ell = \log \binom{k/2}{p/2}$). However, the sketched algorithm is heuristic and does not improve over Prange's algorithm with respect to the time-space complexity product. We leave it as an open question to find more efficient, rigorous algorithms (one possibility is to apply the recently proposed *dissection technique*, see [DDKS12]).

7 Improved Information Set Decoding over $\mathbb{F}_q$

"To be is to do."

Immanuel Kant

Introduction

From a practical point of view, the central goal in code-based cryptography is to find more efficient constructions with a particular focus on reducing the key sizes. For example, the public description of the McEliece T-OWF f is a $(k \times n)$-dimensional "random" generator matrix of the underlying Goppa code. When using f in order to obtain a CCA2-secure public key encryption scheme, by applying the Kobara-Imai transform [KI01] for example, it is possible to use a public generator matrix in standard form. This allows one to fully describe f by the redundancy part of size $(n-k)k$ only. In the binary case, the currently proposed parameters for 128-Bit security are $n = 2960$ and $k = 2288$ yielding a public key of size ≈ 188 KB.

Besides using particularly structured codes, e.g. the recent proposal of Misozcki et al. based on *moderate-density parity-check codes* [MTSB12], some authors proposed to use larger base fields $\mathbb{F}_q$ in order to improve the efficiency and in particular to decrease the key sizes. Put simply, some specific q-ary codes offer better decoding capabilities which allow to add errors of large weight in the McEliece T-OWF f for codes of relatively small length. Thus, one obtains smaller descriptions of f while (hopefully) preserving the hardness of inverting f.

For example, the recent proposals [BLP10, BLP11b] by Bernstein et al. introduced so-called "wild Goppa codes", which are essentially subfield codes over small $\mathbb{F}_q$, together with improved *list-decoding* algorithms. The notion of list decoding was introduced by Elias [Eli57] and considers decoding algorithms that work beyond the error-correction capability of the code by outputting a polynomial size list of candidate codewords. Interestingly, those codes offer an increased error-correction capability by a factor of $\approx q/(q-1)$. Thus, even for relatively small q, one can decode a rather large number of errors for codes of much smaller length. The seemingly best choice in [BLP10] is

$q = 31$, $n = 851$ and $k = 611$ yielding a significantly reduced public key size of ≈ 89 KB. An important open question is to properly study the security of these alternative constructions.

As in the binary case, ISD algorithms offer the best complexity for the q-ary syndrome decoding problem. Consequently, understanding the complexity of ISD algorithms over $\mathbb{F}_q$, and in particular estimating the effect of using *representations*, is important to evaluate the security of the respective cryptographic schemes. The following two questions are of particular interest:

1. How can the known algorithms be generalised and what complexity coefficients do we obtain?

2. How do the algorithms perform for growing q, i.e. how do the complexity coefficients $F(R, W, q)$ behave as a function of q? Do they converge to (possibly distinct) $F(R, W)$ and, if so, do they converge at different speeds?

General remarks. Let us first make some general observations towards the impact of larger base fields to ISD algorithms. Recall that the running time of all ISD algorithms is determined by

1. the (expected) number of iterations and

2. the workload of a single iteration.

We have seen that the first only depends on the notion of a "good" permutation which in turn is determined by the particular weight distribution that will be recovered within a single iteration. Note that this particular weight distribution only concerns the *error positions* of the hidden error vector and completely ignores the respective *error symbols*. Consequently, the number of iterations is *independent* of the size of the base field. However, the workload per iteration is heavily influenced by the size of the base field: When enumerating a set of candidate error vectors of a certain length n and weight ω, we can not only choose ω out of n many *error positions* but we also need to try all possible $q-1$ different *error symbols* per error position. Compared to the binary case, where the search space has size $\binom{n}{\omega}$, we obtain an expanded search space of size $\binom{n}{\omega}(q-1)^\omega$.

Impact on Minimum Distance Decoding. As explained in Chapter 3, it is sufficient to solve the CSD problem for errors of weight $W = \mathrm{D}_{\mathsf{GV}}(R, q) + \mathrm{o}(1)$ in order to solve the MDD problem (in the asymptotic setting). Recall that the relative GV distance is defined as $\mathrm{D}_{\mathsf{GV}}(R, q) = \mathrm{H}_q^{-1}(1 - R)$ which is monotone increasing in q. In particular, it holds

$$\lim_{q \to \infty} \mathrm{H}_q^{-1}(1 - R) = 1 - R \ , \tag{7.1}$$

see [CG90, Theorem 7]. In summary, for larger values of q we need to solve the CSD problem for larger error weights in order to solve the MDD problem. Put simply, the complexity of MDD increases with q.

Plain ISD over $\mathbb{F}_q$

An initial study of ISD over $\mathbb{F}_q$ was given by Coffey and Goodman in [CG90] who provided the first rigorous asymptotic analysis for Prange's plain ISD. Recall that a single iteration in Plain ISD comes (asymptotically) for free which in turn means that the running time of Plain ISD is essentially *independent* of the base field by our initial observation. Thus, we obtain exactly the same complexity coefficient as defined in Eq.(4.6). When using Plain ISD to solve the q-ary MDD problem, we need to find errors of weight $W = \mathrm{D}_{\mathsf{GV}}(R, q)$ and obtain the complexity coefficient

$$F(R,q)_{\mathsf{Pra}} = \left[\mathrm{H}\big(\mathrm{H}_q^{-1}(1-R)\big) - (1-R)\,\mathrm{H}\left(\frac{\mathrm{H}_q^{-1}(1-R)}{1-R}\right)\right]\log_q 2 \ ,$$

see Figure 7.1 for an illustration for varying q. Using Eq.(7.1) it follows $F(R,q)_{\mathsf{Pra}} \to \mathrm{H}(1-R)\log_q 2 \to 0$ for large q and the complexity coefficient tends towards zero with increasing q. Therefore the complexity becomes subexponential *in the symbol size* q for large base fields.

Remark 7.0.1. Note that the above argument *first* uses $n \to \infty$. The resulting complexity coefficient is then viewed as a function of q and further analysed for $q \to \infty$. This implicitly means that n must grow much faster than q. We point out that some interesting cryptographic problems, e.g. the *LWE-problem*, are connected to decoding problems in random q-ary codes of length n with $q = \mathrm{poly}(n)$ or even $q = 2^{\mathcal{O}(n)}$. These scenarios are not covered by the analysis presented in this chapter.

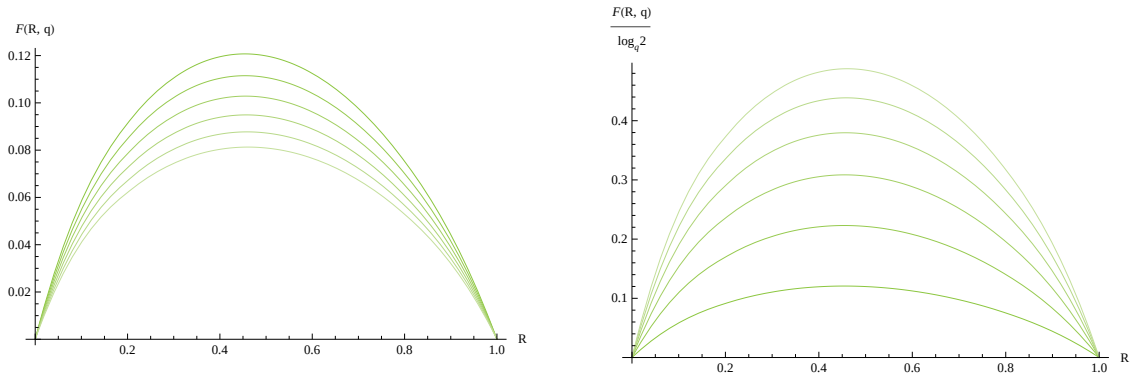


Figure 7.1: Complexity coefficient (left) and binary work factor (right) for different $q = 2^i$, $i = 1, \ldots, 6$ in decreasing (left) and increasing order (right), respectively.

An immediate consequence of $F(R,q)_{\mathsf{Pra}} \to 0$ is that the difference between *any* of the improved algorithms and Plain ISD becomes arbitrarily small with growing q: Since any algorithm A contains Plain ISD as a special case, we have $F(R,q)_A \leq F(R,q)_{\mathsf{Pra}}$ and thus $F(R,q)_A \to 0$ which implies $F_{\mathsf{Pra}} - F_A \to 0$. However, the convergence of Plain ISD towards 0 is caused by the (somehow arbitrary) choice of the base q in the logarithm that

occurs in the definition of $F(R, W, q)$. Note that the *binary work factor* $\frac{F(R,q)_{\mathsf{Pra}}}{\log_q 2}$ tends to $\mathrm{H}(1-R) = \mathrm{H}(R)$ with $q \to \infty$, see Figure 7.1. Furthermore, there might be different algorithms A and B with $F_A - F_B \to 0$ for large q but *distinct* binary work factors (for example consider $F(R,q)_A = \mathrm{H}(R)\log_q 2$ and $F(R,q)_B = \frac{\mathrm{H}(R)}{2}\log_q 2$ with binary work factor $\mathrm{H}(R) \neq \frac{\mathrm{H}(R)}{2}$, respectively). This shows that it is questionable whether the complexity coefficient is the appropriate metric to compare ISD algorithms for $q > 2$. However, we will shortly see that the improved algorithms "converge" to Plain ISD even with respect to their binary work factors.

Related Work. In contrast to the binary case, the study of ISD algorithms over $\mathbb{F}_q$ has not attracted much attention within the last years: The most recent work is due to Peters who presented a straightforward modification of Stern's algorithm in [Pet10]. Shortly after, Niebuhr et al. in [NCBB10] showed how to gain a $\sqrt{q-1}$ factor by exploiting the field structure. Essentially, the improvement of [NCBB10] is based on the following simple observation: Multiplying the given syndrome $\mathbf{s}$ with $\alpha \in \mathbb{F}_q \setminus \{0,1\}$ gives rise to $q-2$ additional syndromes, each of them revealing a related error vector $\alpha\mathbf{e}$. In order to find the original $\mathbf{e}$ it is thus sufficient to decode at least one out of the $q-1$ many different syndromes. Consequently, the authors of [NCBB10] designed a specific algorithm exploiting so-called "normalised" vectors (as introduced by Minder and Sinclair in [MS12]). However, for $q = \mathrm{poly}(n)$, a $\sqrt{q-1}$ improvement does not change the complexity coefficient at all and we omit a deeper description.

Roadmap

Except for Plain ISD, no asymptotic analysis of ISD algorithms over $\mathbb{F}_q$ exist so far. Thus, we initiate our study by restating Stern's algorithm over $\mathbb{F}_q$ and by providing its complexity coefficient in Section 7.1 (in a rather informal way). We then show how to generalise our improved algorithm to $\mathbb{F}_q$ in Section 7.2 and obtain an exponential improvement. However, for both algorithms, the improvement over Plain ISD becomes vanishingly small for large enough q.

7.1 Stern's Algorithm

As in the preceding chapters, we focus on finding weight-p solutions $\mathbf{e} \in \mathbb{F}_q^k$ to the equation $\mathbf{Qe} = \mathbf{s} \in \mathbb{F}_q^\ell$. For simplicity we only describe how to modify a single iteration of Stern's algorithm (the remaining algorithm stays the same). Recall that we decompose $\mathbf{Q} = (\mathbf{Q}_1 || \mathbf{Q}_2)$ into two $\ell \times \frac{k}{2}$ matrices $\mathbf{Q}_i$ and analogously split $\mathbf{e} = (\mathbf{e}_1 || \mathbf{e}_2)$ where both $\mathbf{e}_i$ have Hamming weight $\frac{p}{2}$. Now, the efficiency of a straightforward generalisation of Stern's algorithm is clearly affected by the size of the base field: In every single iteration we enumerate *all* vectors $\mathbf{e}_i$ of length $\frac{k}{2}$ and weight $\frac{p}{2}$. More precisely, we need to compute the two base lists

$$\mathcal{L}_1 := \{(\mathbf{Q}_1\mathbf{e}_1, \mathbf{e}_1) \ : \ \mathbf{e}_1 \in W^q_{\frac{k}{2},\frac{p}{2}}\} \ , \ \mathcal{L}_2 := \{(\mathbf{s} - \mathbf{Q}_2\mathbf{e}_2, \mathbf{e}_2) \ : \ \mathbf{e}_2 \in W^q_{\frac{k}{2},\frac{p}{2}}\}$$

where $W^q_{n,\omega}$ is the set of all $\mathbf{e} \in \mathbb{F}_q^n$ with weight ω. Obviously, both lists contain

$$\binom{\frac{k}{2}}{\frac{p}{2}}(q-1)^{\frac{p}{2}} \tag{7.2}$$

elements. These list are finally scanned for matching elements (with respect to the first entry). As usual, the expected number of matchings is given by $\binom{k/2}{p/2}^2 (q-1)^p q^{-\ell}$.

As already mentioned, the notion of a "good" permutation only depends on the error positions and is thus not affected by the size of the base field. Consequently, the probability of guessing a good permutation and therefore the expected number of iterations is identical to the binary case, see Eq.(4.17). We refrain from giving a more detailed description and analysis of the complete algorithm and merely state the main result. A formal proof could be immediately obtained by copying the proof of Theorem 4.3.1 and by using Lemma 2.3.5 to approximate Eq.(7.2) with

$$\begin{aligned}\Lambda(R,P,q) := \lim_{n\to\infty} \frac{1}{n} \log_q \mathrm{vol}_q(\lfloor \frac{R}{2} n \rfloor, \lfloor \frac{P}{2} n \rfloor) &= \frac{R}{2} \mathrm{H}_q\left(\frac{P}{R}\right) \\ &= \frac{R}{2} \mathrm{H}\left(\frac{P}{R}\right) \log_q 2 + \frac{P}{2} \log_q(q-1)\end{aligned} \tag{7.3}$$

and the number of matchings with $2\Lambda(R,P,q) - L$.

Theorem 7.1.1 (Stern's algorithm over $\mathbb{F}_q$)**.** *Let* $0 < R < 1$ *and* $0 < W \leq \mathrm{D}_{\mathsf{GV}}(R, q)$*. For almost all* q*-ary* $[n, \lfloor Rn \rfloor]$*-codes it holds: Stern's algorithm successfully terminates in time* $q^{F_{\mathsf{Ste}}(R,W,q)n+o(n)}$ *for every error vector* $\mathbf{e} \in W^q_{n,\lfloor Wn \rfloor}$ *with overwhelming probability where*

$$\begin{aligned} F_{\mathsf{Ste}}(R, W, q) &= \min_{P,L} \{N(R, W, P, L) \\ &\quad + \max\{\Lambda(R, P, q), 2\Lambda(R, P, q) - L\}\} \quad , \\ N(R, W, P, L) &= \left[\mathrm{H}(W) - R\,\mathrm{H}\left(\frac{P}{R}\right) - (1 - R - L)\,\mathrm{H}\left(\frac{W - P}{1 - R - L}\right)\right] \log_q 2 \end{aligned}$$

and Λ *as defined in Eq.(7.3) with* $0 \leq L \leq 1 - R$ *and* $\max\{0, R + L + W - 1\} \leq P \leq \min\{R, W\}$*.*

Optimised interpolated curves for $F_{\mathsf{Ste}}(R, W, q)$ for different q and fixed $W = \mathrm{D}_{\mathsf{GV}}(R, q)$ are presented in Figure 7.1.

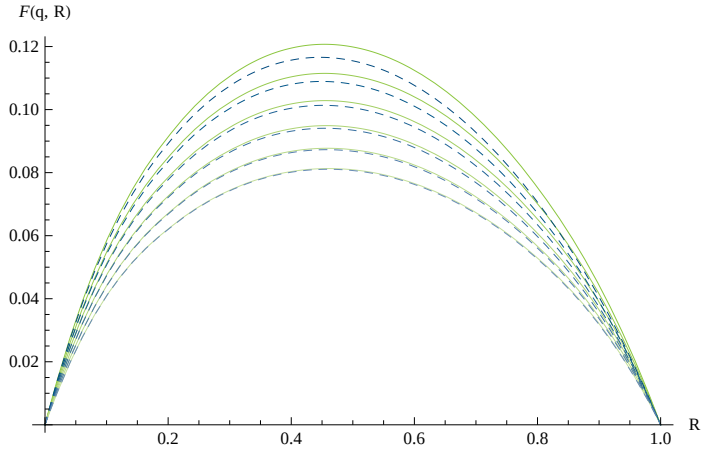


Figure 7.2: Comparison of $F(R, W, q)$ for Plain ISD (solid curves) and Stern's algorithm (dashed curves) for $q = 2, 4, \ldots, 64$ in decreasing order.

As already mentioned and illustrated by Figure 7.1, the larger q, the smaller the difference between Plain ISD and Stern's algorithm. Intuitively, this it not very surprising. For large q, the running time of the algorithm is dominated by the enumeration of all $(q-1)^{p/2}$ error symbols which forces smaller and smaller p. This observation is formalised (w.r.t. the binary work factor) in the next theorem where we prove that a sequence P_q^* of *optimal* P-parameters for Stern's algorithm fulfils $\frac{1}{2}P_q^* \log_2(q-1) \to 0$ for $q \to \infty$. Put differently, Stern's algorithm "converges" to Plain ISD. Let

$$\Delta(R, W, q) := \frac{F_{\mathsf{Pra}}(R, W, q) - F_{\mathsf{Ste}}(R, W, q)}{\log_q 2}$$

denote the (binary work factor) difference between Plain ISD and Stern's algorithm. In particular it holds $\Delta(R, W, q) \geq 0$ for all q (we can always set $P = L = 0$).

Theorem 7.1.2. *For every* $0 < R < 1$ *and* $0 < W < 1 - R$ *it holds* $\lim_q \Delta(R, W, q) = 0$.

Proof. For fixed W let Q^* be large enough such that $W \leq \mathrm{D}_{\mathsf{GV}}(R, q)$. Let $P_q^* := \{P^*(R, W, q)\}_{q \geq Q^*}$ denote a sequence of *optimal* P-parameters for Stern's algorithm. According to Eq.(7.3) it holds

$$\frac{F_{\mathsf{Ste}}(R, W, q)}{\log_q 2} \geq \frac{\Lambda(R, P_q^*, q)}{\log_q 2} \geq \frac{P_q^*}{2} \log_2(q-1) \ .$$

This implies that the sequence $\alpha_q := \frac{P_q^*}{2} \log_2(q-1)$ is bounded (it is always possible to set $P_q^* = 0$ for all q which transforms Stern's algorithm into Plain ISD, thus $\frac{F_{\mathsf{Ste}}}{\log_q 2} \leq \frac{F_{\mathsf{Pra}}}{\log_q 2} \leq 1$ always holds). Thus

$$\tau := \limsup_q \alpha_q < \infty$$

exists. Suppose that $\tau > 0$ and consider a subsequence $\alpha_{\hat{q}}$ with $\lim_{\hat{q}} \alpha_{\hat{q}} = \tau$. Note that $\tau < \infty$ implies $\lim_{\hat{q}} P_{\hat{q}}^* = 0$. Using

$$\frac{F_{\mathsf{Ste}}(R, W, \hat{q})}{\log_{\hat{q}} 2} \geq \mathrm{H}(W) - \frac{R}{2} \mathrm{H}\left(\frac{P_{\hat{q}}^*}{R}\right) - (1-R)\,\mathrm{H}\left(\frac{W - P_{\hat{q}}^*}{1-R}\right) + \alpha_{\hat{q}}$$

we obtain

$$-\Delta(R, W, \hat{q}) \geq \underbrace{(1-R)\left[\mathrm{H}\left(\frac{W}{1-R}\right) - \mathrm{H}\left(\frac{W - P_{\hat{q}}^*}{1-R}\right)\right] - \frac{R}{2}\mathrm{H}\left(\frac{P_{\hat{q}}^*}{R}\right)}_{=:\beta_{\hat{q}}} + \alpha_{\hat{q}} \ .$$

Note that $\lim_{\hat{q}} \beta_{\hat{q}} = 0$ since $\lim_{\hat{q}} P_{\hat{q}}^* = 0$. Thus for fixed $\varepsilon_\alpha > 0$, there is some Q_α with $\alpha_{\hat{q}} > \tau - \varepsilon_\alpha$ for all $\hat{q} \geq Q_\alpha$ and for $\varepsilon_\beta := \tau - \varepsilon_\alpha$ we will find some Q_β with $\beta_{\hat{q}} > -\varepsilon_\beta = \varepsilon_\alpha - \tau$ for all $\hat{q} \geq Q_\beta$. This yields $-\Delta(R, W, \hat{q}) > 0$ for all $\hat{q} \geq \max\{Q_\alpha, Q_\beta\}$ which is a contradiction (to $\Delta \geq 0$). Consequently we have $\tau = 0$ which implies $\lim_q \frac{P_q^*}{2} \log_2(q-1) = 0$. Thus $\lim_q \Lambda(R, P_q^*, q) = 0$ and $\lim_q L_q^* = 0$ follows (where L_q^* is a sequence of optimal L-parameters) and we eventually obtain $\lim_q \Delta(R, W, q) = 0$. □

The above theorem essentially states that Plain ISD is the best choice when q becomes large due to its comparably simple implementation and low memory consumption. Nevertheless, for rather small q (as frequently used in cryptography, e.g. $q = 31$ in [BLP10]) the more sophisticated algorithms might still be interesting.

7.2 Using Representations over $\mathbb{F}_q$

Note that within a single iteration of all improved ISD algorithms presented in Chapter 6 one starts with the enumeration of a certain number of base lists each containing *all* vectors of length $\frac{k+\ell}{2}$ and appropriate weight t (to avoid confusion we point out that t is a function depending on the parameters p and δ_i where the maximal value of i is determined by the number of recursion layers of the algorithm). Thus, when generalising these algorithms to larger base fields, the asymptotic size of the respective base lists is given by $\frac{R+L}{2}\mathrm{H}_q(\frac{T}{R+L})$. An argument similar to the proof of Theorem 7.1.2 shows that $T_q^* \to 0$ must hold for every possible sequence of optimal parameters T_q^* (which implies that sequences of optimal parameters P_q^* and $(\Delta_i^*)_q$ tend to zero with growing q). Stated simply, for large enough q, any improved algorithm degenerates to Plain ISD. Consequently, the question of finding the asymptotically *optimal* algorithm becomes less significant for growing q. Thus, we will only give a simplified description of the improved algorithm in the following sense:

- We only provide a *lower bound* on the number of representations in the asymptotic setting. The actual number of representations might differ by an exponential factor. However, we indicate how a tight estimate can be obtained numerically and discuss the resulting improvement on the algorithm's running time.

- We only present a generalisation of the simple BasicReps algorithm, i.e. we refrain from using representations recursively. This is justified by numerical optimisation which failed to provide improved results for the recursive variant for $q > 3$.

In order to generalise BasicReps, it is merely necessary to study the effect of larger base fields on the number of representations. Fortunately, larger q increase the number of representations by a remarkable amount. Recall that in the binary case, a vector $\mathbf{e}$ of length $k+\ell$ and weight p has $\binom{p}{p/2}\binom{k+\ell-p}{\delta}$ many representations of the form $\mathbf{e} = \mathbf{e}_1 + \mathbf{e}_2$ where both $\mathbf{e}_i$ have Hamming weight $\frac{p}{2}+\delta$. Also, recall that the first factor counts the possible ways of picking half of $\mathbf{e}$'s non-zero coordinates and the second factor counts the possible ways of choosing δ many overlapping coordinates in $\mathbf{e}_1$ and $\mathbf{e}_2$. Over $\mathbb{F}_q$, we obviously gain extra freedom in picking the overlapping coordinates. Every of the δ picked 0's in $\mathbf{e}$ can be written in $q-1$ ways. At first sight, this seems to be the only gain (since there is still only one way to write one out of the p error symbols x as either $x+0$ or $0+x$) and one might estimate the number of representations by

$$\binom{p}{p/2}\binom{k+\ell-p}{\delta}(q-1)^\delta \ .$$

However, some of the δ overlapping coordinates might also occur *within* the error positions of $\mathbf{e}$: In this case, the error symbol x can be written in $q-2$ different ways as $x = y + z$ where $y \notin \{0, x\}$. The overall effect is summarised in the next lemma.

Lemma 7.2.1. *Let* $\mathbf{e} \in \mathbb{F}_q^{k+\ell}$ *with* $\mathrm{wt}(\mathbf{e}) = p$. *For* $0 \le \delta \le k + \ell - p$ *denote* $\rho(k, \ell, p, \delta, q)$ *the number of pairs* $(\mathbf{e}_1, \mathbf{e}_2)$ *with* $\mathrm{wt}(\mathbf{e}_i) = \frac{p}{2} + \delta$ *and* $\mathbf{e} = \mathbf{e}_1 + \mathbf{e}_2$. *It holds*

$$\rho(k, \ell, p, \delta, q) = \sum_{i=0}^{\min\{\frac{p}{2}, \delta\}} \binom{p-2i}{\frac{p}{2}-i}\binom{p}{2i}(q-2)^{2i}\binom{k+\ell-p}{\delta-i}(q-1)^{\delta-i} \quad . \tag{7.4}$$

Proof. Let us assume wlog that the errors of $\mathbf{e}$ occur in the first p positions. Let p_1 denote the Hamming weight of $\mathbf{e}_1$ on these positions, i.e. $\frac{p}{2} \le p_1 \le \min\{p, \frac{p}{2} + \delta\}$. Set $i := p_1 - \frac{p}{2}$, i.e. $0 \le i \le \min\{\frac{p}{2}, \delta\}$. Since $\mathbf{e}_1$ has total Hamming weight $\frac{p}{2} + \delta$, it has Hamming weight $\delta - i$ on the last $k + \ell - p$ positions. Since $\mathbf{e} = \mathbf{e}_1 + \mathbf{e}_2$, $\mathbf{e}_1$ and $\mathbf{e}_2$ must overlap on these $\delta - i$ positions and there are exactly $\binom{k+\ell-p}{\delta-i}(q-1)^{\delta-i}$ ways of choosing these overlaps. Moreover, $\mathbf{e}_1$ and $\mathbf{e}_2$ must overlap on $2p_1 - p = 2i$ out of the first p error positions of $\mathbf{e}$, i.e. there are $2i$ error symbols $x \neq 0$ in $\mathbf{e}$ that can be written as $x = y + z$ with $y, z \neq 0$ (i.e. we can freely choose $y \notin \{0, x\}$). There are exactly $\binom{p}{2i}(q-2)^{2i}$ ways of choosing these overlaps. Finally, we can arbitrarily assign half of the remaining $p - 2i$ error symbols of $\mathbf{e}$ to $\mathbf{e}_1$ (which also determines the remaining non-zero symbols of $\mathbf{e}_2$). This explains the $\binom{p-2i}{\frac{p}{2}-i}$ factor in the above formula. □

Note that Lemma 7.2.1 generalises Lemma 5.1.5 for the binary case: The only non-vanishing summand is $\binom{p}{p/2}\binom{k+\ell-p}{\delta}$ for $i = 0$. Knowing the number of representations immediately enables us to generalise the BasicReps algorithm from Chapter 6 in a very straightforward way: We define $\rho := \rho(k, \ell, p, \delta, q)$ and we need to compute two intermediate lists

$$\mathcal{L}_1 := \{\mathbf{e}_1 \in W^q_{k+\ell, \frac{p}{2}+\delta} \ : \ (\mathbf{Q}\mathbf{e}_1)_{[r]} = \mathbf{t}\} \ ,$$
$$\mathcal{L}_2 := \{\mathbf{e}_2 \in W^q_{k+\ell, \frac{p}{2}+\delta} \ : \ (\mathbf{Q}\mathbf{e}_2)_{[r]} = \mathbf{s}_{[r]} - \mathbf{t}\}$$

where $\mathbf{t} \in_R \mathbb{F}_q^r$ is a random target vector with $r := \lfloor \log_q \rho \rfloor$. As usual, one computes the final candidate list $\mathcal{L}$ containing all solutions $\mathbf{e} \in W^q_{k+\ell,p}$ with $\mathbf{Q}\mathbf{e} = \mathbf{s}$ by invoking Merge-Join, i.e. $\mathcal{L} \leftarrow$ Merge-Join$(\mathcal{L}_1, \mathcal{L}_2, p, \mathbf{Q}, \mathbf{s})$. Obviously, Merge-Join also works over arbitrary base fields with exactly the same performance by simply using the respective lexicographic order over $\mathbb{F}_q$.

Clearly, the intermediate lists $\mathcal{L}_1$ and $\mathcal{L}_2$ can be computed analogously to the binary case: For $\mathcal{L}_1$, choose a random partition $[k + \ell] = P_1 \cup P_2$, $|P_1| = |P_2| = \frac{k+\ell}{2}$ and construct two support-disjoint base lists

$$\mathcal{B}_1(P_1) := \{\mathbf{y} \in W^q_{k+\ell, \frac{p}{4}+\frac{\delta}{2}} \text{ and } \mathrm{supp}(\mathbf{y}) \subset P_1\} \ ,$$
$$\mathcal{B}_2(P_2) := \{\mathbf{z} \in W^q_{k+\ell, \frac{p}{4}+\frac{\delta}{2}} \text{ and } \mathrm{supp}(\mathbf{z}) \subset P_2\}$$

which can them be merged into $\mathcal{L}_1 \leftarrow$ Merge-Join$(\mathcal{B}_1(P_1), \mathcal{B}_2(P_2), \frac{p}{2} + \delta, \mathbf{Q}_{[r]}, \mathbf{t})$. We omit the identical description for $\mathcal{L}_2$. Altogether, we obtain exactly the same algorithm as presented in Algorithm 6 with adapted choice of the parameter ρ and expanded lists as described above. To obtain the usual asymptotic statement about the running time,

it is necessary to generalise Theorem 6.2.2 and Lemma 6.2.3 to $\mathbb{F}_q$ which can easily be done by substituting 2 with q. Consequently, BasicReps is τ-solution preserving over $\mathbb{F}_q$ which can be proven analogously to Lemma 6.2.4.

Theorem 7.2.2. *For a fixed matrix $\mathbf{Q} \in \mathbb{F}_q^{m\times n}$, a target vector $\mathbf{t} \in \mathbb{F}_q^m$ and an arbitrary set $\mathcal{B} \subset \mathbb{F}_q^n$, $\varnothing \neq \mathcal{B}$ we define*

$$P_{\mathbf{Q}}(\mathcal{B}, \mathbf{t}) := \frac{1}{|\mathcal{B}|} \left|\{\mathbf{x} \in \mathcal{B} \ : \ \mathbf{Q}\mathbf{x} = \mathbf{t}\}\right| \ .$$

It holds

$$\sum_{\mathbf{Q}\in\mathbb{F}_q^{m\times n}} \sum_{\mathbf{t}\in\mathbb{F}_q^m} (P_{\mathbf{Q}}(\mathcal{B}, \mathbf{t}) - \frac{1}{q^m})^2 = q^{mn} \frac{q^m - 1}{q^m |\mathcal{B}|} \ . \tag{7.5}$$

Lemma 7.2.3. *Let $m \leq n$ and $|\mathcal{B}| \geq q^m$. Let $\Gamma \in \mathbb{N}$. For all but a $\frac{1}{\Gamma-1}$ fraction of all $\mathbf{Q} \in \mathbb{F}_q^{m\times n}$ it holds*

$$\mathbf{Pr}\left[P_{\mathbf{Q}}(\mathcal{B}, \mathbf{t}) = 0\right] \leq 1 - \frac{1}{\Gamma}$$

where the probability is over the random choice of $\mathbf{t} \in \mathbb{F}_q^m$.

Lemma 7.2.4. *For almost all input matrices $\mathbf{Q}$,* BasicReps *is τ-solution-preserving for arbitrary $\tau > 0$.*

Runtime Analysis and Main Theorem

According to Chapter 6 we define

$$\Sigma(R, L, P, \Delta, q) := (R + L)\, \mathrm{H}_q\left(\frac{\frac{P}{2} + \Delta}{R + L}\right) \tag{7.6}$$

and

$$\mathfrak{P}(R, L, P, \Delta, q) := P \log_q 2 + (R + L - P)\, \mathrm{H}_q\left(\frac{\Delta}{R + L - P}\right) \ . \tag{7.7}$$

Note that in Eq.(7.7) we implicitly lower bound $\rho \geq \binom{p}{p/2}\binom{k+\ell-p}{\delta}(q-1)^\delta$ in the asymptotic setting. Using Vandermonde's inequality, see Lemma 2.3.2, together with $\log_q 2 < \mathrm{H}_q(\frac{1}{2})$ for $q > 2$ yields

$$\begin{aligned} \mathfrak{P}(R, L, P, \Delta, q) &= P \log_q 2 + (R + L - P)\, \mathrm{H}_q\left(\frac{\Delta}{R + L - P}\right) \\ &< P\, \mathrm{H}_q\left(\frac{\frac{P}{2}}{P}\right) + (R + L - P)\, \mathrm{H}_q\left(\frac{\Delta}{R + L - P}\right) \leq (R + L)\, \mathrm{H}_q\left(\frac{\frac{P}{2} + \Delta}{R + L}\right) = \Sigma(R, L, P, \Delta, q) \end{aligned} \tag{7.8}$$

for $q > 2$. This is sufficient to obtain the following estimate for the running time of BasicReps which is similar to Lemma 6.2.6.

Lemma 7.2.5. *Let $0 < R < 1$. For $0 \le L \le 1 - R$, $0 \le P \le R + L$, $0 \le \Delta < R + L - P$ and $\mathfrak{P}(L, P, \Delta, q) \le L$ with Σ and $\mathfrak{P}$ as defined in Eq.(7.6) and (7.7) the following holds: For almost all $\mathbf{Q}$,* BasicReps *runs in time $q^{C(R,L,P,\Delta,q)n+o(n)}$ where*

$$C(R, L, P, \Delta, q) = \max\left\{\frac{\Sigma}{2}, \Sigma - \mathfrak{P}, 2\Sigma - \mathfrak{P} - L\right\} \quad .$$

Proof. We only indicate the differences to the proof of Lemma 6.2.6. We need to bound the size of the lists $\mathcal{B}_i$ and $\mathcal{L}_i$ and the number of matchings between $\mathcal{L}_1$ and $\mathcal{L}_2$, denoted by M. It holds

$$\lim_{n\to\infty} \frac{1}{n} \log_q |\mathcal{B}_i| = \frac{\Sigma}{2}$$

and

$$\lim_{n\to\infty} \frac{1}{n} \log_q |\mathcal{L}_i| = \Sigma - \mathfrak{P}$$

for almost all $\mathbf{Q}$ (this follows from Chebychev's inequality since $\mathbb{E}[|\mathcal{L}_i|] = q^{(\Sigma-\mathfrak{P})n+o(n)}$, which is exponentially increasing in n due to Eq.(7.8), and $\mathrm{Var}[|\mathcal{L}_i|] \le \mathbb{E}[|\mathcal{L}_i|]$ by 2.4.6). By construction of the lists $\mathcal{L}_i$, every pair yields a matching on all ℓ coordinates with probability $q^{r-\ell}$ and we obtain

$$\mathbb{E}[M] = \frac{|\mathcal{L}_1 \times \mathcal{L}_2|}{q^{\ell-r}} = q^{(2\Sigma-\mathfrak{P}-L)n+o(n)}$$

since $\mathfrak{P} \le L$. Using standard arguments it follows $\lim_{n\to\infty} \frac{1}{n} \log_q M = \max\{2\Sigma - \mathfrak{P} - L, \epsilon\}$ for $\epsilon > 0$ arbitrarily small. □

As usual, the main result follows by aborting BasicReps in every bad iteration and by combining Lemma 5.1.9 with Lemma 7.2.4 and Lemma 7.2.5.

Main Theorem 7.2.1 (BasicReps over $\mathbb{F}_q$)**.** *Let $0 < R < 1$ and $0 < W \le \mathrm{D}_{\mathsf{GV}}(R, q)$. For almost all q-ary $[n, \lfloor Rn \rfloor]$-codes it holds: Generalised ISD instantiated with* BasicReps *successfully terminates in time $q^{F_{\mathsf{BReps}}(R,W,q)n+o(n)}$ for every error vector $\mathbf{e} \in W^q_{n,\lfloor Wn \rfloor}$ with overwhelming probability where*

$$F_{\mathsf{BReps}}(R, W, q) = \min_{P,L,\Delta}\{N(R, W, L, P, q) + \tau + \max\left\{\frac{\Sigma}{2}, \Sigma - \mathfrak{P}, 2\Sigma - \mathfrak{P} - L\right\}\}$$

with $N(R, W, L, P, q)$ is as defined in Theorem 7.2.1, $0 \le L \le 1 - R$, $\max\{0, R + L + W - 1\} \le P \le \min\{R + L, W\}$, $0 \le \Delta \le R + L - P$, $0 \le \mathfrak{P} \le L$ and $\tau > 0$ arbitrarily small.

We conclude by providing the interpolated complexity curves (see Figure 7.3) and worst-case running times (see Table 7.1) for Plain ISD, Stern's ISD and two variants using representations for different $q \in \{2, 4, 8, 16, 32, 64\}$: The first variant corresponds to the coefficient F_{BReps} as given in the above theorem and is denoted BReps. In contrast to the first variant, the second, extended variant (denoted xBReps) makes use of a *tight* asymptotic estimate for the number of representations. According to Eq.(7.4) we numerically computed the maximal number of representations for fixed P, L, Δ as the local (real) maximum of the function

$$f(x) := \log_q 2\left[(P-2x) + P\,\mathrm{H}\left(\frac{2x}{P}\right) + 2x\log(q-2)\right] + (R+L-P)\,\mathrm{H}_q\left(\frac{\Delta - x}{R+L-P}\right) \ .$$

	Plain ISD		Stern		BReps		xBReps	
q	$\max_R F(R)$	$\arg\max F(R)$	$\max_R F(R)$	$\arg\max F(R)$	$\max_R F(R)$	$\arg\max F(R)$	$\max_R F(R)$	$\arg\max F(R)$
2	.1208	.4539	.1166	.4468	.1053	.4282		
4	.1115	.4543	.1090	.4494	.1033	.4346	.1014	.4355
8	.1029	.4554	.1014	.4521	.0989	.4442	.0969	.4417
16	.0950	.4569	.0941	.4549	.0929	.4508	.0918	.4468
32	.0878	.4587	.0873	.4575	.0867	.4555	.0863	.4525
64	.0813	.4607	.0811	.4600	.0808	.4589	.0806	.4576

Table 7.1: Worst-case complexities $F(R)$ and corresponding rates R for different q and all algorithms. For $q = 128$ and $q = 256$ the difference between Plain ISD and xBReps becomes 0.003 and 0.001, respectively.

7.3 Conclusion and Open Problems

The main lesson of this chapter is that

Sophisticated algorithms become less powerful for large q.

Moreover, the numerical data indicates that Plain ISD already performs surprisingly good for rather small $q \approx 100$. To confirm this claim from a practical point of view, a finer analysis, as initiated for the binary case in Chapter 8, is needed. As we have seen, enumerating the *error symbols* is the main reason for the inefficiency of the improved algorithms. In some cryptographically motivated settings, the error symbols are likely to be contained in a rather small subset $E \subset \mathbb{F}_q$. Clearly, this can immediately be used in order to decrease the size of the intermediate lists in all algorithms.

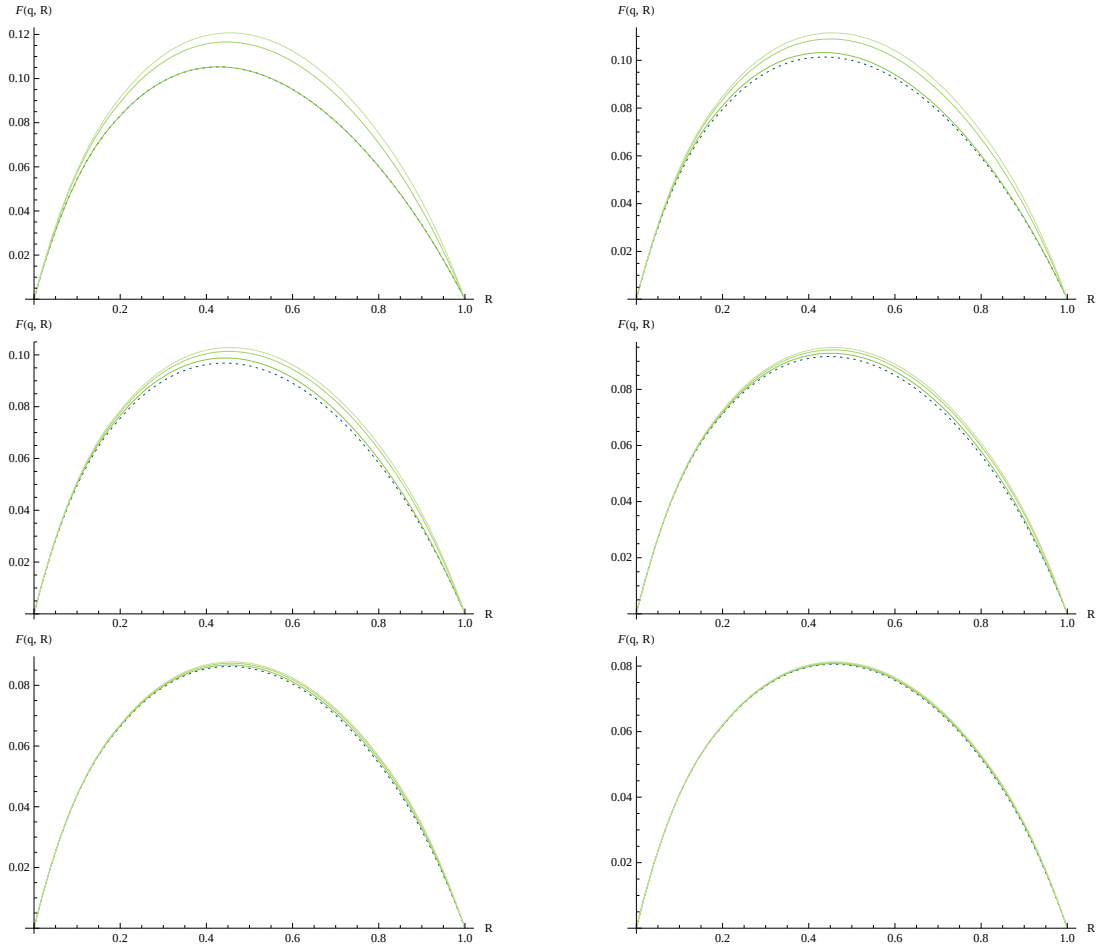


Figure 7.3: Complexity coefficients between $q = 2$ (top left corner) and $q = 64$ (bottom right corner) from left to right. Complexity curves are given for Plain ISD, Stern's algorithm, BReps and xBReps. Note that for $q = 2$ Eq.(7.4) is tight and there is no difference between BReps and xBReps

8 Applications

So far, we exclusively carried out a very rough asymptotic analysis of all ISD algorithms. As a consequence, it remains absolutely unclear whether our new algorithms have any practical impact. Compared to all known ISD algorithms, our improved algorithm requires the implementation of a rather complex computation tree where different expensive operations occur when merging two lists of one layer into a new list of the next layer (like sorting lists, eliminating weight-inconsistent solutions, updating labels of list elements, and so on and so forth). In this chapter we will resolve this unsatisfactory situation by answering the following question:

Are there codes of practical interest for which our algorithm performs better than Ball-collision decoding?

More precisely, we aim to give a (acceptably simple) concrete implementation of our algorithm (which we call BJMM for short) and compare its *binary work factor*, see Definition 8.1.1, with the one of BCD. For this purpose, we proceed as follows:

- We give a thorough analysis of BJMM and BCD, see Section 8.1.
- We present optimal parameters and the resulting work factors for some interesting codes arising from cryptographic applications, see Section 8.2.

From a practical point of view, the main result of this chapter is a revised security analysis of concrete McEliece parameter sets which shows that

high security levels are notably affected by our work.

We present a very simple implementation of BJMM (using only one layer of representations) that reduces the running time for the best generic attack by a factor 4, 2^6 and 2^{20} for the 80 Bit, 128 Bit and 256 Bit security level, respectively (Section 8.2.1). Another interesting application of our algorithm is a refined security analysis for the so-called *LPN problem* (see Definition 8.2.1). The LPN assumption is widely used in many recent cryptographic constructions. Although there is a well-known link between the LPN problem and coding theory, no cryptanalyst evaluated the concrete hardness of the LPN problem when attacked with ISD thus far. Consequently, we present the first concrete security analysis for different LPN parameters of practical interest. In Section 8.2.2 we show that

almost all low-noise instances of the LPN problem can be efficiently attacked with ISD algorithms.

8.1 Refined Complexity Analysis for $\mathbb{F}_2$

For the sake of optimal comparability with BCD, our analysis follows the analysis of Bernstein et al. in [BLP11a]. This analysis is based on the following assumptions:

- Memory allocation and computing Hamming weights comes for free (this might be worth discussing).
- A collision search between two lists $\mathcal{L}_1$ and $\mathcal{L}_2$ can be implemented in time linear in $\max|\mathcal{L}_i|$ via hash tables (without sorting).
- We estimate the complexity as the expected number of *field additions* (ignoring the costs of field multiplications is only reasonable for $\mathbb{F}_2$ where computing a matrix-vector product $\mathbf{He}$ can be seen as first *selecting* and then *adding* a particular subset of columns of $\mathbf{H}$).

Under these assumptions, we make the following definition.

Definition 8.1.1. Let $\mathcal{A}$ be an algorithm for the binary CSD problem with parameter space $P_{\mathcal{A}}(n,k,\omega)$. By $T_{\mathcal{A}}(n,k,\omega,\pi)$ we denote the expected number of additions required by $\mathcal{A}$ for a random CSD instance where $\pi \in P_{\mathcal{A}}(n,k,\omega)$ is some valid parameter set for $\mathcal{A}$. The *binary work factor* of $\mathcal{A}$ is given by

$$\mathrm{WF}_{\mathcal{A}}(n,k,\omega) := \min_{\pi \in P_{\mathcal{A}}(n,k,\omega)} T_{\mathcal{A}}(n,k,\omega,\pi) \ .$$

The following analysis will make use of some folklore complexity estimates and optimisation techniques for the three basic steps of general ISD algorithms.

Standard form computation. In every iteration, the (quasi-)standard form of the randomly column-permuted input matrix $\mathbf{H}$ has to be computed. In the last two decades, many researches invented variants of ISD algorithms with the only goal to reduce the workload of this task. The most famous proposal is due to Canteaut and Chabaud [CC98]. Instead of choosing a completely random permutation in every single iteration, one merely swaps a random column of the identity matrix part of $\mathbf{H}$ with a random column of the non-identity part of $\mathbf{H}$. Consequently, one merely needs to transform a single column into an appropriate unit vector (if this step fails one simply swaps the column again). Unfortunately, this modification completely destroys the independence between the different iterations and one needs to carry out a technical analysis based on *Markov chains* in order to evaluate the actual number of iterations. For small code parameters, this optimisation is worth considering. In particular, it has been further optimised by Bernstein et al. in the first practical attack on McEliece in [BLP08]) by introducing another optimisation parameter $c > 0$ representing a flexible number of columns to be swapped in each round. However, the most recent parameters sets used in cryptography are so large that the standard form computation consumes negligible time compared to

all other steps. Thus, we only consider a naïve implementation of Gaussian elimination whose running time (in terms of the expected number of binary additions) can be estimated as

$$\frac{(n-k)^2(n+k)}{2} \tag{8.1}$$

when applied to a $(n-k) \times n$-dimensional binary matrix.

Building the base lists. Recall that every iteration of both BCD and BJMM starts with the computation of particular base lists. Let us assume that such a base list contains all binary vectors $\mathbf{e}$ of length n and weight p together with the respective label $\mathbf{Qe}+\mathbf{t}$ of length ℓ where $\mathbf{t}$ is some appropriately chosen randomised target vector (alternatively, when using hash tables, every pair $(\mathbf{e}, \mathbf{Qe}+\mathbf{t})$ is stored at the position defined by its label $\mathbf{Qe}+\mathbf{t}$). A naive implementation of this step requires to compute $\binom{n}{p}$ matrix-vector products $\mathbf{Qe}+\mathbf{t}$, i.e. every single label requires the addition of p columns of $\mathbf{Q}$ (of length ℓ) plus one extra addition for $\mathbf{t}$. This results in a total number of $\binom{n}{p}p\ell$ additions.

In [BLP08] the authors propose to use the following simple strategy which reuses intermediate sums and allows to reduce the workload by almost a factor of p: First compute *all* $\binom{n}{2}$ sums of two columns of $\mathbf{Q}$ where each sum costs *one* addition in $\mathbb{F}_2^\ell$ and add the target vector $\mathbf{t}$. After this step, one has computed all labels $\mathbf{Qe}+\mathbf{t}$ with $\mathrm{wt}(\mathbf{e}) = 2$. Continue by computing all $\binom{n}{3}$ sums of columns of $\mathbf{Q}$ by adding one extra column to appropriate sums $\mathbf{Qe}+\mathbf{t}$ precomputed in the first step. Again, each single sum requires *one* addition in $\mathbb{F}_2^\ell$ and we end up with a lists of all labels $\mathbf{Qe}+\mathbf{t}$ with $\mathrm{wt}(\mathbf{e}) = 3$. Proceed similarly until all $\binom{n}{p}$ sums of p columns are obtained. The resulting number of additions is given by $\ell \cdot L(n,p)$ where

$$L(n,p) := \binom{n}{2} + \sum_{i=2}^{p} \binom{n}{i} \ . \tag{8.2}$$

The first extra summand in Eq.(8.2) is due to adding $\mathbf{t}$ in the first step. Altogether one gains a $\binom{n}{p} \cdot p \cdot L(n,p)^{-1}$ factor over the naive method. Note that $\binom{n}{p} \cdot L(n,p)^{-1}$ is very close to 1 for typically small values of p. For simplicity, we will assume that all sums of p out of n columns of length ℓ can be computed using exactly

$$\binom{n}{p}\ell \tag{8.3}$$

additions. We thus slightly underestimate the complexity of all attacks (by roughly the same factor) which results in a *conservative* security analysis of the proposed cryptographic constructions (see Section 8.2). For comparison, we also refer to the recently published non-asymptotic analysis of Hamdaoui and Sendrier [HS13] which offers slightly better results under additional (heuristic) assumptions.

Extending solutions. Every candidate solution $\mathbf{e}$ found in a single iteration has to be checked for correctness by computing $\mathrm{wt}(\mathbf{Qe}+\mathbf{s}) = \omega - p$ (assuming that all candidate solutions have weight p). Since candidate solutions match the syndrome $\mathbf{s}$ on ℓ

coordinates by construction, a naive implementation of the weight check would require the addition of p columns of $\mathbf{Q}$ (ignoring the first ℓ rows) plus one extra addition for $\mathbf{s}$. This results in a total number of $p \cdot (n-k-\ell)$ additions per candidate solution. Since a candidate solution can safely be discarded as soon as the weight becomes $> \omega - p$, one only has to compute the first $2(\omega - p + 1)$ coordinates of $\mathbf{Qe} + \mathbf{s}$ on average. Thus, using this "early abort" strategy the expected number of additions per candidate solution is given by

$$2p(\omega - p + 1) \ . \tag{8.4}$$

The main difference in the following analysis of BCD and BJMM stems from the fact that the BJMM algorithm requires one additional computation layer. We start with BCD following the lines of [BLP11a].

8.1.1 Refined Analysis for BCD

The probability of guessing a good permutation for BCD is given by Eq.(4.19) which yields the expected number of iterations

$$N_{\mathsf{BCD}}(n,k,\omega,\ell,p,q) = \frac{\binom{n}{\omega}}{\binom{k/2}{p/2}^2 \binom{\ell/2}{q/2}^2 \binom{n-k-\ell}{\omega-p-q}} \ . \tag{8.5}$$

Recall that for BCD, the base list $\mathcal{L}_1$ contains all pairs $(\mathbf{e}_{1,1}, \mathbf{e}_{2,1})$ where $\mathbf{e}_{1,1}$ is of length $\frac{k}{2}$ and weight $\frac{p}{2}$ and $\mathbf{e}_{2,1}$ is of length $\frac{\ell}{2}$ and weight $\frac{q}{2}$, respectively. Moreover, the corresponding labels are given by $\mathbf{Q}_1\mathbf{e}_{1,1} + \mathbf{J}_1\mathbf{e}_{2,1}$ where $\mathbf{Q}_1$ is the left submatrix (of dimension $\ell \times \frac{k}{2}$) of $\mathbf{Q}$ and $\mathbf{J}_1$ is the left submatrix (of dimension $\ell \times \frac{\ell}{2}$) of the identity matrix $\mathbf{I}_\ell$, see Section 4.3. The best way to compute $\mathcal{L}_1$ is to use intermediate sums for both $\mathbf{e}_{1,1}$ and $\mathbf{e}_{2,1}$: One first computes all labels $\mathbf{Q}_1\mathbf{e}_{1,1}$ using $\binom{k/2}{p/2}\ell$ additions and then adds all possible vectors $\mathbf{J}_1\mathbf{e}_{2,1}$ using $\binom{\ell/2}{q/2}$ extra additions per $\mathbf{e}_{1,1}$ (note that only a single bit has to be added for different $\mathbf{e}_{2,1}$ since all columns of $\mathbf{J}_1$ have Hamming weight 1). Altogether, $\mathcal{L}_1$ can be computed using

$$\binom{k/2}{p/2}\left[\ell + \min\{1,q\}\binom{\ell/2}{q/2}\right] \tag{8.6}$$

additions (for $q = 0$ this covers Stern's algorithm as well). A similar statement holds for $\mathcal{L}_2$.

Since both $\mathcal{L}_1$ and $\mathcal{L}_2$ contain $\binom{k/2}{p/2}\binom{\ell/2}{q/2}$ elements, one expects a total number of $\binom{k/2}{p/2}^2\binom{\ell/2}{q/2}^2 2^{-\ell}$ collisions. Using early abort, testing for correct solutions can be implemented using

$$2p(\omega - p - q + 1)\frac{\binom{k/2}{p/2}^2\binom{\ell/2}{q/2}^2}{2^\ell} \tag{8.7}$$

additions. Recall that the final weight check for BCD is given by $\mathrm{wt}(\mathbf{Q}\mathbf{e}_1 + (\mathbf{e}_2\|\mathbf{0}) + \mathbf{s}) = \omega - p - q$ where $\mathbf{e}_1 = (\mathbf{e}_{1,1}\|\mathbf{e}_{1,2})$, $\mathbf{e}_2 = (\mathbf{e}_{2,1}\|\mathbf{e}_{2,2})$ with $(\mathbf{e}_{1,i}, \mathbf{e}_{2,i}) \in \mathcal{L}_i$. Since $\mathbf{e}_2$ is of length ℓ, it does not affect $\mathbf{Q}\mathbf{e}_1 + (\mathbf{e}_2\|\mathbf{0}) + \mathbf{s}$ outside the first ℓ coordinates. This explains why the first factor in Eq.(8.7) is only p instead of $p+q$. Combing Eq.(8.6) and (8.7) with Eq.(8.1) allows to estimate the workload per iteration by

$$C_{\mathsf{BCD}}(n,k,\omega,\ell,p,q) = \frac{(n-k)^2(n+k)}{2} + 2\binom{k/2}{p/2}\left[\ell + \min\{1,q\}\binom{\ell/2}{q/2}\right] + 2p(\omega-p-q+1)\frac{\binom{k/2}{p/2}^2\binom{\ell/2}{q/2}^2}{2^\ell} \tag{8.8}$$

and we obtain the following result.

Theorem 8.1.2. *It holds*

$$\mathrm{WF}_{\mathsf{BCD}}(n,k,\omega) = \min_{\ell,p,q} N_{\mathsf{BCD}}(n,k,\omega,\ell,p,q)\cdot C_{\mathsf{BCD}}(n,k,\omega,\ell,p,q)$$

where $0 \le \ell \le n-k$, $0 \le q \le \min\{\ell,\omega\}$, $\max\{0, k+\ell+\omega-q-n\} \le p \le \min\{\omega-q, k\}$.

Remark 8.1.3. The space complexity of BCD is given by

$$S_{\mathsf{BCD}}(n,k,\omega,\ell,p,q) = n(n-k) + \frac{k+\ell\cdot\min\{1,q\}}{2}\binom{k/2}{p/2}\binom{\ell/2}{q/2}\ .$$

The first term accounts for the size of the parity check matrix. The second term represents the size of one of the base lists (as usual, storing one list and checking for collisions on-the-fly is sufficient). When using hash tables, the list entries are pairs $(\mathbf{e}_{1,1}, \mathbf{e}_{2,1})$ where $\mathbf{e}_{1,1}$ and $\mathbf{e}_{2,1}$ are of length $\frac{k}{2}$ and $\frac{\ell}{2}$, respectively. In particular, there is no need to store the labels $\mathbf{Q}_1\mathbf{e}_{1,1} + \mathbf{J}_1\mathbf{e}_{2,1}$.

8.1.2 Refined analysis for BJMM

The BJMM algorithm considered in this section is a simplified variant of the algorithms presented in Chapter 6 in the following sense:

- We only employ one layer of representations, i.e. we only analyse the generalised ISD framework instantiated with Basic-Reps. Thus the algorithm depends on the three parameters ℓ, p and δ with a fixed number of $\rho_1 = \binom{p}{p/2}\binom{k+\ell-p}{\delta}$ representations.
- We omit all duplicate checks when merging different lists.
- We check *all* candidate solutions for correctness, i.e. even those with weight $\neq p$ (see Remark 5.2.2).

The first point implies that the resulting computation tree will consist of three layers, starting with the computation of the base lists on layer two and ending with the final output list on layer zero. This simple structure eventually allows to derive a manageable formula for the workload per iteration.

Obviously, the second point simplifies the cost analysis for all merging steps. Moreover, it does not affect the original analysis for the merging between layer two and one at all. Due to the disjoint construction of the four base lists, no duplicate matchings occur. Omitting the duplicate check between layer one and zero only affects the number of candidate solutions that have to be processed in the final extension step which implies that some candidate solutions might appear several times. However, when fixing $r = \lfloor \log \rho \rfloor$, the number of duplicate solutions, and thus the additional workload, is very small (according to Section 6.4.1).

The third point is motivated as follows. Let $(\mathbf{e}_1, \mathbf{e}_2)$ be a collision in $\mathcal{L}_1 \times \mathcal{L}_2$. Checking whether this collision is weight-consistent requires one vector addition $\mathbf{e}_1 + \mathbf{e}_2$ in $\mathbb{F}_2^{k+\ell}$. Once the Hamming weight of $\mathbf{e}_1 + \mathbf{e}_2$ is known, it is very simple to check for correctness, even if $\mathrm{wt}(\mathbf{e}_1 + \mathbf{e}_2) = p' \neq p$, using only $2p'(\omega - p' + 1) \approx 2p(\omega - p + 1)$ extra additions on average. Keeping weight-inconsistent solutions in the final candidate list prevents a cumbersome computation of the actual expected number of *weight-consistent* candidate solutions. The negative effect on the running time turns out to be insignificant for all parameters under consideration.

Let us now analyse the cost of all steps in the BJMM algorithm. Recall that the base lists contain all vectors $\mathbf{x}$ of length $\frac{k+\ell}{2}$ (or more precisely all vectors of length $k+\ell$ whose support is contained in a randomly chosen subset in $\{1, \dots, k+\ell\}$ of size $\frac{k+\ell}{2}$) and weight $\frac{p}{4} + \frac{\delta}{2}$.

Workload per iteration

Building and merging base lists. Using intermediate sums, each of the four base lists can be computed using

$$\binom{\frac{k+\ell}{2}}{\frac{p}{4} + \frac{\delta}{2}} \ell \tag{8.9}$$

additions when computing the labels $\mathbf{Q}\mathbf{x}+\mathbf{t}$ on *all* ℓ coordinates for every initial vector $\mathbf{x}$. Since the first merge between layer two and one only considers the first r coordinates, one could alternatively compute $\mathbf{Q}\mathbf{x} + \mathbf{t}$ on r coordinates only. However, computing the label on all ℓ coordinates allows to update the labels for all collisions $(\mathbf{x}, \mathbf{y})$ that occur between layer two and one using only *one* addition of the precomputed vectors $\mathbf{Q}\mathbf{x}$ and $\mathbf{Q}\mathbf{y}$ on $\ell - r$ coordinates (instead of $\frac{p}{2} + \delta$ vector additions in $\mathbb{F}_2^{\ell-r}$ without the precomputation on the bottom layer). Clearly, all base lists contain $\binom{(k+\ell)/2}{p/4+\delta/2}$ elements and one expects $\binom{(k+\ell)/2}{p/4+\delta/2}^2 2^{-r}$ collisions where $r = \lfloor \log \rho \rfloor$ is fixed. Thus, both $\mathcal{L}_1$ and $\mathcal{L}_2$

can be computed with an expected number of

$$\frac{\binom{\frac{k+\ell}{2}}{\frac{p}{4}+\frac{\delta}{2}}^2}{2^r}(\ell - r) \tag{8.10}$$

additions.

Computing and checking candidate solutions. The expected number of collision between the intermediate lists $\mathcal{L}_1$ and $\mathcal{L}_2$ is given by

$$\frac{|\mathcal{L}_1| \cdot |\mathcal{L}_2|}{2^{\ell-r}} = \frac{\binom{\frac{k+\ell}{2}}{\frac{p}{4}+\frac{\delta}{2}}^4}{2^{\ell+r}} \quad .$$

Every collision $(\mathbf{e}_1, \mathbf{e}_2)$ can be checked for correctness by computing $p' = \mathrm{wt}(\mathbf{e}_1 + \mathbf{e}_2)$ and checking $\mathrm{wt}(\mathbf{Q}(\mathbf{e}_1 + \mathbf{e}_2) + \mathbf{s}) = \omega - p'$ on the remaining $n - k - \ell$ coordinates. The first step requires $k + \ell$ additions whereas the second can be done with $2p'(\omega - p' + 1) \approx 2p(\omega - p + 1)$ additions on average using early abort. Thus, computing and checking candidate solutions can implemented with

$$\frac{\binom{\frac{k+\ell}{2}}{\frac{p}{4}+\frac{\delta}{2}}^4}{2^{\ell+r}}[k + \ell + 2p(\omega - p + 1)] \tag{8.11}$$

additions on average.

Computing quasi standard forms. Computing a quasi standard form

$$\begin{pmatrix} \mathbf{Q} & \mathbf{0} \\ \mathbf{Q}' & \mathbf{I}_{n-k-\ell} \end{pmatrix}$$

can naively be done by first computing the partial standard form $\begin{pmatrix} \mathbf{Q}' & \mathbf{I}_{n-k-\ell} \end{pmatrix}$ and then creating the upper half $\begin{pmatrix} \mathbf{Q} & \mathbf{0} \end{pmatrix}$ by adding appropriate rows from the lower half. According to Eq.(8.1), the first step costs $\frac{(n-k-\ell)^2(n+k+\ell)}{2}$ additions and the second step requires $\frac{\ell(k+\ell)}{2}$ extra additions on average. The overall number of additions is thus given by

$$\frac{(n-k-\ell)^2(n+k+\ell) + \ell(k+\ell)}{2} \quad . \tag{8.12}$$

Combining Eq.(8.9)-(8.12) allows to estimate the overall (average) workload per iteration as

$$\begin{aligned} C_{\mathsf{BJMM}}(n,k,\omega,\ell,p,\delta) = 4\binom{\frac{k+\ell}{2}}{\frac{p}{4}+\frac{\delta}{2}}\ell + 2\frac{\binom{\frac{k+\ell}{2}}{\frac{p}{4}+\frac{\delta}{2}}^2}{2^r}(\ell - r) \\ + \frac{\binom{\frac{k+\ell}{2}}{\frac{p}{4}+\frac{\delta}{2}}^4}{2^{\ell+r}}[k + \ell + 2p(\omega - p + 1)] + \frac{(n-k-\ell)^2(n+k+\ell) + \ell(k+\ell)}{2} \quad . \end{aligned} \tag{8.13}$$

Number of iterations

Recall that a single iteration of the BJMM algorithm is successful if the following two independent events occur:

- A good permutation is picked.
- At least one representation of the wanted solution is contained in the list of candidate solutions.

The first event occurs with probability

$$\frac{\binom{k+\ell}{p}\binom{n-k-\ell}{\omega-p}}{\binom{n}{\omega}} \tag{8.14}$$

and the second event occurs with probability

$$[1-\exp(-2^{-r}\rho)]\frac{\binom{\frac{k+\ell}{2}}{\frac{p}{4}+\frac{\delta}{2}}^4}{\binom{k+\ell}{\frac{p}{2}+\delta}^2} \ . \tag{8.15}$$

Here, the first factor represents the probability that at least one representation $(\mathbf{e}_1, \mathbf{e}_2)$ matches a randomly chosen target vector $\mathbf{t}$, i.e. $\mathbf{Q}\mathbf{e}_1 = \mathbf{t}$ (neglecting dependencies amongst different $\mathbf{e}_1$). The second factor takes into account that one needs to choose a good partition when constructing the base lists for both $\mathbf{e}_1$ and $\mathbf{e}_2$. Altogether, the expected number of iterations is given by

$$N_{\mathsf{BJMM}}(n,k,\omega,\ell,p,\delta) = \frac{\binom{n}{\omega}\binom{k+\ell}{\frac{p}{2}+\delta}^2}{[1-\exp(-2^{-r}\rho)]\binom{k+\ell}{p}\binom{n-k-\ell}{\omega-p}\binom{\frac{k+\ell}{2}}{\frac{p}{4}+\frac{\delta}{4}}^4} \ . \tag{8.16}$$

Combining Eq.(8.13) and (8.16) proves the next result.

Theorem 8.1.4. *It holds*

$$\mathrm{WF}_{\mathsf{BJMM}}(n,k,\omega) = \min_{\ell,p,\delta} N_{\mathsf{BJMM}}(n,k,\omega,\ell,p,\delta)\cdot C_{\mathsf{BJMM}}(n,k,\omega,\ell,p,\delta)$$

where $0 \le \ell \le n-k$, $\max\{0, k+\ell+\omega-n\} \le p \le \min\{\omega, k\}$ *and* $0 \le \delta \le \frac{k+\ell-p}{2}$.

Remark 8.1.5. The space complexity of the simplified BJMM algorithm is given by

$$S_{\mathsf{BJMM}}(n,k,\omega,\ell,p,\delta) = n(n-k) + \left(\frac{k+\ell}{2}+\ell\right)\binom{\frac{k+\ell}{2}}{\frac{p}{4}+\frac{\delta}{2}} + (k+\ell)\frac{\binom{\frac{k+\ell}{2}}{\frac{p}{4}+\frac{\delta}{2}}^2}{2^r}$$

where $r = \lfloor \log \rho \rfloor$. The second term represents the size of a base lists. Every list entry consists of a binary vector of length $\frac{k+\ell}{2}$ and its label of length ℓ (the label is needed in order to benefit from intermediate sums when computing the labels of collisions on the next layer of the algorithm). The third term accounts for the expected size of a merged list on the first layer. Here, every list element only consists of a binary vector of length $k + \ell$ since no label needs to be stored when using hash maps. Note that only one base list and one merged list need to be stored at the same time.

8.2 Applications in Cryptanalysis

With the formulas presented in the preceding section, we can now estimate concrete security levels of different cryptographic assumptions related to the CSD problem, namely we study

- the hardness of inverting the McEliece T-OWF as introduced in Chapter 3.2,
- the hardness of the (computational) *Learning Parities with Noise (LPN)* problem, see Definition 8.2.1.

8.2.1 McEliece OWF and Variants

We revisit the parameters of Bernstein et al. as proposed in [BLP08] and presented in Table 3.1 for three security levels (i.e. 80 Bit, 128 Bit and 256 Bit). In Table 8.1 we compare the binary work factors (column $\log \mathrm{WF}$) of Stern's algorithm, BCD and BJMM and also give the resulting space complexity. For all algorithms, we restricted to parameters that would actually allow for a straightforward implementation, namely:

- p must be even in Stern's algorithm,
- p, q and ℓ must be even in BCD and
- $k + \ell$ and $\frac{p}{2} + \delta$ must be even in BJMM.

Note that allowing for arbitrary parameters is also possible by allowing for unbalanced lists (this would complicate the analysis but might allow for slightly improved results). If the code parameter k is odd, one needs to implement unbalanced base list both in Stern's algorithm and BCD.

sec. level	Stern				BCD					BJMM				
	ℓ	p	$\log \mathrm{WF}$	space	ℓ	p	q	$\log \mathrm{WF}$	space	ℓ	p	δ	$\log \mathrm{WF}$	space
80 Bit	30	6	**80.5**	34.7	30	6	0	**80.5**	34.7	73	12	2	**77.9**	47.4
128 Bit	33	6	**128.1**	38.1	52	8	2	**127.9**	51.0	126	20	4	**121.6**	73.1
256 Bit	74	14	**255.9**	78.3	94	16	2	**254.2**	92.2	285	46	9	**235.9**	153.8

Table 8.1: Binary work factors for generic attacks on the McEliece T-OWF.

Note that BCD does not improve over Stern's algorithm for the lowest security level (and even its effect on higher security levels is marginal). In contrary, BJMM "breaks" all three security levels but uses significantly more memory. For example, even the attack against 128-Bit security requires $\approx 2^{73} \approx 1.2 \cdot 10^9$ TB (Terabyte) of memory which is totally out of scope. For a better comparison, we also computed optimal parameters with restricted memory capacity: We defined the memory consumption of Stern's algorithm as a limit for both BCD and BJMM, i.e. we only allowed for a space complexity of 2^{40} for the first two security levels and 2^{80} for the highest security level, see Table 8.2. In summary, BJMM still performs faster by a factor 2^{10} for the highest security but does no longer threaten lower security levels.

security level	BCD					BJMM				
	ℓ	p	q	$\log \mathrm{WF}$	space	ℓ	p	δ	$\log \mathrm{WF}$	space
80 Bit	30	6	0	**80.5**	34.7	43	8	0	**79.8**	39.8
128 Bit	33	6	0	**128.1**	38.1	46	6	1	**127.8**	35.0
256 Bit	76	12	2	**254.6**	75.1	127	18	3	**245.5**	79.7

Table 8.2: Binary work factors for generic attacks on the McEliece T-OWF with restricted memory.

8.2.2 Learning Parities with Noise

In this section, we study the hardness of the computational *Learning Parities with Noise (LPN)* problem as defined below. In the sequel, Ber_η denotes the **Bernoulli distribution** with noise parameter $\eta \in (0, \frac{1}{2})$. More precisely, $e \leftarrow \mathrm{Ber}_\eta$ means that the error bit $e \in \mathbb{F}_2$ is distributed according to a random variable with $\mathbf{Pr}[e = 1] = \eta$ (and $\mathbf{Pr}[e = 0] = 1 - \eta$). Similarly, $\mathrm{Bin}_\eta(n)$ denotes the **binomial distribution** with parameters $n \in \mathbb{N}$ and $\eta \in (0, \frac{1}{2})$. Thus, $\mathbf{e} \leftarrow \mathrm{Bin}_\eta(n)$ describes an error vector of length n whose coordinates are independently drawn from Ber_η. The **LPN oracle** $\Pi_{\mathbf{m},\eta}$ with secret $\mathbf{m} \in \mathbb{F}_2^k$ and noise ratio η returns independent noisy inner products of $\mathbf{m}$, i.e. it outputs pairs $(\mathbf{g}, \mathbf{m}^\intercal \cdot \mathbf{g} + e)$ where $\mathbf{g} \in_R \mathbb{F}_2^k$ and $e \leftarrow \mathrm{Ber}_\eta$ are independently chosen for every oracle query. Alternatively, querying the LPN oracle n times and arranging the single samples $\mathbf{g}_i$ as column vectors in a $(k \times n)$-matrix $\mathbf{G}$ gives the distribution

$$(\mathbf{G}, \mathbf{m}^\intercal \mathbf{G} + \mathbf{e}) \tag{8.17}$$

with $\mathbf{e} \leftarrow \mathrm{Bin}_\eta(n)$. Note that recovering $\mathbf{m}$ in Eq.(8.17) is exactly the problem of decoding a random code with generator matrix $\mathbf{G}$ where the error vector has expected Hamming weight ηn. The computational LPN problem is to recover $\mathbf{m}$ given access to the LPN oracle $\Pi_{\mathbf{m},\eta}$ as formalised next.

Definition 8.2.1 (LPN problem)**.** Let $k \in \mathbb{N}$ be fixed. We say that an algorithm $\mathcal{A}$ $(\mathsf{q}, \mathsf{t}, \mathsf{m}, \theta)$-solves the LPN problem with noise parameter $\eta \in (0, \frac{1}{2})$ if $\mathcal{A}$ has binary work

factor $\leq \mathsf{t}$, requires at most m bits of memory, makes at most q oracle queries and succeeds with probability θ, i.e. for every $\mathbf{m} \in \mathbb{F}_2^k$ it holds

$$\mathbf{Pr}\left[\mathcal{A}^{\Pi_{\mathbf{m},\eta}}(1^k) = \mathbf{m}\right] \geq \theta$$

where the probability space is defined over the random coins of $\Pi_{\mathbf{m},\eta}$ and $\mathcal{A}$.

Thus, LPN algorithms might be compared to the following (possibly opposing) criteria.

- *Time complexity:* The best known algorithm is due to Blum, Kalai and Wasserman [BKW03]. The BKW algorithm offers slightly subexponential time complexity $2^{\mathcal{O}(k/\ln k)}$ for every *fixed* noise rate η and requires $2^{\mathcal{O}(k/\ln k)}$ oracle queries. A more practical variant of the BKW algorithm is due to Levieil and Fouque [LF06] as briefly described in Section 8.2.2.

- *Oracle complexity:* The BKW algorithm inherently relies on the availability of $2^{\mathcal{O}(k/\ln k)}$ samples. In [Lyu05], Lyubashevsky presented a method to apply the BKW algorithm when only $k^{1+\varepsilon}$ samples (for some $\varepsilon > 0$) are available at the cost of a slightly increased running time of $2^{\mathcal{O}(k/\ln\ln k)}$.

- *Memory complexity:* All known algorithms require $2^{\mathcal{O}(k/\ln k)}$ memory.

As indicated by Eq.(8.17), the LPN problem can be phrased as the problem of decoding random binary codes of dimension k where the attacker can freely choose the code length n (thus the code rate $R = \frac{k}{n}$ can be made arbitrarily small). Although this connection is obvious (and well-known in the cryptographic community), it has not been concretely exploited in cryptanalysis (in [FMICM06], Fossorier et al. formulate the LPN problem as a decoding problem but do not use ISD algorithms for their attack). While ISD algorithms have inferior time complexity of $2^{\mathcal{O}(k)}$ compared to the BKW algorithm, studying the *concrete* hardness of the LPN problem in terms of a decoding problem might be interesting due to the following reasons:

- For small noise rates η, ISD algorithms have actual running time $2^{c(\eta)k}$ for a small constant $c(\eta) \ll 1$. Consequently, ISD algorithms might outperform the BKW algorithm for moderate dimensions k.

- ISD algorithms need a remarkably smaller number of oracle queries (in fact, permuting error positions can be seen as a way of *recycling* oracle queries).

- ISD algorithms allow for a reduced memory consumption.

The above points are very relevant for some interesting cryptographic applications: For example, the HB authentication protocol [HB01] and its variants, e.g. [JW05, KPCJV11] to name a few, are most efficient for rather small noise rates η. In a nutshell, a large noise rate results in a rather high probability of rejecting an honest identity (in a single round of the protocol). Consequently, one needs a larger number of rounds in order to keep the overall *completeness error* small. Moreover, the number of oracle queries available to a

potential attacker depends on the number of observable protocol executions (in many practical applications, this number will be much smaller than the required number of oracle queries for the BKW algorithm). These two facts motivate the following central question:

Is the hardness of concrete LPN instances affected by ISD algorithms?

We will answer this question in the affirmative for all practical instances $k \in \{128, \ldots, 1024\}$ with low noise rates $\eta \leq 0.05$. Even for larger noise rates $\eta = 0.125$ the ISD attacks compare reasonably well with the BKW algorithm (in particular when taking the number of oracle queries into account). We actually break the parameter set $k = 768$ and $\eta = 0.05$ that has been recommended in [LF06] for actual use in the HB protocol with 2^{72} binary operations (cf. Table 8.6) opposed to the estimated ≈ 100 Bit security in [LF06] (see Table 8.3). Besides this, we initiate an asymptotic study of the resulting algorithms and leave some interesting open questions for future research.

The BKW algorithm

A detailed description of the BKW algorithm goes beyond the scope of this thesis. The main idea of the original BKW algorithm is to obtain different unit vectors $\mathbf{u}$ by adding a small number of ν LPN samples $\mathbf{g}_i$. By adding the corresponding labels $\mathbf{m}^\top \mathbf{g}_i + e_i$, one obtains a label for $\mathbf{u}$ with increased noise $\frac{1+\delta^\nu}{2}$ where $\delta := 1 - 2\eta$ for convenience. Now, the i-th unit vector $\mathbf{u}_i$ leaks information about the i-th bit of the secret $\mathbf{m}$ due to $\mathbf{m}^\top \cdot \mathbf{u}_i = m_i$. If ν is "small enough" this information might be sufficient to determine m_i with high probability from an appropriately large set $\mathcal{S}(i)$ of labeled unit vectors $\mathbf{u}_i$. The main idea to compute the sets $\mathcal{S}(i)$ is to apply Wagner's generalised birthday algorithm [Wag02]. Therefore, every sample $\mathbf{g}_j$ is thought as a vector of a blocks of length b (i.e. $k = ab$) and one proceeds as follows:

1. Start from a large set $\mathcal{S}_0$ of samples $\mathbf{g}_j$ and partition this set according to the first b bits in block one. For every set in this partition, randomly choose a sample $\mathbf{g}_j$ which is added to all other samples $\mathbf{g}_k$ in the same partition. All sums $\mathbf{g}_j + \mathbf{g}_k$ are added to the set $\mathcal{S}_1$ (i.e. all samples in $\mathcal{S}_1$ are zero on the first b bits).

2. Proceed similarly for the next $a - 2$ blocks.

3. Look for unit vectors in the final set $\mathcal{S}_{a-1}$.

In [BKW03], the authors show how to choose a and b and the initial number of samples in order to recover $\mathbf{m}$ with high probability in time $2^{\mathcal{O}(k/\ln k)}$. In [LF06], Levieil and Fouque proposed some slight modifications in order to increase the practical performance of the BKW algorithm. The most important modification concerns the final step of the algorithm: Restricting to unit vectors in $\mathcal{S}_{a-1}$ unnecessarily ignores a lot of information. Instead, one can directly compute the maximum-likelihood candidate solution for $\mathbf{m}$ on the respective b bits (which can be done efficiently using the Walsh-Hadamard

transform). We refrain from a deeper description - which can be found in [LF06] - and merely state their main theorem which allows to estimate the concrete complexity of the BKW algorithm for different LPN instances, see Table 8.3.

Theorem 8.2.2 (Theorem2 in [LF06]). *For $k = ab$, the BKW algorithm $(\mathsf{q}, \mathsf{m}, \mathsf{t}, \theta)$-solves the LPN problem with noise η where*

$$\begin{aligned} \mathsf{q} &= (8b + 200)\delta^{-2^a} + (a-1)2^b \ , \\ \mathsf{t} &= ka\mathsf{q} \ , \\ \mathsf{m} &= k\mathsf{q} + b2^b \ , \\ \theta &= 1/2 \end{aligned}$$

and $\delta = 1 - 2\eta$.

Optimal parameters $a \in \mathbb{N}$ with respect to a minimal running time t can be found in Table 8.3. We generously ignored rounding issues for b. The resulting (optimistic) performance of the BKW algorithm will serve as a benchmark for our decoding-based approach. In Table 8.3, the values $\mathsf{q}, \mathsf{m}, \mathsf{t}$ can be found in the form $\boxed{a \parallel \log \mathsf{q} \mid \log \mathsf{m} \mid \log \mathsf{t}}$ for different pairs (k, η).

k \ η	$\frac{1}{100}$				$\frac{1}{20}$				$\frac{1}{8}$				$\frac{1}{4}$			
128	8	18	26	**28**	6	23	30	**33**	5	27	34	**36**	4	33	40	**42**
256	9	31	39	**42**	7	39	47	**49**	6	44	53	**55**	5	53	61	**63**
512	10	54	63	**66**	8	66	75	**78**	7	75	84	**87**	6	87	96	**99**
768	11	73	82	**86**	9	89	98	**101**	7	112	121	**124**	6	130	139	**142**
1024	11	96	106	**109**	9	116	126	**129**	8	130	140	**143**	7	148	158	**161**

Table 8.3: Attacking LPN with the BKW algorithm.

Of particular interest is the parameter set $(k, \eta) = (768, \frac{1}{20})$ that has been suggested for actual use in the HB protocol by the authors of [LF06] (for at least 80 Bit security). Will shortly see that ISD allows to break the claimed security level.

Solving LPN via Syndrome Decoding

From now on, $\mathbf{G}$ denotes the generator matrix obtained from n LPN samples, $\mathbf{c} = \mathbf{m}^\intercal\mathbf{G}+\mathbf{e}$ denotes the corresponding erroneous label vector, $\mathbf{H}$ denotes a parity check matrix for the code generated by $\mathbf{G}$ and $\mathbf{s} = \mathbf{Hc} = \mathbf{He}$ is the syndrome of $\mathbf{c}$. When transforming the LPN problem into a decoding problem, the attacker has free choice over the code length n and the target weight ω of the wanted error vector $\mathbf{e}$ (recall that ISD algorithms require ω as an input).

Choosing the code length n. Intuitively, when phrasing the LPN problem as a

decoding problem, an attacker might be tempted to choose a large code dimension n. This allows for unique decoding even for large noise rates η. Moreover, it allows for a favourable parameter choice in ISD algorithms: Very large n increases the chance of guessing a good permutation even for relatively small values of p (and q and δ in BCD and BJMM, respectively).

More precisely, depending on the actual choice of n and ω, the equation $\mathbf{He} = \mathbf{s}$ will have i) no solution, ii) a unique solution or iii) multiple solutions with high probability. The first case occurs for $\omega \ll \eta n$ and makes recovery of the wanted error vector $\mathbf{e}$ impossible. As indicated by Proposition 8.2.3, choosing $\omega = \lceil \eta n \rceil$ gives a decoding instance that contains the wanted error vector $\mathbf{e}$ with probability $\geq \frac{1}{2}$.

Proposition 8.2.3. *Let $\eta \in (0, \frac{1}{2})$ and $\mathbf{e} \leftarrow \mathrm{Bin}_\eta(n)$. It holds*

$$\mathbf{Pr}\left[\mathrm{wt}(\mathbf{e}) \leq \lceil \eta n \rceil\right] \geq \frac{1}{2} \ . \tag{8.18}$$

We thus fix $\omega = \lceil \eta n \rceil$ in all attacks. Alternatively, choosing ω slightly larger would allow to push the probability in Eq.(8.18) close to 1.

Remark 8.2.4. In order to apply Proposition 8.2.3, we need to modify all ISD algorithms (in a straightforward way) to search for all solutions of $\mathbf{He} = \mathbf{s}$ with weight $\leq \omega$ (instead of weight $= \omega$).

Let us now discuss the case where n is chosen too small, i.e. $\mathbf{He} = \mathbf{s}$ has multiple solutions with high probability. Thus, every "successful iteration" of the respective ISD algorithm might yield multiple candidate error vectors $\tilde{\mathbf{e}}$ (or candidate secrets $\tilde{\mathbf{m}}$) for the LPN problem. Since we expect a rather large number of iterations, we will eventually be faced with a possibly large set $\mathcal{M}$ of candidate solutions. In order to determine the correct $\mathbf{m}$, one thus has to carry out an additional statistical test. Therefore, one queries the LPN oracle for another sample $(\mathbf{G}', \mathbf{c}')$ and simply outputs $\tilde{\mathbf{m}} \in \mathcal{M}$ which minimizes $\mathrm{wt}(\tilde{\mathbf{m}}^\top \mathbf{G}' + \mathbf{c}')$. Depending on the noise rate η and the size of $\mathcal{M}$, the required number of additional oracle queries can become very large. We thus prefer to chose n large enough in order to avoid multiple solutions: There are $N(n, \eta) := \sum_{i=0}^{\lceil \eta n \rceil} \binom{n}{i} - 1$ vectors $\tilde{\mathbf{e}} \neq \mathbf{e}$ of length n and weight $\leq \lceil \eta n \rceil$. Every single vector fulfils $\mathbf{H}\tilde{\mathbf{e}} = \mathbf{s}$ with probability $\frac{1}{2^{n-k}}$. By the union bound it follows

$$\mathbf{Pr}\left[\mathbf{H}\tilde{\mathbf{e}} \neq \mathbf{s} \forall \tilde{\mathbf{e}} \neq \mathbf{e}\right] = 1 - \mathbf{Pr}\left[\bigvee_{\tilde{\mathbf{e}} \neq \mathbf{e}} \{\mathbf{H}\tilde{\mathbf{e}} = \mathbf{s}\}\right] \geq 1 - \sum_{\tilde{\mathbf{e}} \neq \mathbf{e}} \mathbf{Pr}\left[\mathbf{H}\tilde{\mathbf{e}} = \mathbf{s}\right] \geq 1 - \frac{N}{2^{n-k}} \ . \tag{8.19}$$

Requiring $N(n, \eta) \leq 2^{n-k+1}$ thus guarantees a unique solution with probability $\geq \frac{1}{2}$. In summary, according to these observations, we will now study the applicability of Plain ISD, BCD and BJMM to the LPN problem under the constraints

$$\begin{aligned} \omega &= \lceil \eta n \rceil \\ N(\eta, n) &\leq 2^{n-k+1} \ . \end{aligned} \tag{8.20}$$

Consequently, the algorithms will succeed in solving the LPN problem with probability

$$\theta \geq \frac{1}{2}\left(1 - \frac{N(n,\eta)}{2^{n-k}}\right) \geq \frac{1}{4} \ . \tag{8.21}$$

Attacking LPN via Plain ISD. In every iteration of Plain ISD one needs to check $\mathrm{wt}(\mathbf{Ts}) = \omega$ where $\mathbf{T}$ denotes the transformation matrix that brings the column-permuted parity check matrix into standard form. Note that we do not need to compute the corresponding standard form explicitly and the costs for computing $\mathbf{T}$ is simply given by the cost of inverting the corresponding $(n-k)$-dimensional submatrix of $\mathbf{H}$ (which is $\frac{(n-k)^3}{2}$ on average). By adapting Eq.(4.5) to the case where Plain ISD is modified in order to find all $\mathbf{e}$ with $\mathrm{wt}(\mathbf{e}) \leq \omega$ (instead of $\mathrm{wt}(\mathbf{e}) = \omega$), the binary work factor of Plain ISD can be estimated as

$$\mathrm{WF}_{\mathsf{Plain}}(n,k,\omega) = \frac{\sum_{i=0}^{\omega}\binom{n}{i}}{\sum_{i=0}^{\omega}\binom{n-k}{i}} \cdot \frac{(n-k)^3}{2} \ . \tag{8.22}$$

Thus, when attacking an LPN instance with parameters k and η, we need to compute $n \in \mathbb{N}$ that minimizes $\mathrm{WF}_{\mathsf{Plain}}(n,k,\lceil \eta n \rceil)$ under the additional constraint $N(\eta,n) \leq 2^{n-k+1}$, see Eq.(8.20). In Table 8.4 we present optimal values for n (with fixed $\omega = \lceil \eta n \rceil$) and the resulting number of oracle queries $\log \mathsf{q}$, memory consumption $\log \mathsf{m}$, binary work factor $\log \mathsf{t}$ and success probability θ according to Eq.(8.21). The data is ordered as described in Figure 8.1.

n / ω	$\log \mathsf{q}$	$\log \mathsf{m}$	$\log \mathsf{t}$	θ

Figure 8.1: Legend for table entries of Table 8.4.

k \ η	$\frac{1}{100}$					$\frac{1}{20}$					$\frac{1}{8}$					$\frac{1}{4}$				
128	143 / 2	8	15	**18**	0.57	180 / 9	8	15	**33**	0.53	312 / 39	9	16	**54**	0.54	748 / 187	10	17	**86**	0.52
256	279 / 3	9	17	**24**	0.39	478 / 24	9	17	**50**	0.56	1056 / 132	11	19	**85**	0.52	3476 / 869	12	20	**145**	0.51
512	583 / 6	10	19	**36**	0.63	1680 / 84	11	20	**75**	0.53	4600 / 575	13	22	**140**	0.51	7032 / 1758	13	22	**259**	0.51
768	996 / 10	10	20	**44**	0.59	4840 / 242	13	22	**97**	0.52	7256 / 907	13	23	**194**	0.51	11392 / 2848	14	24	**371**	0.51
1024	1700 / 17	11	21	**50**	0.56	6280 / 314	13	23	**119**	0.52	12272 / 1534	14	24	**246**	0.51	15056 / 3764	14	24	**483**	0.5

Table 8.4: Attacking LPN via Plain ISD.

In comparison to the BKW algorithm, Plain ISD improves by almost a square-root factor for the smallest noise rate $\eta = \frac{1}{100}$ and all k. For the next noise level $\eta = \frac{1}{20}$, the binary work factor of the Plain ISD attack is roughly the same as for BKW but can be achieved with a much smaller number of oracle queries and a drastically reduced memory consumption. For example, the parameter set $(k, \eta) = (256, \frac{1}{20})$ must be considered practically broken by ISD algorithms whereas the huge memory consumption of ≈ 18 TB makes the application of the BKW algorithm much more expensive. For larger noise rates, one should prefer the BKW algorithm over ISD.

Attacking LPN via BCD. Similarly to Plain ISD, we need to find an optimal code length $n \in \mathbb{N}$ that minimizes the binary work factor $\mathrm{WF}_{\mathsf{BCD}}(n, k, \lceil \eta n \rceil)$ as defined in Theorem 8.1.2. Note that $\mathrm{WF}_{\mathsf{BCD}}$ is implicitly defined for optimal algorithm parameters p and q. A completely accurate analysis requires to modify all formulas to the case, where BCD recovers error vectors of weight $\leq \omega$, e.g. Eq.(8.6) must be replaced by $\sum_{i=0}^{p/2} \binom{k/2}{i} + \min\{1, q\} \sum_{j=0}^{q/2} \binom{\ell/2}{j}$. We omit a full description for ease of presentation. Nevertheless, the data presented in Table 8.5 has been computed using exact formulas. In Table 8.5 one finds optimal n (and fixed $\omega = \lceil \eta n \rceil$) and optimal algorithm parameters ℓ, p, q together with the resulting number of oracle queries q, memory and time complexities m and t for different pairs (k, η) as explained in Figure 8.2.

n	ℓ	p	q
ω	$\log \mathsf{q}$	$\log \mathsf{m}$	$\log \mathsf{t}$

Figure 8.2: Legend for table entries of Table 8.5.

In contrary to Table 8.4, we omit the θ values which are all in the range $(\frac{1}{4}, \frac{3}{4})$ for simplicity.

k \ η	$\frac{1}{100}$				$\frac{1}{20}$				$\frac{1}{8}$				$\frac{1}{4}$			
128	143	2	2	0	178	10	4	0	504	32	8	2	1824	42	10	4
	2	8	15	**17**	9	8	19	**27**	63	9	31	**41**	456	11	38	**67**
256	279	2	2	0	519	22	6	0	2880	44	10	2	3700	42	10	2
	3	9	17	**21**	26	10	27	**38**	360	12	41	**64**	925	12	41	**120**
512	594	16	4	0	2704	34	8	0	5868	50	10	2	7628	62	12	4
	6	10	25	**27**	136	12	37	**57**	734	13	47	**112**	1907	13	57	**227**
768	885	18	4	0	6236	36	8	0	8800	54	10	2	11592	90	20	4
	9	10	26	**35**	312	13	40	**75**	1100	14	51	**161**	2898	14	84	**335**
1024	1590	30	6	0	10508	38	8	0	11928	58	10	2	15528	106	24	4
	16	11	35	**39**	526	14	42	**93**	1491	14	54	**211**	3882	14	100	**443**

Table 8.5: Attacking LPN via BCD.

Compared to Plain ISD, the BCD attack requires a slightly increased number of

oracle queries but additionally allows to break the 80 Bit security of the parameter sets $(768, \frac{1}{20})$, $(256, \frac{1}{8})$ and $(128, \frac{1}{4})$. In particular, the claimed 80 Bit security level of the suggested parameter set $(768, \frac{1}{20})$ of [LF06] is broken by a factor of 2^5.

Attacking LPN via BJMM. Analogously to BCD, we now present optimal values for n and the BJMM parameters ℓ, p and δ minimizing $\mathrm{WF}_{\mathsf{BJMM}}(n, k, \lceil \eta n \rceil)$ (when adapted to the case where we actually search for all vectors of weight $\leq \lceil \eta n \rceil$) in Table 8.6 below. Although the efficiency for solving almost all instances can be improved (compared with BCD), we did not manage to break the 80 Bit security of any additional parameter set. However, breaking the parameter set $(k, \eta) = (1024, \frac{1}{20})$ is now within the range of possibility (we sketch some ideas for improved attacks that further reduce the security of this parameter set in the concluding section below).

n	ℓ	p	δ
ω	$\log \mathsf{q}$	$\log \mathsf{m}$	$\log \mathsf{t}$

Figure 8.3: Legend for table entries of Table 8.6.

k \ η	$\frac{1}{100}$				$\frac{1}{20}$				$\frac{1}{8}$				$\frac{1}{4}$			
128	149	20	2	1	179	26	6	1	448	26	8	0	2496	58	18	1
	2	8	14	**16**	9	8	21	**26**	56	9	25	**41**	624	12	38	**67**
256	279	14	2	1	533	32	8	0	3928	56	14	1	4988	68	18	1
	3	9	15	**22**	27	10	29	**38**	491	12	39	**63**	1247	13	46	**118**
512	559	22	4	0	2636	50	10	1	9984	64	14	1	15104	110	26	3
	6	10	24	**28**	132	12	36	**55**	1248	14	46	**109**	3776	14	64	**220**
768	886	40	6	1	9596	68	12	2	45696	86	16	2	61248	178	44	4
	9	10	30	**34**	480	14	43	**72**	5712	16	52	**153**	15312	16	101	**317**
1024	1298	42	6	1	19288	72	12	2	40704	100	18	3	42736	314	84	8
	13	11	31	**39**	965	15	45	**89**	5088	16	58	**201**	10684	16	165	**416**

Table 8.6: Attacking LPN via BJMM.

Asymptotic Analysis

We only present an asymptotic analysis for the LPN attack based on Plain ISD. For this purpose, we parametrise $n = \nu \cdot k$ for some $\nu > 1$. The simple structure of Plain ISD allows to express the attack's running time as $2^{c(\eta)k + o(k)}$ for large enough k and ν due to the next lemma.

Lemma 8.2.5. *Let $\eta \in (0, \frac{1}{2})$ and $\nu > 1$. For $n = \lceil \nu k \rceil$ and $\omega = \lceil \eta n \rceil$ it holds*

$$c(\eta, \nu) := \lim_{k\to\infty} \frac{1}{k} \log \mathrm{WF}_{\mathsf{Plain}}(n, k, \omega) = \nu \left[\mathrm{H}(\eta) - \mathrm{H}\left(\eta \frac{\nu}{\nu - 1}\right)\right] + \mathrm{H}\left(\eta \frac{\nu}{\nu - 1}\right) \tag{8.23}$$

and

$$\lim_{\nu\to\infty} c(\eta, \nu) = -\log(1 - \eta) \ . \tag{8.24}$$

Proof. Eq.(8.23) directly follows from Eq.(8.22) and the standard approximations for binomial coefficients, i.e. Eq.(2.14) and Eq.(2.15). For the second claim, we write the first summand $\nu\left[\mathrm{H}(\eta) - \mathrm{H}\left(\eta \frac{\nu}{\nu-1}\right)\right]$ of $c(\eta, \nu)$ as $\frac{f(\eta,\nu)}{g(\eta,\nu)}$ with $f(\eta, \nu) = \mathrm{H}(\eta) - \mathrm{H}\left(\eta \frac{\nu}{\nu-1}\right)$ and $g(\eta, \nu) = \frac{1}{\nu}$. Since $\lim_{\nu\to\infty} f(\eta, \nu) = 0 = \lim_{\nu\to\infty} g(\eta, \nu)$, L'Hôpital's rule gives

$$\begin{aligned}
\lim_{\nu\to\infty} c(\eta, \nu) &= \lim_{\nu\to\infty} \frac{f'(\eta, \nu)}{g'(\eta, \nu)} + \lim_{\nu\to\infty} \mathrm{H}\left(\eta \frac{\nu}{\nu - 1}\right) \\
&= \lim_{\nu\to\infty} \frac{\nu^2}{(\nu - 1)^2} \eta \left[\log\left(\eta \frac{\nu}{\nu - 1}\right) - \log\left(1 - \eta \frac{\nu}{\nu - 1}\right)\right] + \mathrm{H}(\eta) \\
&= \eta[\log \eta - \log(1 - \eta)] + \mathrm{H}(\eta) = -\log(1 - \eta)
\end{aligned}$$

as claimed. □

In Figure 8.4 we compare the theoretical complexity of the Plain ISD attack on LPN given by Lemma 8.2.5 with the real attack cost as presented in Table 8.4 for different η. When neglecting the costs for the Gaussian Elimination in Plain ISD, the theoretical estimate is fairly accurate (although we used rather small values for n, or equivalently small values for ν, in practice).

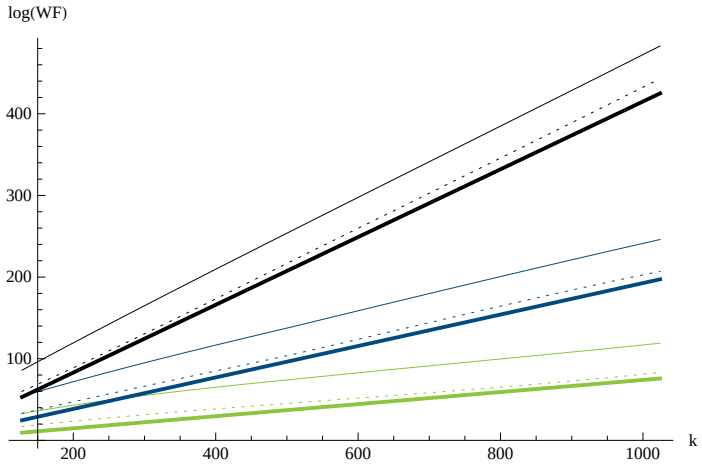


Figure 8.4: Comparison of $-\log(1 - \eta)k$ (thick lines), $\log \mathrm{WF}_{\mathsf{Plain}}$ (thin lines) and $\log \mathrm{WF}_{\mathsf{Plain}}$ without costs of Gaussian elimination (dotted lines) for $\eta \in \{\frac{1}{20}, \frac{1}{8}, \frac{1}{4}\}$ (from bottom to top).

Clearly, the BKW algorithm will eventually outperform all ISD algorithms for large enough $k(\eta)$ due to its asymptotic superiority. For fixed η, the break-even point between Plain ISD and BKW is (asymptotically) given by the smallest k such that $-\log(1-\eta) > 1/\ln(k)$, i.e.

$$\log k(\eta) = \left(\frac{-1}{\log(1-\eta)}\right)^{\log(e)} \tag{8.25}$$

where $e = \exp(1)$ is Euler's number, see Figure 8.5 for an illustration. We did not find similar closed formulas for the attack cost when using BCD and BJMM and leave the asymptotic study of those attacks as an open question.

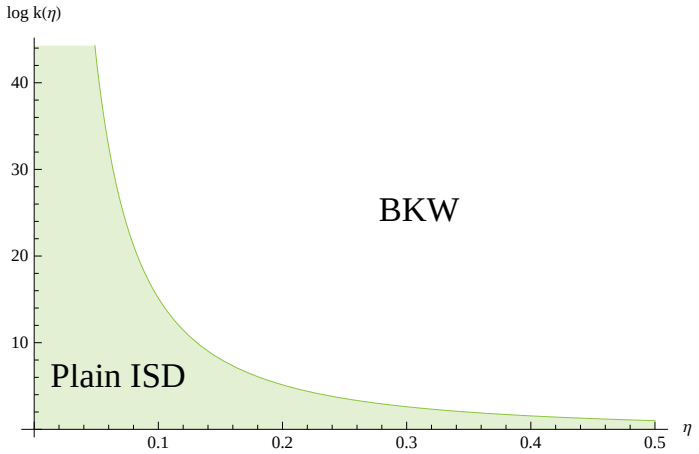


Figure 8.5: Break-even code dimension $k(\eta)$.

For $\eta = \frac{1}{20}$ we have $\log k(\eta) \approx 42$, i.e. even in a long-term decoding attacks will be more efficient than BKW.

Conclusion and Open Questions

We have seen that ISD algorithms allow to solve many practically relevant instances of the LPN problem more efficiently than the asymptotically faster BKW algorithm. In Table 8.7 we summarise our findings: For every parameter pair (k, η) we present the most efficient attack and indicate the actual security level (if two attacks are equally time efficient we choose the one with lower oracle complexity).

Altogether, almost all low-noise instances ($\eta \leq 0.05$) must be considered insecure. Moreover, for such low-noise instances, we expect that ISD algorithms are considerably faster than the BKW algorithm even for large $k \approx 2^{40}$ due to Eq.(8.25). Thus, every serious parameter proposal for cryptographic constructions based on the hardness of the LPN assumption has to take ISD algorithms into account.

Open Questions. We conclude this chapter with some observations that might stimulate future research. A way to improve the ISD attacks is to reduce the code dimension

k \ η	$\frac{1}{100}$	$\frac{1}{20}$	$\frac{1}{8}$	$\frac{1}{4}$
128	BJMM 16	BJMM 26	BKW 36	BKW 42
256	BCD 21	BCD 38	BKW 55	BKW 63
512	BCD 27	BJMM 38	BKW 87	BKW 99
768	BJMM 34	BJMM 72	BKW 124	BKW 142
1024	BJMM 37	BJMM 89	BKW 143	BKW 161

Table 8.7: Best known attacks and their binary work factors $\log \mathtt{t}$. Gray shaded cells mark insecure parameter sets (i.e. $\log \mathtt{t} \leq 80$). Pale green cells are broken more efficiently due to our work, full green cells were considered secure before our work.

k when phrasing the LPN problem as a decoding problem. For example, when building the generator matrix $\mathbf{G}$, one could only use those LPN samples whose first b bits vanish. In this case, the resulting code is of dimension $k - b$ and does not depend on the first b bits of the secret $\mathbf{m}$ at all. Thus, applying an ISD algorithm to the dimension-reduced code can recover all but the first b bits of $\mathbf{m}$. Once the first $k - b$ bits of $\mathbf{m}$ are known, the remaining b bits can either be guessed or computed by solving another LPN problem of dimension b (in any case, the overall attack cost will be dominated by the first step). For example, when using this idea together with BJMM, the overall complexity can be estimated as

$$\min_{b,n} 2^b \cdot n + \mathrm{WF}_{\mathsf{BJMM}}(n, k - b, \lceil \eta n \rceil)$$

where $\mathrm{WF}_{\mathsf{BJMM}}$ is implicitly defined for optimal BJMM parameters ℓ, p, q. Here, the term $2^b n$ accounts for the expected number of oracle queries needed in order to construct a code of length n and dimension $k - b$. This approach might speed up ISD attacks at the cost of an increased oracle complexity.

As a proof of concept, we applied this idea to the parameter set $(k, \eta) = (1024, \frac{1}{20})$ (our hope was to actually break the 80 Bit security of these parameters). As a result, we further reduced the binary work factor from 2^{89} (see Table 8.7) to 2^{85} by setting $n = 12032$, $\ell = 70$, $p = 12$, $\delta = 2$ and $b = 62$. This requires an increased number of $n2^b \approx 2^{76}$ oracle queries.

Alternatively, it is possible to view the initial blockwise elimination of the BKW algorithm as a way to reduce the code dimension yielding the following interesting question:

Can we obtain more efficient attacks by combining the BKW algorithm with ideas from ISD?

9 Noisy Integer Factorisation

> "We can see the point where the chip is unhappy if a wrong bit is sent and consumes more power from the environment."
>
> *Adi Shamir*

Introduction

The **RSA trapdoor one-way permutation** $f_{(N,e)}$ is probably the most widely deployed cryptographic primitive in practice. It is defined as follows:

- Pick a random **RSA modulus** $N = pq$ where p and q are random prime numbers of bit length $\frac{n}{2}$.
- Generate a key pair $(e, d) \in \mathbb{Z}^*_{\phi(N)}$ such that $ed = 1 \mod \phi(N)$ where $\phi(N)$ is Euler's totient function. The tuple (N, e) serves as the public description of the function whereas d represents the trapdoor that allows for efficient inversion.
- Define $f_{(N,e)} : \mathbb{Z}_N \to \mathbb{Z}_N$ via $f_{(N,e)}(x) := x^e \mod N$.

Note that the key pair (e, d) can be efficiently computed via the Extended Euclidean algorithm when the factorisation of N and thus $\phi(N) = (p-1)(q-1)$ is known. Since $x^{ed} = x \mod N$ one can efficiently invert $f_{(N,e)}$ using the trapdoor d. Clearly, a *necessary* condition for the onewayness of $f_{(N,e)}$ is the *average-case hardness* of the **factorisation problem**, i.e. given a random RSA moduls N it must be hard to compute the prime factors p, q in polynomial time $\text{poly}(\log N)$ with non-negligible probability. It is a long-standing open problem whether the hardness of the factorisation problem is also *sufficient* for the onewayness of $f_{(N,e)}$. At Crypto 2004, May proved the equivalence of computing d from (N, e) and factoring N [May04]. However, it is not clear whether it is actually necessary to compute d in order to invert $f_{(N,e)}$. Despite the existence of efficient quantum algorithms for the factorisation problem due to Shor [Sho97], no classical polynomial time factorisation algorithm is known.

Motivated by *side-channel attacks*, where an attacker might obtain auxiliary information about p and q, the framework of *oracle-assisted* factorisation algorithms was

introduced by Rivest and Shamir in 1985 [RS85]: They used an oracle that allowed for querying bits of p in *chosen* positions and showed that $\frac{3}{5}\log p$ queries are sufficient to factor N in polynomial time. This was later improved by Coppersmith [Cop97] to only $\frac{1}{2}\log p$ queries. In 1992, Maurer [Mau92] showed that for stronger oracles, which allow for any type of oracle queries with YES/NO answers, $\epsilon \log p$ queries are sufficient for any $\epsilon > 0$.

In this oracle-based line of research, the goal is to minimise both the power of the oracle and the number of queries to the oracle. At Crypto 2009, Heninger and Shacham [HS09] presented a (heuristic) polynomial time factorisation algorithm that works whenever a 0.57-fraction of the bits of p and q is given, provided that the given bits are uniformly spread over the whole bits of p and q. So as opposed to the oracle used by Rivest, Shamir and Coppersmith, the attacker has no control about the positions in which he receives some of the bits, but he knows the positions and on expectation the erasure intervals between two known bits are never too large.

In contrast to the aforementioned attacks which all require a limited number of *fault-free* information provided by the oracles - mostly in the form of secret key bits - we now consider a different scenario where erroneous copies $\tilde{p}$ and $\tilde{q}$ are provided by the oracle:

Noisy Integer Factorisation Problem. Let $N = pq$ be an RSA modulus with balanced prime factors p and q of bit-length $\frac{n}{2}$. Given N and noisy copies $\tilde{p}$, $\tilde{q}$ for some noise rate $0 < \delta < \frac{1}{2}$, i.e.

$$\tilde{p}_i = p_i + \mathrm{Ber}(\delta) \mod 2 \tag{9.1}$$
$$\tilde{q}_i = q_i + \mathrm{Ber}(\delta) \mod 2 \tag{9.2}$$

independently for every bit $i = 0, \ldots, \frac{n}{2} - 1$ (where $\mathrm{Ber}(\delta)$ is the Bernoulli distribution with probability δ), the task is to recover the factorisation p, q.

Clearly, the efficiency of an algorithm for the Noisy Integer Factorisation problem will depend on the noise rate δ. In particular, the noisy copies $\tilde{p}, \tilde{q}$ will not provide any helpful information for $\delta \to \frac{1}{2}$. The main goal of this chapter is to introduce a *heuristic polynomial-time algorithm* that recovers p, q from $N, \tilde{p}, \tilde{q}$ for all $\delta < 0.0837$. This algorithm was published as a special case in a more general work [HMM10] at Crypto 2010. In [HMM10], we presented a similar algorithm for a slightly generalised scenario where the attacker is given a fully redundant erroneous RSA secret key (p, q, d, d_p, d_q), i.e. $d_p = d \mod p - 1$ and $d_q = d \mod q - 1$ are the CRT exponents as specified in the PKCS#1 standard [PKCS#1]. In this case, we were able to recover the whole secret key for a much larger noise rates $\delta < 0.237$ by further exploiting the redundancy in the secret key. Compared to [HMM10], the main contribution of this chapter is two-fold:

- We provide a link to coding theory that was recently discovered in a follow-up work of Paterson et al. [PPS12]. In particular, this link allows to (heuristically) derive an upper bound on δ: For list-decoding algorithms, recovering p, q with good probability is only possible for $\delta < 0.111$. This also allows to evaluate the *optimality* of our approach (which has not been answered in [HMM10]).
- We extend the original analysis of [HMM10] in the following sense: We originally proved that our algorithm has *expected running time* $\mathcal{O}\left(n^{2+c(\delta)}\right)$ where $c(\delta)$ is a constant depending on the error rate δ. In this analysis, we computed an upper bound for the expected number of "partial solutions" processed by the algorithm but we ignored a conceivably large deviation of the actual number of partial solutions. In this work, we provide a rather technical proof that fixes this problem at the cost of a slightly increased upper bound $\mathcal{O}\left(n^{3+c(\delta)}\right)$ for the algorithm's running time.

In particular, the first point allows to integrate this chapter into the coding-theoretic framework of this thesis. We point out that restricting to the special case of factorisation allows for a much cleaner presentation of the algorithm compared to [HMM10]. Since both algorithms are very similar, the above points can easily be adapted to the general setup of [HMM10] as indicated in the conclusion.

Roadmap

The remainder of this chapter is organised as follows: Based on Hensel's lemma, we first provide a "dyadic" exponential time algorithm for the classical noise-free factorisation problem in Section 9.1. This algorithm grows a large binary computation tree whose leaves are candidate solutions to the factorisation problem. Moreover, this algorithm can easily be adapted to the erasure scenario of the HS algorithm. By viewing the leaves of the computation tree as a code, the aforementioned link to coding theory already follows. In this setup, the erasure scenario of the HS algorithm can be seen as transmitting the leaf corresponding to the correct factorisation (p, q) over a *binary erasure channel* with some erasure probability ρ. Analogously, the erroneous factorisation $(\tilde{p}, \tilde{q})$ in our scenario can be modelled by transmitting the codeword (p, q) over the *binary symmetric channel* with some crossover probability δ. Recently, Paterson et al. presented a different algorithm for the non-symmetric binary channel in [PPS12]. In Section 9.3 we will formally exploit the coding-theoretic link and show how the converse of Shannon's noisy-channel coding theorem [Sha48] yields a theoretical limit for the maximal error rate that allows for reliable recovery of (p, q). In Section 9.2, we present our factorisation algorithm and its corrected runtime analysis and we end with some concluding remarks in Section 9.4.

9.1 A Dyadic Factorisation Algorithm

In the sequel, we denote the binary representation of an n-bit integer x by $(x_{n-1}, \ldots, x_0)$ where x_0 is the least significant bit. We start by rephrasing the factorisation problem as the problem of computing integer roots of a bivariate polynomial $f(X, Y) \in \mathbb{Z}[X, Y]$: For a given N, the polynomial

$$f_N(x, y) := N - xy$$

has the unknown integer root (p, q). The idea is to compute a large set of candidate roots by iteratively lifting roots $\mod 2^i$ to roots $\mod 2^{i+1}$, i.e. by revealing the factorisation bit-by-bit. Starting from the initial root $f_N(1, 1) = 0 \mod 2$, the bivariate version of Hensel's lemma provides an efficient procedure to implement the lifting.

Lemma 9.1.1 (Hensel's lemma)**.** *Let $f \in \mathbb{Z}[X, Y]$ be a polynomial with root $f(x, y) = 0 \mod 2^i$. Then $f(x + x_i 2^i, y + y_i 2^i) = 0 \mod 2^{i+1}$ iff*

$$f(x, y) = x_i 2^i \partial_X f(x, y) + y_i 2^i \partial_Y f(x, y) \mod 2^{i+1}$$

where ∂_X and ∂_Y is the formal partial derivative of f w.r.t. X and Y, respectively.

Proof. Consider f as a polynomial in $X - x$ and $Y - y$, i.e.

$$f = \sum_{k,l} f_{k,l} (X - x)^k (Y - y)^l$$

and observe that $f(x, y) = f_{0,0}$, $\partial_X f(x, y) = f_{1,0}$ and $\partial_Y f(x, y) = f_{0,1}$. It follows

$$\begin{aligned} f(x + x_i 2^i, y + y_i 2^i) &= \sum_{k,l} f_{k,l} (x_i 2^i)^k (y_i 2^i)^l \\ &= f(x, y) + x_i 2^i \partial_X f(x, y) + y_i 2^i \partial_Y f(x, y) + x_i y_i 2^{2i} \sum_{k,l \geq 1} f_{k,l} (x_i 2^i)^{k-1} (y_i 2^i)^{l-1} \end{aligned}$$

which gives $f(x + x_i 2^i, y + y_i 2^i) = 0 \mod 2^{i+1}$ iff $0 = f(x, y) + x_i 2^i \partial_X f(x, y) + y_i 2^i \partial_Y f(x, y) \mod 2^{i+1}$. □

Given a candidate root (p', q') of f_N for the first i bits, application of Hensel's lemma yields the **lifting equation**

$$q'_i + p'_i = (N - p'q')_i \mod 2 \tag{9.3}$$

that has to be fulfilled by the next two bits of p' and q'. More precisely, every candidate root (p', q') of f_N modulo 2^i gives rise to two additional candidate roots

$$(0 \| p', (N - p'q')_i \| q') \quad \text{and} \quad (1 \| p', \overline{(N - p'q')_i} \| q')$$

modulo 2^{i+1} (where $\overline{x} = 1 \oplus x$ denotes complementing the bit x). This immediately gives the following inefficient factorisation algorithm for $N = pq$ with balanced prime factors p, q of bit length $\frac{n}{2}$.

Algorithm 8: FACTORING

input : N
output: p, q

$\mathcal{L}_0 = \{(1,1)\}$;
for $i = 1$ **to** $\frac{n}{2} - 1$ **do**
  forall $(p', q') \in \mathcal{L}_{i-1}$ **do**
    $b \leftarrow (N - p'q')_i$;
    $\mathcal{L}_i = \mathcal{L}_i \cup \{(0||p', b||q'), (1||p', \overline{b}||q')\}$;
forall $(p', q') \in \mathcal{L}_{\frac{n}{2}-1}$ **do**
  if $N = p'q'$ **then**
    return p, q

This algorithm computes a complete binary tree of depth $\frac{n}{2}$ whose leaves represent candidate factorisations for N that have to be checked for correctness. Clearly, the running time of the algorithm is exponential in n as substantiated next.

Theorem 9.1.2. *On input $N = pq$ with $\lceil \log p \rceil = \lceil \log q \rceil = \frac{n}{2}$,* FACTORING *outputs p, q in time $\mathcal{O}(n2^{n/2})$.*

Proof. When keeping track of the values $(N - p'q')$ for preceding levels, the right-hand side of Eq.(9.3) for each lifting can be obtained with a constant number of additions of n-bit integers, i.e. every lifting can be done in time $\mathcal{O}(n)$. The total number of liftings is clearly $\mathcal{O}(2^{n/2})$. Similarly, using the precomputed values $(N - p'q')$ allows to check correctness of each candidate solution in the final list $\mathcal{L}_{\frac{n}{2}-1}$ in time $\mathcal{O}(n)$. □

Remark 9.1.3. By a results of Coppersmith [Cop97] an amount of $\frac{n}{4}$ bits of p (or q) would suffice for factoring N in polynomial time. Thus, running the above algorithm (and all subsequent algorithms) for only $\frac{n}{4}$ iterations is sufficient in order to factor N. For simplicity, we omit this optimisation in the above and all subsequent descriptions of algorithms.

Since w.l.o.g. $p < \sqrt{N}$, this algorithm does not even improve over a trivial enumeration of all possible prime factors. Nevertheless, it lays the foundation for the HS erasure correction algorithm: In the HS scenario, the attacker is given a certain fraction of bits of the correct factorisation (in random positions). This auxiliary information can directly be used to exclude (large) subtrees with inconsistent roots from the computation tree. As long as a sufficiently large number of bits is known, this will result in a search tree whose total number of leaves can be bounded by a polynomial. By a straightforward heuristic analysis based on estimating the first two moments of the number of candidate solutions examined by the algorithm, Heninger and Shacham proved the following theorem.

Theorem 9.1.4 (HS Erasure Correction, cf. [HS09, Theorem1]). *Given a $\delta = 0.59$ fraction of the bits of p and q, one can (heuristically) recover the factorisation of an n-bit RSA modulus N in time $\mathcal{O}(n^2)$ with probability $1 - \frac{1}{\Omega(n^2)}$.*

We point out that this theorem is stated ambiguously; The proof implicitly assumes that the algorithm knows every single bit of p and q independently with probability δ (which can be different from knowing a δ fraction of the bits). Note that the HS algorithm itself is *deterministic*. The probability in the above theorem is defined over the random choice of the prime factors p and q and the erasure positions. In particular, the algorithm will fail in recovering the factorisation within the $\mathcal{O}(n^2)$ time-bound for a vanishing fraction of inputs. We would like to stress that the HS algorithm will *never* discard the correct solution from the computation tree, since the correct partial solution will always agree with the given incomplete key material.

The analysis of our algorithm shares some similarities with the analysis of Heninger and Shacham, in particular it is based on the same heuristic assumption, and it is thus instructive to take a closer look at it. The overall running time of the HS algorithm is dominated by the total number of *bad* partial solutions that have to be processed. These bad partial solutions can be of the following two different types:

- Either a bad partial solution originates from lifting the correct partial solution to the next two bits of p and q
- *or* it originates from lifting a bad partial solution.

Let us first consider the case when the good partial solution is lifted: Denote $Z_g \in \{0, 1\}$ the number of bad solutions that arise from lifting the good solution at step i of the algorithm: Since the lifting equation (9.3) has a *unique* solution as soon as one of the bits p_i or q_i is known, a bad partial solution can only be generated when *both* positions are erased, i.e. $\mathbb{E}[Z_g] = (1 - \delta)^2$.

For the second case, a similar analysis for the random variable $W_b \in \{0, 1, 2\}$ (that counts the number of bad solutions generated from a single bad solution at step i) can be done based on the following heuristic assumption (cf. [HS09, Conj.1]).

Heuristic 9.1.5. For random p, q and for p', q' with $f_N(p', q') = 0 \mod 2^i$ with $p_j \neq p'_j$ or $q_j \neq q'_j$ for at least one $j = 0, \ldots, i-1$ it holds

$$\mathbf{Pr}[(N - p'q')_i = (N - pq)_i] = \frac{1}{2} \ . \tag{9.4}$$

That is, the right hand side of the lifting equation of a bad partial solution is equal to the right hand side of the lifting equation of the correct solution with probability $\frac{1}{2}$. Under this assumption it is easy to estimate $\mathbb{E}[W_b]$ at step i as follows: If both bits p_i

and q_i are erased, a bad partial solution generates two additional bad partial solutions, i.e. $\mathbf{Pr}[W_b = 2] = (1-\delta)^2$. If either p_i or q_i is known, the lifting equation provides a unique solution and thus generates a single bad partial solution. If both p_i and q_i are known, the lifting equation is fulfilled iff $(N - p'q')_i = p_i + q_i = (N - pq)_i$ which happens with probability $\frac{1}{2}$, i.e. $\mathbf{Pr}[W_b = 1] = 2\delta(1-\delta) + \frac{\delta^2}{2}$. This finally gives

$$\mathbb{E}[W_b] = \frac{(2-\delta)^2}{2} \tag{9.5}$$

where the expectation is over the randomness of p, q and the erasure positions. Using similar arguments, Heninger and Shacham also give formulas for $\mathbb{E}\left[Z_g^2\right]$ and $\mathbb{E}\left[W_b^2\right]$ which allows to compute the respective variances.

Once Z_g and W_b are known, it is possible to compute the first two moments of another random variable X_i that counts the overall number of bad partial solutions that occur at step i of the algorithm due to the following important theorems.

Theorem 9.1.6 (cf. [HS09, Theorem2]). *It holds*

$$\mathbb{E}[X_i] = \frac{\mathbb{E}[Z_g]}{1 - \mathbb{E}[W_b]}\left(1 - \mathbb{E}[W_b]^i\right) \quad . \tag{9.6}$$

Theorem 9.1.7 (cf. [HS09, Theorem3]). *It holds*

$$\mathrm{Var}[X_i] = \alpha_1 + \alpha_2 \mathbb{E}[W_b]^i + \alpha_3 \mathbb{E}[W_b]^{2i} \tag{9.7}$$

with

$$\alpha_1 = \frac{\mathbb{E}[Z_g]\,\mathrm{Var}[W_b] + (1 - \mathbb{E}[W_b])\,\mathrm{Var}[Z_g]}{(1 - \mathbb{E}[W_b]^2)(1 - \mathbb{E}[W_b])} \quad , \tag{9.8}$$

$$\alpha_2 = \frac{\mathbb{E}\left[W_b^2\right] + \mathbb{E}[W_b] - 2\mathbb{E}[W_b]\,\mathbb{E}[Z_g] - \mathbb{E}[Z_g]}{1 - \mathbb{E}[W_b]} + 2\left(\frac{\mathbb{E}[Z_g]}{1 - \mathbb{E}[W_b]}\right)^2 \quad , \tag{9.9}$$

$$\alpha_3 = -(\alpha_1 + \alpha_2) \quad . \tag{9.10}$$

Now define $X := \sum_{i=1}^{n/2-1} X_i$ as the total number of bad partial solutions that occur in all iterations. According to Eq.(9.16) it holds $\mathbb{E}[W_b] < 1$ if $\delta > 2 - \sqrt{2} > 0.59$ (which gives the lower bound on δ or equivalently the upper bound $1 - \delta < 0.41$ on the erasure probability). By the above theorems, this implies that $\mathbb{E}[X_i]$ in Eq.(9.6) and $\mathrm{Var}[X_i]$ in Eq.(9.7) can be upper bounded by two constants dependent on δ but not on i. We thus upper bound the expected total number of bad partial solutions by $\mathbb{E}[X] = \mathcal{O}(n)$ and also $\mathrm{Var}[X] = \mathcal{O}(n^2)$ by Lemma 2.2.1. By Chebychev's inequality it eventually follows

$$\mathbf{Pr}\left[X \geq \Theta(n^2)\right] \leq 1 - \frac{1}{\Omega(n^2)}$$

as stated in Theorem 9.1.4. We conclude with the following observation which is important for the analysis in the next section.

Remark 9.1.8. The proof of Theorem 9.1.6 and Theorem 9.1.7 (see AppendixA in [HS09]) is based on probability generating functions, i.e. the distribution of a discrete random variable X can be represented by its **generating function** $F(s) := \sum \mathbf{Pr}[X = k]\, s^k$ (e.g. $\frac{dF(1)}{ds} = \mathbb{E}[X]$ reveals the expectation of X). If $G_i(s)$ denotes the generating function for the above random variable X_i and $w(s)$ and $z(s)$ denote the respective generating functions for W_b and Z_g, the proof is solely based on the recurrence

$$G_{i+1}(s) = G_i(w(s))z(s) \ . \tag{9.11}$$

This results from the fact that the number of bad solutions at step $i+1$ equals the number of bad solutions lifted from bad solutions at step i *plus* the number of bad solutions lifted from the good solution at step i (the generating function for the sum of two independent random variables is given by the convolution of their generating functions). Moreover, it uses $G_0(s) = 1$ (which holds since there is *no* bad solution in the initial step of the algorithm).

The analysis of our algorithm uses similar random variables Z_g, W_b and X_i sharing the same recurrence relation for their respective generating functions (although the actual definition of Z_g and W_b will be slightly different in our analysis). Consequently, we can apply Theorem 9.1.6 and Theorem 9.1.7 in our setting (but computing and bounding the first two moments of Z_g and W_b will be more complicated).

9.2 A Noisy Integer Factorisation Algorithm

Clearly, the Heninger-Shacham comparison of partial solutions with the given fault-free information about p and q cannot naively been transferred to the error correction scenario. The reason is that a disagreement of a partial solution may originate from a bad partial solution *or* from faulty bits in the noisy input $\tilde{p},\tilde{q}$. Thus, in our construction we do no longer compare bit by bit but we compare larger blocks of bits. More precisely, we run t successive liftings without eliminating any solution, i.e. we grow subtrees of depth t for each partial solution p', q'. For every partial solution p', q' in step i, this results in 2^t new partial solutions for step $i+1$ which we all compare with our noisy input $\tilde{p},\tilde{q}$. More precisely, every new partial solution p', q' (corresponding to a leaf in the subtree) consists of $2t$ fresh bits (t fresh bits for p' and q') and we compare those $2t$ bits with the respective bits in the noisy input $\tilde{p}, \tilde{q}$. If the bit agreement in these $2t$ bits is above some threshold parameter C we keep the candidate, otherwise we discard it. Put simply, the i-th iteration of our algorithm consists of an **expansion phase** (that generates 2^t new partial solutions from every partial solution that survived the preceding iteration) combined with a **pruning phase** (that eliminates all candidate solutions that differ from the noisy input by some threshold value C).

Clearly, we obtain a trade-off for the choice t of the depth of our subtrees. On the one hand, t cannot be chosen too large since in each iteration we grow our search tree by

at least 2^t candidates. Thus, t must be bounded by $\mathcal{O}(\log\log N)$ in order to guarantee a polynomial size of the search tree. On the other hand, depending on the error rate, t has to be chosen sufficiently large to guarantee that the correct partial solution has large agreement with our noisy input in each $2t$ successive bit positions, such that the correct candidate will never be discarded during the execution of our algorithm. Moreover, t has to be large enough such that the distribution corresponding to the correct candidate and the distribution derived from an incorrect candidate are separable by some threshold parameter C. If this property does not hold, we obtain too many faulty candidates for the next iteration.

We show that the above trade-off restrictions can be fulfilled whenever we have an error rate $\delta < 0.0837 - \epsilon$ for some fixed $\epsilon > 0$. That is, if we choose t of size polynomial in $\log\log N$ and $\frac{1}{\epsilon}$, we are able to define a threshold parameter C such that the following holds.

1. With probability close to 1 the correct factorisation will never be discarded during the execution of the algorithm.

2. For fixed $\epsilon > 0$, our algorithm will consider no more than a total number of $\mathrm{poly}(\log N)$ key candidates (with high probability over the random choice of p and q and the error distribution), giving our our algorithm a running time polynomial in the bit-size of N (with high probability).

We would like to point that one needs to know the error δ for a proper choice of the parameters t and C. In practical side-channel attacks, where δ might be unknown to an attacker, one can apply an additional search step which successively increases the value of δ until a solution is found. Moreover, our algorithm comes with a one-sided error. Whenever it outputs a solution the output is the correct factorisation p, q. The description of the algorithm is elementary and the main work that has to be done is to carefully choose the subtree depth t and the threshold parameter C, such that all trade-off restrictions hold. We achieve this goal by using a statistical analysis via Hoeffding bounds, cf. Theorem 2.2.4.

Some Remarks on the Heuristic Assumption. Our analysis is based on the same heuristic assumption (Assumption 9.1.5) as introduced and experimentally verified by Heninger and Shacham in [HS09]. In our setting, this assumption implies that a single path in a subtree that is generated from a bad partial solution consists of $2t$ bits where every single bit matches the respective bit of the correct factorisation independently with probability $\frac{1}{2}$. In particular, the $2t$ bits of a fixed path in such a subtree will match the respective bits in $\tilde{p}, \tilde{q}$ independently with probability $\frac{1}{2}$ as well. This property is sufficient in order to apply Hoeffing's inequality to upper bound a particular random variable W_b^i counting the number of matching bits of the i-th leaf in a subtree generated from a bad partial solution with the noisy bits in $\tilde{p}$ and $\tilde{q}$, see Section 9.2.1 for details. We would like to stress that we do not need to assume *independence* of the random variables W_b^i (which is clearly not the case: every neighboured leaves in the same subtree share all but the last two bits). As in the HS proof, we merely need to upper bound $\mathbb{E}[W_b] < 1$

which can be done by linearity of expectation (and does not require independent W_b^i). The dependencies amongst the W_b^i only complicate the study of $\mathrm{Var}[W_b]$ which can still be appropriately upper bounded for carefully chosen parameters t and C.

Algorithm Description

In every iteration, the error correction algorithm recovers t more bits of p and q. Thus, repeating the expansion and pruning phase $\lceil \frac{n/2-1}{t} \rceil$ times allows to recover p, q. In the sequel, we write $\tau(n,t) := \lceil \frac{n/2-1}{t} \rceil$ for ease of presentation.

Algorithm 9: NOISY-FACTORING

input : N, erroneous $\tilde{p}$ and $\tilde{q}$, error rate δ.
output: p, q or failure symbol $\perp$.

Initialisation:
- Compute parameters $t(\delta)$ and $C(\delta)$, cf. Theorem 9.2.1
- $\mathcal{L}_0 := \{(1,1)\}$

For $i = 1$ **to** $\tau(n,t)$

Expansion phase: For every partial solution $(p', q') \in \mathcal{L}_{i-1}$, i.e. p' and q' are candidates solutions for the first $(i-1)t+1$ bits, compute a subtree $T(p', q')$ by t successive liftings, i.e. each of the 2^t leaves in $T(p', q')$ is a partial solution for the first $it+1$ bits.

Pruning phase: For every subtree T and all candidates $(p', q') \in T$ count the number of matchings between the $2t$ bits $it, \ldots, (i-1)t+1$ of p', q' and $\tilde{p}, \tilde{q}$. If this number is below some threshold parameter C, discard the solution. Otherwise add (p', q') to list $\mathcal{L}_i$.

Finalisation phase: Test every $(p', q') \in \mathcal{L}_{\tau(n,t)}$, i.e. output p', q' if $N = p'q'$.

In the subsequent section, we will analyse the probability that our algorithm succeeds in computing the factorisation p, q. We will show that a choice of $t = \Theta(\frac{\ln n}{\epsilon^2})$ will be sufficient for error rates $\delta < 0.0837 - \epsilon$. The threshold parameter C will be chosen such that the correct partial solution will survive each pruning phase with probability close to 1 and such that we expect that the number of candidates per iteration is bounded by 2^{t+1}. For every fixed $\epsilon > 0$, this leads to a running time that is polynomial in n (with high probability).

9.2.1 Choice of Parameters and Runtime Analysis

We now give a detailed analysis for algorithm Noisy-Factoring from the previous section. Remember that in every iteration, we count the number of matching bits between $2t$-bit blocks of $\tilde{p}, \tilde{q}$ and every partial candidate solution p', q'. Let us define a random variable M_c for the number of matching bits between $\tilde{p}, \tilde{q}$ and the correct partial solution. The distribution of M_c is clearly the binomial distribution with parameters $2t$ and probability $(1-\delta)$, denoted by $M_c \sim \mathrm{Bin}(2t, 1-\delta)$. The expected number of matches is thus $\mathbb{E}[M_c] = 2t(1-\delta)$.

Similarly, let M_b be a random variable representing the number of matching bits of $\tilde{p}, \tilde{q}$ with a partial solution that was generated by expanding a *bad* partial solution. As explained above, under Heuristic 9.1.5, it holds $M_b \sim \mathrm{Bin}(2t, \frac{1}{2})$ and thus $\mathbb{E}[M_b] = t$. Now, we basically have to choose our threshold C such that the two distributions are sufficiently separated. The remainder of this section is devoted to the proof of our main result.

Main Theorem 9.2.1. *Under Heuristic 9.1.5 for every fixed $\epsilon > 0$ the following holds. Let $N = pq$ be a n-bit RSA modulus with balanced p and q. We choose*

$$t = \left\lceil \frac{\ln n}{4\epsilon^2} \right\rceil,\ \gamma_0 = \sqrt{(1+\tfrac{1}{t}) \cdot \tfrac{\ln 2}{4}} \text{ and } C = 2t(\tfrac{1}{2} + \gamma_0).$$

Further, let $\tilde{p}, \tilde{q}$ be an erroneous copy of p, q with noise rate

$$\delta \leq \frac{1}{2} - \gamma_0 - \epsilon.$$

Then algorithm Noisy-Factoring *reveals p, q in time $\mathcal{O}\left(n^{3+\frac{\ln 2}{2\epsilon^2}}\right)$ with success probability at least $1 - \left(\frac{2\epsilon^2}{\ln n} + \frac{1}{n} + \frac{1}{\Omega(n^2)}\right)$.*

Remark 9.2.1. Notice that for sufficiently large n, t converges to infinity and thus γ_0 converges to $\sqrt{\ln(2)/4} > 0.4163$. This means that Noisy-Factoring asymptotically allows for error rates $\frac{1}{2} - \gamma_0 - \epsilon < 0.0837 - \epsilon$ and succeeds with probability close to 1.

Proof of Main Theorem 9.2.1. The proof is organised as follows:

1. Upper bound the total number X of bad solutions examined by the algorithm (with high probability).
2. Upper bound the probability of eliminating the correct solution.

Let us start with the simpler second part and recall that $\tau(n,t) := \lceil \frac{n/2-1}{t} \rceil$.

Lemma 9.2.2. *The correct solution is eliminated with probability at most*

$$\frac{2\epsilon^2}{\ln n} + \frac{1}{n} \ . \tag{9.12}$$

Proof. The probability of pruning the correct solution at one single round is given by $\mathbf{Pr}[M_c < C]$, where $M_c \sim \mathrm{Bin}(2t, 1-\delta)$. Using $\frac{1}{2} + \gamma_0 \leq 1 - \delta - \epsilon$ and applying Hoeffding's bound (Theorem 2.2.4) yields

$$\mathbf{Pr}[M_c < C] = \mathbf{Pr}\left[M_c < 2t\left(\frac{1}{2} + \gamma_0\right)\right] \leq \mathbf{Pr}[M_c < 2t(1-\delta-\epsilon)] \leq \exp(-4t\epsilon^2) \leq \frac{1}{n}.$$

Now, let $E_i \in \{0,1\}$ be a random variable with $E_i = 1$ iff the correct solution is eliminated in the i-th iteration, i.e. $\mathbf{Pr}[E_i = 1] \leq \frac{1}{n}$. Clearly, the algorithm fails in finding the correct solution iff $E := \sum_{i=1}^{\tau(n,t)} E_i \geq 1$. By applying the union bound and using $\tau(n,t) \leq \frac{n}{2t} + 1$ we finally obtain

$$\mathbf{Pr}[E \geq 1] = \mathbf{Pr}\left[\bigvee_{i=1}^{\tau(n,t)} \{E_i = 1\}\right] \leq \frac{\tau(n,t)}{n} \leq \frac{1}{2t} + \frac{1}{n} \leq \frac{2\epsilon^2}{\ln n} + \frac{1}{n} \tag{9.13}$$

by definition of t. □

Let us no proceed with the first part of the proof of our main theorem. Similarly to the HS analysis we define the following random variables.

- $Z_g \in \{0, \ldots, 2^t - 1\}$ counts the number of bad partial solutions generated from the good partial solution in one iteration.
- $W_b \in \{0, \ldots, 2^t\}$ counts the number of bad partial solutions generated from a single bad partial solution in one iteration.
- X_i counts the overall number of bad partial solutions that occur in the i-th iteration.

Our final goal is to upper bound the random variable $X := \sum_{i=1}^{\tau(n,t)} X_i$ counting the total number of bad solutions examined by the algorithm. Obviously, we have the following rough upper bounds

$$\mathbb{E}[Z_g] < 2^t = \mathcal{O}(n^{\frac{\ln 2}{4\epsilon^2}}) \tag{9.14}$$

and

$$\mathrm{Var}[Z_g] \leq \mathbb{E}\left[Z_g^2\right] = \sum_{i=0}^{2^t-1} \mathbf{Pr}[Z_g = i]\, i^2 < 2^{2t} = \mathcal{O}(n^{\frac{\ln 2}{2\epsilon^2}}) \ . \tag{9.15}$$

By applying Hoeffding's inequality again, we obtain a similar statement for W_b.

Lemma 9.2.3. *It holds*

$$\mathbb{E}[W_b] \leq \frac{1}{2} \tag{9.16}$$

and

$$\mathrm{Var}[W_b] < 2^t = \mathcal{O}(n^{\frac{\ln 2}{4\epsilon^2}}) \ . \tag{9.17}$$

Proof. Write $W_b := \sum_{i=1}^{2^t} W_b^i$ where

$$W_b^i = \begin{cases} 1 & i\text{-th bad candidate passes,} \\ 0 & \text{otherwise.} \end{cases}$$

Since all the W_b^i are identically distributed, we can simplify $\mathbb{E}[W_b] = 2^t \mathbb{E}\left[W_b^i\right]$ and upper bound $\mathbb{E}\left[W_b^i\right]$ for some fixed i. Note, that $W_b^i = 1$ iff at least C out of $2t$ bits of the i-th bad partial solution match the corresponding bits of $\tilde{p}, \tilde{q}$, i.e.

$$\mathbb{E}\left[W_b^i\right] = \mathbf{Pr}\left[W_b^i = 1\right] = \mathbf{Pr}\left[M_b \geq C\right],$$

where $M_b \sim \mathrm{Bin}(2t, \frac{1}{2})$. Applying Hoeffding's bound (Theorem 2.2.4) directly yields

$$\begin{aligned} \mathbf{Pr}\left[M_b \geq C\right] &= \mathbf{Pr}\left[M_b \geq 2t\left(\frac{1}{2} + \gamma_0\right)\right] \\ &\leq \exp(-4t\gamma_0^2) = 2^{-(1+\frac{1}{t})t} \leq 2^{-(t+1)} \end{aligned}$$

which implies $\mathbb{E}[W_b] \leq \frac{1}{2}$. By Lemma 2.2.1 we also obtain

$$\mathrm{Var}[W_b] = \mathrm{Var}\left[\sum_{i=1}^{2^t} W_b^i\right] \leq 2^{2t} \max \mathrm{Var}\left[W_b^i\right] \leq 2^{2t}\mathbb{E}\left[W_b^i\right] < 2^t$$

where the second last inequality holds since W_b^i is a $\{0,1\}$-variable, i.e. $\mathbb{E}\left[(W_b^i)^2\right] = \mathbb{E}\left[W_b^i\right]$. □

The next lemma upper bounds the total number X of bad partial solutions using the preceding results.

Lemma 9.2.4. *It holds*

$$\mathbf{Pr}\left[X \geq \Theta(n^{2+\frac{\ln 2}{4\epsilon^2}})\right] \leq \frac{1}{\Omega(n^2)} \ . \tag{9.18}$$

Proof. Recall $X = \sum_{i=1}^{\tau(n,t)} X_i$. Due to Lemma 9.2.3 we have $\mathbb{E}[W_b] \leq \frac{1}{2}$. Combing this with Eq.(9.14) and Theorem 9.1.6 gives

$$\mathbb{E}[X_i] < \frac{\mathbb{E}[Z_g]}{1 - \mathbb{E}[W_b]} < 2^{t+1} = \mathcal{O}(n^{\frac{\ln 2}{4\epsilon^2}})$$

and thus

$$\mathbb{E}[X] = \mathcal{O}(n^{1+\frac{\ln 2}{4\epsilon^2}})$$

since $\tau(n,t) \leq n$. Moreover, $\mathbb{E}[W_b] < 1$ and Theorem 9.1.7 yields

$$\mathrm{Var}[X_i] = |\mathrm{Var}[X_i]| \leq |\alpha_1| + |\alpha_2| + |\alpha_3|$$

where the α_i are defined by Eq.(9.8)-(9.10). The dominating terms in α_1 are $\mathrm{Var}[Z_g]$ and $\mathbb{E}[Z_g] \cdot \mathrm{Var}[W_b]$ and and by Eq.(9.17), (9.14) and (9.15) it holds

$$|\alpha_1| = \mathcal{O}(n^{\frac{\ln 2}{2\epsilon^2}}) \ .$$

Since $\mathbb{E}\left[W_b^2\right] = \mathrm{Var}[W_b] + \mathbb{E}[W_b]^2 < 2^t + \frac{1}{4} = \mathcal{O}(n^{\frac{\ln 2}{4\epsilon^2}})$, the dominating term in α_2 is $\mathbb{E}[Z_g]^2$ and thus

$$|\alpha_2| = \mathcal{O}(n^{\frac{\ln 2}{2\epsilon^2}}) \ .$$

This directly implies $|\alpha_3| = \mathcal{O}(n^{\frac{\ln 2}{2\epsilon^2}})$ and we finally obtain

$$\mathrm{Var}[X_i] = \mathcal{O}(n^{\frac{\ln 2}{2\epsilon^2}}) \ . \tag{9.19}$$

Now, we can apply Chebychev's inequality (Theorem 2.2.3) which yields

$$\mathbf{Pr}[|X - \mathbb{E}[X]| \geq \alpha n] \leq \frac{\mathrm{Var}[X]}{(\alpha n)^2} \leq \frac{\max \mathrm{Var}[X_i]}{\alpha^2} = \frac{1}{\Omega(n^2)} \tag{9.20}$$

for $\alpha = \Theta(n^{1+\frac{\ln 2}{4\epsilon^2}})$. □

To finish the proof of our main theorem, we first define the following failure event: We say that Noisy-Factoring *fails* if the correct solution is eliminated *or* the actual total number of bad candidate solutions is $\omega(n^{2+\frac{\ln 2}{4\epsilon^2}})$ (we artificially abort the computation if the computation tree exceeds a certain bound and output the failure symbol $\perp$). By Lemma 9.2.2 and Lemma 9.2.4 it holds

$$\mathbf{Pr}[\text{Noisy-Factoring fails}] \leq \frac{2\epsilon^2}{\ln n} + \frac{1}{n} + \frac{1}{\Omega(n^2)} \ .$$

To estimate the running time of the algorithm, we can safely ignore the costs of the initialisation. We now upper bound the runtime needed by the expansion and pruning phase for one single partial solution:

- During the expansion phase, each partial solution implies the computation of $\sum_{i=0}^{t-1} 2^i < 2^t$ liftings according to Eq.(9.3). As already mentioned in Section 9.1, the right hand side of of Eq.(9.3) can be computed in time $\mathcal{O}(n)$ – when storing the results of the previous iterations. This yields a total computation time of $\mathcal{O}(n2^t)$ for the expanding phase.

- The pruning phase can be realised in time $\mathcal{O}(t)$ for each of the fresh 2^t partial solutions, summing up to $\mathcal{O}(t2^t)$.

We can upper bound $t \leq n$, which results in an overall runtime of

$$\mathcal{O}((n+t)\cdot 2^t) = \mathcal{O}(n2^t) = \mathcal{O}(n^{1+\frac{\ln 2}{4\epsilon^2}})$$

per candidate solution. Finally, testing candidate solutions for correctness can also be done in time $\mathcal{O}(n)$.

Since we abort the computation whenever the size of our computation tree is $\omega(n^{2+\frac{\ln 2}{4\epsilon^2}})$ (note that there will only be $\mathcal{O}(n)$ good solutions in any case), the total workload can be upper bounded by

$$\mathcal{O}\left(n^{2+\frac{\ln 2}{4\epsilon^2}} n^{1+\frac{\ln 2}{4\epsilon^2}}\right) = \mathcal{O}\left(n^{3+\frac{\ln 2}{2\epsilon^2}}\right)$$

as claimed. This finishes the proof of Theorem 9.2.1.

□

9.3 The Coding-Theoretic Link

Very recently Paterson et al. [PPS12] further extended the noisy factorisation problem (or, more precisely, the more general problem of correcting errors in fully redundant RSA secret keys) to the setting where the 0 and 1 bits of p and q are flipped with possibly distinct error probabilities δ_1 and δ_2, respectively. Moreover, [PPS12] introduces a neat coding-theoretic viewpoint on the problem. This viewpoint allows to use interesting tools from coding and information theory, such as *channel capacity* and *list decoding*, to develop a new (slightly different) algorithm for their more general framework. Beyond that, Paterson et al. propose upper bounds on the performance of their algorithm which also apply to the HS algorithm and our work. In this section, we

- present the coding-theoretic viewpoint of [PPS12],
- discuss a (heuristic) upper bound on the error rate δ for a class of algorithms (that contains our Noisy-Factoring algorithm) which is related to the *Shannon capacity* of the underlying binary symmetric channel and
- prove that our algorithm and its analysis allow for an error rate that achieves the second-order expansion of the theoretical upper bound (as first observed in [KSI13]).

The Coding-theoretic Viewpoint

Let us begin with a brief description of the coding-theoretic link of [PPS12]: Consider the naïve dyadic factorisation algorithm from Section 9.1 and recall that the $2^{\frac{n}{2}}$ leaves of the resulting computation tree give a set of candidate factorisations (each containing n

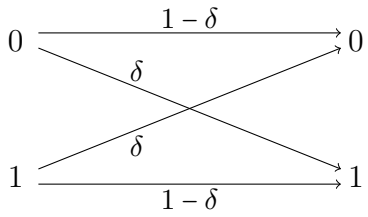


Figure 9.1: The BSC with cross-over probability δ: A single bit is flipped with probability $0 < \delta < \frac{1}{2}$.

bits). This candidate set depends on the modulus N and is denoted by $\mathcal{C}_N := \{\mathsf{c}_1, \ldots, \mathsf{c}_\lambda\}$ where we write $\lambda := 2^{\frac{n}{2}}$ for short. Now, Paterson et al. simply propose to view the set $\mathcal{C}_N$ as a *block code* of length n and rate $\log|\mathcal{C}_N|/n = \frac{1}{2}$. In contrary to all preceding chapters in this thesis, this code is non-linear, non-random (due to the iterative nature of the Hensel lifting) and offers bad general decoding capabilities (the minimum distance of the code is 2: every two leaves with the same ancestor agree on all but the last two bits).

Now, Paterson et al. argue that the noisy factoring problem can be reformulated as a decoding problem in $\mathcal{C}_N$ as follows (cf.[PPS12, Section3]): Let $\mathsf{c}_i^* \in \mathcal{C}_N$ be the codeword that belongs to the wanted factorisation p, q. This codeword is simply transmitted over the binary symmetric channel (BSC) with crossover probability δ (see Figure 9.1) yielding a received word r. The task is to decode r to c_i^*. In this light, the NOISY-FACTORING algorithm can be seen as a *list decoding algorithm* for $\mathcal{C}_N$. The notion of list decoding goes back to Elias [Eli57] and refers to decoding algorithms that output a list of candidate codewords instead of a single, unique codeword. For such a list decoding algorithm D, a decoding error occurs if a transmitted codeword c results in a received word r such that the output list $D(\mathsf{r})$ does not contain c. As claimed in [PPS12], "this coding-theoretic viewpoint allows to derive limits on the performance of *any* procedure for selecting which candidate solution to keep in the HMM algorithm". In the next section, we explain this argument in more detail and discuss some technical caveats.

Theoretical Upper Bound on δ

A well-known result in information theory is the converse of Shannon's noisy-channel coding theorem [Sha48]: Asymptotically, no combination of a code and decoding algorithm can achieve reliable decoding when the code rate exceeds the *capacity* of the underlying communication channel. Although Shannon's original statement only considers the classical framework where one decodes to a unique codeword, it is easy to obtain a generalisation for list decoding algorithms with output lists of polynomially bounded size. More precisely, the argument in [PPS12] is based on the following simple statement.

Theorem 9.3.1 (List-Decoding Capacity, cf. [Gur06, Theorem 3.4])**.** *Let* $0 < \delta < \frac{1}{2}$ *and*

$\epsilon > 0$. *Then for large enough* n *and* any *code* $\mathcal{C}$ *of block length* n *and rate* $R := \frac{\log|\mathcal{C}|}{n} \geq 1 - \mathrm{H}(\delta) + \epsilon$, *there is at least one (received) word* $\mathsf{r} \in \{0,1\}^n$ *such that*

$$|\mathcal{B}(\mathsf{r}, \delta n) \cap \mathcal{C}| \geq 2^{\frac{\epsilon}{2}n} \ ,$$

that is at least one received word r *can not be list-decoded with polynomial sized lists.*

Remark 9.3.2. Note that the term $1 - \mathrm{H}(\delta)$ resembles the capacity of the BSC with crossover probability δ. Moreover, the following proof also shows, that the above statement holds in the average case, i.e. the expected number of codewords in a Hamming ball of radius δn around a received word r is $2^{\epsilon n - \mathrm{o}(n)}$ where the probability is defined over the random choice of r (and fixed code $\mathcal{C}$).

Proof of Theorem 9.3.1. Define a random variable $Z := |\mathcal{B}(\mathsf{r}, \delta n) \cap \mathcal{C}|$ (over the random choice of r). Clearly,

$$\mathbb{E}[Z] = |\mathcal{C}| \frac{|\mathcal{B}(\mathsf{r}, \delta n)|}{2^n} \geq 2^{\epsilon n - \mathrm{o}(n)} \ ,$$

thus there must exist at least one r with $|\mathcal{B}(\mathsf{r}, \delta n) \cap \mathcal{C}| \geq 2^{\epsilon n - \mathrm{o}(n)} \geq 2^{\frac{\epsilon}{2}n}$. □

Based on Theorem 9.3.1, Paterson et al. argue that no list-decoding algorithm with output lists of polynomial size for the noisy integer factorisation problem can exist whenever

$$\delta > \mathrm{H}^{-1}(\frac{1}{2}) > 0.111 \ . \tag{9.21}$$

This can be seen as follows: First, the Hamming distance between the received word r and the codeword c_i^* is at least $(\delta - \epsilon)n$ with probability close to 1 for sufficiently large n and arbitrary $\epsilon > 0$. Thus, a list decoding algorithm must ouput a list of all codewords within Hamming distance δn of r. Consequently, the above theorem tells us that such an output list will have exponential-size if the code rate ($= \frac{1}{2}$) exceeds the capacity of the underlying BSC ($= 1 - \mathrm{H}(\delta)$) which yields Eq.(9.21).

However, the above argumentation suffers from the following technical caveats:

1. We have to assume that the received word r is chosen uniformly at random in $\{0,1\}^n$, in particular its choice must be *independent* from the code $\mathcal{C}_N$ (this assumption seems questionable since the transmitted word c_i^* is clearly *not* independent from $\mathcal{C}_N$).

2. The proof of Theorem 9.3.1 only provides an average-case lower bound on the random variable $Z = |\mathcal{B}(\mathsf{r}, \delta n) \cap \mathcal{C}|$. However, depending on the actual code $\mathcal{C}$, strong deviations are conceivable: For example, consider the code $\mathcal{C} := \{\mathsf{c}_0, \ldots, \mathsf{c}_{2^{n/2}-1}\}$ of length n and rate $\frac{1}{2}$ where $\mathsf{c}_0 = (0, \ldots, 0, 1 \ldots, 1)$ and $\mathsf{c}_k = (k_0, \ldots, k_{n/2-1}, 0 \ldots, 0)$ (with the first $n/2$ coordinates of the codewords $\mathsf{c}_k, k \geq 1$ being the binary representation of k). Obviously, $\mathrm{d}(\mathsf{c}_i, \mathsf{r}) \geq \frac{n}{2}$ for every $\mathsf{r} \in \{0,1\}^{n/2} \times \{1\}^{n/2}$ and all $i \geq 1$. Thus, for exponentially many r we have $|\mathcal{B}(\mathsf{r}, \frac{n}{2} - 1) \cap \mathcal{C}| = 1$.

Clearly, the second point is very artificial and will probably not occur for the codes $\mathcal{C}_N$. However, the above observations are related to the fact that we do not need to design a list decoding algorithm for $\mathcal{C}_N$ that successfully decodes on the average: We only need to decode the particular codeword c_i^* with high probability (if a classical factorisation algorithm existed, such a decoding algorithm would be trivial to define). Consequently, the proposed upper bound on δ has to be considered *heuristic*. Obtaining a rigorous upper bound on δ would clearly require a proper formal definition of a specific class of algorithms (which is left as an open question).

In the rest of this chapter, let us now assume the validity of the upper bound (9.21). Recall that the theoretical upper bound of Noisy-Factoring provided by our analysis is

$$\delta \leq \frac{1}{2} - \sqrt{\frac{\ln 2}{4}} < 0.0837 \ , \tag{9.22}$$

thus there might be space for minor improvements. In fact, the recent algorithm of Paterson et al. *tightly* matches the theoretical upper bound but its analysis is based on stronger heuristic randomness assumptions. Furthermore, the next section shows that the upper bound (9.22) of our algorithm matches the second-order expansion of the heuristic upper bound (9.21).

Second-order Optimality of Noisy-Factoring

It is well-known that the Taylor series expansion of the binary entropy function in a neighbourhood of $\frac{1}{2}$ is given by

$$\mathrm{H}(\delta) = 1 - \frac{1}{2\ln 2} \sum_{i=1}^{\infty} \frac{(1-2\delta)^{2i}}{i(2i-1)} \ .$$

Let

$$\Delta := \left\{ 0 < \delta < \frac{1}{2} : 1 - \mathrm{H}(\delta) \geq \frac{1}{2} \right\} = \left\{ 0 < \delta < \frac{1}{2} : \sum_{i=1}^{\infty} \frac{(1-2\delta)^{2i}}{i(2i-1)} \geq \ln 2 \right\}$$

be the set of error rates δ below the information-theoretic upper bound. By using the expansion of H up to order k one obtains

$$\Delta = \bigcup_{k=1}^{\infty} \Delta_k$$

where $\Delta_k := \left\{ 0 < \delta < \frac{1}{2} : \sum_{i=1}^{k} \frac{(1-2\delta)^{2i}}{i(2i-1)} \geq \ln 2 \right\}$. For $k = 1$ we obtain

$$\Delta_1 = \left\{ 0 < \delta < \frac{1}{2} : (1-2\delta)^2 \geq \ln 2 \right\} = \left\{ 0 < \delta < \frac{1}{2} : \delta \leq \frac{1}{2} - \sqrt{\frac{\ln 2}{4}} \right\}$$

which is the same condition as in Eq.(9.22). That is, our algorithm allows for error rates up to the second-order expansion of the theoretical upper bound.

9.4 Conclusion

We conclude by indicating how the extended analysis for Noisy-Factoring and the coding-theoretic link can be adapted to the following more general scenario: Given a public RSA key (N, e) and noisy bits of a full RSA secret key (p, q, d, d_p, d_q), the Hensel lift gives four lifting equation in five unknowns (the unknowns are the respective bits of p, q, d, d_p, d_q). Note that every partial solution at step i of the algorithm now consists of $it + 1$ bits for each of the five elements (p, q, d, d_p, d_q) and t iterative Hensel liftings yield a computation tree with 2^t leaves where each leaf consists of $5t$ fresh bits. More generally, one can also consider scenarios where one is given a different combination of the elements p, q, d, d_p and d_q. Altogether, when given noisy bits of $2 \leq m \leq 5$ of such elements , denoted by $\tilde{\mathsf{sk}} = (\tilde{\mathsf{sk}}_1, \ldots, \tilde{\mathsf{sk}}_m)$, one obtains the following variant of Theorem 9.2.1 for a generalised algorithm Error-Correction.

Theorem 9.4.1. *Under Heuristic 9.1.5 for every fixed $\epsilon > 0$ the following holds. Let (N, e) be an n-bit RSA public key with fixed e. We choose*

$$t = \left\lceil \tfrac{\ln n}{2m\epsilon^2} \right\rceil,\ \gamma_0 = \sqrt{(1 + \tfrac{1}{t}) \cdot \tfrac{\ln 2}{2m}} \text{ and } C = mt(\tfrac{1}{2} + \gamma_0).$$

Further, let $\tilde{\mathsf{sk}}$ be an erroneous copy of sk with noise rate $\delta \leq \frac{1}{2} - \gamma_0 - \epsilon$. Then algorithm Error-Correction *reveals sk in time $\mathcal{O}\left(n^{3+\frac{\ln 2}{m\epsilon^2}}\right)$ with success probability at least $1 - \left(\frac{m\epsilon^2}{\ln n} + \frac{1}{n} + \frac{1}{\Omega(n^2)}\right)$.*

We point out that the public exponent e must be fixed in order to compute an initial solution for the Hensel lifting, see [HMM10] for details. From a coding-theoretic viewpoint, the generalised scenario generates a code of rate $\frac{1}{m}$ and the theoretical upper bound on δ becomes

$$\delta \leq \mathrm{H}^{-1}(1 - \frac{1}{m}) \ .$$

Thus, the larger m the larger the theoretically achievable noise rate δ. To summarize, we present values for δ in comparison with the respective upper bound δ^* for different m in Table 9.1. Note that the gap between the error rate achieved by our algorithm and the upper bound goes to 0 with $m \to \infty$.

m	δ	δ^*
2	0.084	0.111
3	0.160	0.174
4	0.206	0.214
5	0.237	0.243

Table 9.1: Bounds on δ and theoretical limit δ^* for varying m.

Bibliography

[Sha48] C.E. Shannon, *"A Mathematical Theory of Communication"*, Bell System Technical Journal 27, 1948.

[Eli57] P. Elias, *"List Decoding for Noisy Channels"*, Research Laboratories of Electronics, Technical Report 335, MIT, 1957.

[Pra62] E. Prange, *"The use of information sets in decoding cyclic codes"*, IRE Transactions on Information Theory, Vol.8(5), 1962.

[McE78] R. McEliece, *"A Public-Key Cryptosystem Based On Algebraic Coding Theory"*, DSN progress report, 1978.

[Hwa79] T.Y. Hwang, *"Decoding linear block codes for minimizing word error rate"*, IEEE Transactions on Information Theory, 1979.

[LH85] L. Levitin and C.R.P. Hartmann, *"A new approach to the general minimum distance decoding problem: The zero-neighbors algorithm"*, IEEE Transactions on Information Theory, 1985.

[RS85] R. Rivest and A. Shamir, *"Efficient Factoring Based on Partial Information"*, Advances in Cryptology - EUROCRYPT '85, 1985.

[Nie86] H. Niederreiter, *"Knapsack-type cryptosystems and algebraic coding theory"*, Problems of Control and Information Theory, 1986.

[LB88] P.J. Lee and E.F. Brickell, *"An observation on the security of McEliece's public-key cryptosystem."*, Advances in Cryptology - EUROCRYPT '88, 1988.

[Leo88] J.S. Leon, *"A probabilistic algorithm for computing minimum weights of large error-correcting codes."*, IEEE Transactions on Information Theory, 1988.

[Dum89] I. Dumer, *"Two decoding algorithms for linear codes"*, Problems of Information Transmission, 1989.

[Ste89] J. Stern., *"A method for finding codewords of small weight."*, Third International Colloquium on Coding Theory and Applications, 1989.

[CG90] J.T. Coffey and R.M. Goodman, *"The Complexity of Information Set Decoding"*, IEEE Transactions on Information Theory, 1990.

[Dum91] I. Dumer, *"On minimum distance decoding of linear codes"*, Joint Soviet-Swedish International Workshop Information Theory, 1991.

[Mau92] , U. Maurer, *"Factoring with an Oracle"*, Advances in Cryptology - EUROCRYPT '92, 1992.

[SS92] V. Sidelnikov and S. Shestakov, *"On insecurity of cryptosystems based on generalized Reed-Solomon codes"*, Discrete Mathematics and Applications, 1992.

[BR93] M. Bellare and P. Rogaway, *"Random Oracles are Practical: A Paradigm for Designing Efficient Protocols"*, ACM Conference on Computer and Communications Security, 1993.

[Ste93] J. Stern, *"A New Identification Scheme Based on Syndrome Decoding"*, Advances in Cryptology - CRYPTO '93, 1993.

[Bau96] H. Bauer, *"Probability Theory"*, Walter de Gruyter, 1996.

[FS96] J. Fischer and J. Stern, *"An Efficient Pseudo-Random Generator Provably as Secure as Syndrome Decoding"*, Advances in Cryptology - EUROCRYPT '96, 1996.

[Cop97] D. Coppersmith, *"Small Solutions to Polynomial Equations, and Low Exponent RSA Vulnerabilities"*, Journal of Cryptology, Vol.10(4), 1997.

[Sho97] P. Shor, *"Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer"*, SIAM Journal of Computing Vol.26(5), 1997.

[AB98] A. Ashikhmin and A. Barg, *"Minimal Vectors in Linear Codes"*, IEEE Transactions on Information Theory, 1998.

[Bar98] A. Barg, *"Complexity Issues in Coding Theory"*, Chapter 7 in [PHB98]

[CC98] A.Canteaut and F.Chabaud, *"A new algorithm for finding minimum-weight words in a linear code: Application to McEliece's cryptosystem and to narrow-sense BCH codes of length 511."*, IEEE Transactions on Information Theory, 1998.

[PHB98] V.Pless, W.C. Huffman and R.Brualdi, *"Handbook of Coding Theory"*, Elsevier Science, 1998.

[vL98] J.H. van Lint, *"Introduction to Coding Theory (Third Edition)"*, Springer Graduate Texts in Mathematics, 1998.

[BKvT99] A. Barg, E. Krouk and H. van Tilborg, *"On the Complexity of Minimum Distance Decoding of Long Linear Codes"*, IEEE Transactions on Information Theory, 1999.

[HGS99] C. Hall, I. Goldberg and B. Schneier, *"Reaction Attacks against several Public-Key Cryptosystems"*, Information and Communication Security - ICICS'99, 1999.

[vOW99] P.C. van Oorschot and M.J. Wiener, *"Parallel Collision Search with Cryptanalytic Applications"*, Journal of Cryptology, Vol.12(1), 1999.

[Sen00] N. Sendrier, *"Finding the permutation between equivalent linear codes: The support splitting algorithm"*, IEEE Transactions on Information Theory, 2000.

[AlJ01] A. Kh. Al Jabri, *"A Statistical Decoding Algorithm for General Linear Block Codes"*, Cryptography and Coding, 2001.

[FS01] N. Courtois, M. Finiasz and N. Sendrier, *"How to Achieve a McEliece-Based Digital Signature Scheme"*, Advances in Cryptology - ASIACRYPT 2001, 2001.

[HB01] N.J Hopper and M. Blum, *"Secure Human Identification Protocols"*, in Advances in Cryptology - ASIACRYPT 2001, 2001.

[KI01] K. Kobara and H. Imai, *"Semantically Secure McEliece Public-Key Cryptosystems-Conversions for McEliece PKC"*, Public Key Cryptography - PKC 2001, 2001.

[NSS01] P.Q. Nguyen, I.E. Shparlinski and J.Stern, *"Distribution of modular sums and the security of the server aided exponentiation."*, Progress in Computer Science and Applied Logic, 2001.

[Wag02] D. Wagner, *"A Generalized Birthday Problem"*, Advances in Cryptology - CRYPTO 2002, 2002.

[PKCS#1] RSA Laboratories. PKCS #1 v2.1: RSA Cryptography Standard, June 2002.

[BKW03] A. Blum, A. Kalai and H. Wasserman, *"Noise-tolerant Learning, the Parity Problem, and the Statistical Query Model"*, in Journal of the ACM, Vol.50, 2003.

[May04] A. May, *"Computing the RSA Secret Key Is Deterministic Polynomial Time Equivalent to Factoring"*, Advances in Cryptology - CRYPTO 2004, 2004.

[JW05] A. Juels and S.A. Weis, *"Authenticating Pervasive Devices with Human Protocols"*, in Advances in Cryptology - CRYPTO 2005, 2005.

[Lyu05] V. Lyubashevski, *"The Parity Problem in the Presence of Noise, Decoding Random Linear Codes, and the Subset Sum Problem"*, in LNCS Approximation, Randomization and Combinatorial Optimization, Algorithms and Techniques - APPROX / InternationalWorkshop on Randomization and Computation - RANDOM, 2005.

[FMICM06] M.P.C. Fossorier, M.J. Mihaljevic, H. Imai, Y. Cui and K. Matsuura, *"An Algorithm for Solving the LPN Problem and Its Application to Security Evaluation of the HB Protocols for RFID Authentication"*, Progress in Cryptology - INDOCRYPT 2006, 2006.

[Gur06] V. Guruswami, *"Algorithmic Results in List Decoding"*, Foundations and Trends in Theoretical Computer Science, Vol.2(2), 2006.

[Ove06] R. Overbeck, *"Statistical Decoding Revisited"*, Information Security and Privacy - ACISP 2006, 2006.

[LF06] E. Levieil and P.A. Fouque, *"An Improved LPN Algorithm"*, in LNCS Security and Cryptography for Networks - SCN 2006, 2006.

[MS07] L. Minder and A. Shokrollahi, *"Cryptanalysis of the Sidelnikov Cryptosystem"*, Advances in Cryptology - EUROCRYPT 2007, 2007.

[AS08] N. Alon and J.H. Spencer, *"The Probabilistic Method"*, Wiley Series in Discrete Mathematics and Optimization, 2008.

[BLP08] D.J. Bernstein, T. Lange and C. Peters, *"Attacking and Defending the McEliece Cryptosystem"*, Post-Quantum Cryptography - PQCrypto 2008, 2008.

[MR08] D. Micciancio and O.Regev, *"Lattice-based cryptography"*, book chapter in *"Post-quantum cryptography"*, Springer, 2008.

[MV08] J. Matousek and J. Vondrak, *"The Probabilistic Method"*, Lecture Notes, Charles University Praha, `http://kam.mff.cuni.cz/~matousek/lectnotes.html`, 2008.

[NIKM08] R. Nojima, H. Imai, K. Kobara and K. Morozov, *"Semantic security for the McEliece cryptosystem without random oracles"*, Designs, Codes and Cryptography, 2008.

[DMQN09] R. Dowsley, J. Müller-Quade and A. Nascimento, *"A CCA2 Secure Public Key Encryption Scheme Based on the McEliece Assumptions in the Standard Model"*, Topics in Cryptology - CT-RSA 2009, 2009.

[FS09] M. Finiasz and N. Sendrier, *"Security Bounds for the Design of Code-Based Cryptosystems."*, Advances in Cryptology - ASIACRYPT 2009, 2009.

[HS09] N.Heninger and H.Shacham, *"Reconstructing RSA Private Keys from Random Key Bits"*, Advances in Cryptology - CRYPTO 2009, 2009.

[BLP10] D.J. Bernstein, T. Lange and C. Peters, *"Wild McEliece"*, Selected Areas in Cryptography - SAC 2010, 2010.

[FOPT10] J. Faugère, A. Otmani, L. Perret and Jean-Pierre Tillich, *"Algebraic Cryptanalysis of McEliece Variants with Compact Keys"*, Advances in Cryptology - EUROCRYPT 2010, 2010.

[HGJ10] N. Howgrave-Graham and A. Joux, *"New Generic Algorithms for Hard Knapsacks."*, Advances in Cryptology - EUROCRYPT 2010, 2010.

[HMM10] W. Henecka, A.May and A.Meurer *"Correcting Errors in RSA Private Keys"*, Advances in Cryptology - CRYPTO 2010, 2010.

[NCBB10] R. Niebuhr, P.-L. Cayrel, S. Bulygin and J. Buchmann, *"On lower bounds for Information Set Decoding over $\mathbb{F}_q$", Symbolic Computation and Cryptography - SCC 2010, 2010.*

[Pet10] C. Peters, *"Information-Set Decoding for Linear Codes over $\mathbb{F}_q$"*, Post-Quantum Cryptography - PQCrypto 2010, 2010.

[BCJ11] A. Becker, J.S. Coron and A. Joux, *"Improved Generic Algorithms for Hard Knapsacks."*, Advances in Cryptology - EUROCRYPT 2011, 2011.

[BLP11a] D.J. Bernstein, T. Lange and C. Peters, *"Smaller Decoding Exponents: Ball-Collision Decoding"*, Advances in Cryptology - CRYPTO 2011, 2011. See `http://eprint.iacr.org/2010/585` for the full version.

[BLP11b] D.J. Bernstein, T. Lange and C. Peters, *"Wild McEliece Incognito"*, IACR Cryptology ePrint Archive 502/2011, 2011.

[HMR11] H. Dinh, C. Moore and A. Russell, *"McEliece and Niederreiter Cryptosystems That Resist Quantum Fourier Sampling Attacks"*, Advances in Cryptology, CRYPTO 2011, 2011.

[KPCJV11] E. Kiltz, K. Pietrzak, D. Cash, A. Jain and D. Venturi, *"Efficient Authentication from Hard Learning Problems"*, in Advances in Cryptology - EUROCRYPT 2011, 2011.

[MMT11] A. May, A. Meurer and E. Thomae, *"Decoding Random Linear Codes in $\tilde{\mathcal{O}}(2^{0.054n})$"*, Advances in Cryptology - ASIACRYPT 2011, 2011.

[Pet11] C. Peters, *"Curves, Codes, and Cryptography"*, PhD thesis, Technische Universiteit Eindhoven, 2011.

[Sen11] N.Sendrier, *"Decoding One Out of Many"*, Post-Quantum Cryptography - PQCrypto 2011, 2011.

[BJMM12] 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."*, Advances in Cryptology - EUROCRYPT 2012, 2012.

[DDKS12] I. Dinur, O. Dunkelman, N. Keller and A. Shamir, *"Efficient Dissection of Composite Problems, with Applications to Cryptanalysis, Knapsacks, and Combinatorial Search Problems."*, Advances in Cryptology - CRYPTO 2012, 2012.

[HM12] G. Herold and A. Meurer, *"New Attacks for Knapsack Based Cryptosystems"*, Security and Cryptography for Networks - SCN 2012, 2012.

[MS12] L. Minder and A. Sinclair, *"The Extended k-tree Algorithm"*, Journal of Cryptology Vol.25(2), 2012.

[MTSB12] R. Misoczki, J.-P.Tillich, N. Sendrier and P. Barreto, *"MDPC-McEliece: New McEliece Variants from Moderate Density Parity-Check Codes"*, IACR Cryptology ePrint Archive 409/2012, 2012.

[PPS12] K. Paterson, A. Polychroniadou and D. Sibborn, *"A Coding-Theoretic Approach to Recovering Noisy RSA Keys"*, Advances in Cryptology - ASIACRYPT 2012, 2012.

[Per12] E. Persichetti, *"On a CCA2-secure variant of McEliece in the standard model"*, IACR Cryptology ePrint Archive 2012/268, 2012.

[HS13] Y. Hamdaoui and N. Sendrier, *"A Non Asymptotic Analysis of Information Set Decoding"*, IACR Cryptology ePrint Archive 2013/162, 2013.

[KSI13] N. Kunihiro, N. Shinohara and T. Izu, *"Recovering RSA Secret Keys from Noisy Key Bits with Erasures and Errors"*, (to appear at) Public-Key Cryptography - PKC 2013, 2013