Statistical Decoding 2.0: Reducing Decoding to LPN

Kévin Carrier, Thomas Debris-Alazard, Charles Meyer-Hilfiger, Jean-Pierre Tillich

Statistical Decoding 2.0: Reducing Decoding to LPN

Kévin Carrier[1], Thomas Debris-Alazard[2], Charles Meyer-Hilfiger[3], and Jean-Pierre Tillich[3]

[1] ETIS Laboratory, CY Cergy-Paris University, kevin.carrier@ensea.fr
[2] Project GRACE, Inria Saclay-Ile de France, thomas.debris@inria.fr
[3] Project COSMIQ, Inria de Paris,
charles.meyer-hilfiger@inria.fr,jean-pierre.tillich@inria.fr

Abstract. The security of code-based cryptography relies primarily on the hardness of generic decoding with linear codes. The best generic decoding algorithms are all improvements of an old algorithm due to Prange: they are known under the name of information set decoders (ISD). A while ago, a generic decoding algorithm which does not belong to this family was proposed: statistical decoding. It is a randomized algorithm that requires the computation of a large set of parity-checks of moderate weight, and uses some kind of majority voting on these equations to recover the error. This algorithm was long forgotten because even the best variants of it performed poorly when compared to the simplest ISD algorithm. We revisit this old algorithm by using parity-check equations in a more general way. Here the parity-checks are used to get LPN samples with a secret which is part of the error and the LPN noise is related to the weight of the parity-checks we produce. The corresponding LPN problem is then solved by standard Fourier techniques. By properly choosing the method of producing these low weight equations and the size of the LPN problem, we are able to outperform in this way significantly information set decoders at code rates smaller than 0.3. It gives for the first time after 60 years, a better decoding algorithm for a significant range which does not belong to the ISD family.

1 Introduction

1.1 The Decoding Problem and Code-based Cryptography

Code-based cryptography relies crucially on the hardness of decoding generic linear codes which can be expressed as follows in the binary case

Problem 1.1 (decoding a linear code). *Let $\mathscr{C}$ be a binary linear code over $\mathbb{F}_2$ of dimension k and length n,* i.e. *a subspace of $\mathbb{F}_2^n$ of dimension k. We are given $\mathbf{y} \in \mathbb{F}_2^n$, an integer t and want to find a codeword $\mathbf{c} \in \mathscr{C}$ and an error vector $\mathbf{e} \in \mathbb{F}_2^n$ of Hamming weight $|\mathbf{e}| = t$ for which $\mathbf{y} = \mathbf{c} + \mathbf{e}$.*

This terminology stems from information theory, $\mathbf{y}$ is a noisy version of a codeword $\mathbf{c}$: $\mathbf{y} = \mathbf{c} + \mathbf{e}$ where $\mathbf{e}$ is a vector of weight t and we want to recover the original codeword $\mathbf{c}$. It can also be viewed as solving an underdetermined linear system with a weight constraint. Indeed, we can associate to a subspace $\mathscr{C}$ of dimension k of $\mathbb{F}_2^n$ a binary $(n-k) \times n$ matrix $\mathbf{H}$ (also called a *parity-check* matrix of the code) whose kernel defines $\mathscr{C}$, namely $\mathscr{C} = \{\mathbf{x} \in \mathbb{F}_2^n : \mathbf{H}\mathbf{x}^\intercal = \mathbf{0}\}$. The decoding problem is equivalent to find an $\mathbf{e}$ of Hamming weight t such that $\mathbf{H}\mathbf{e}^\intercal = \mathbf{s}^\intercal$ where $\mathbf{s}$ is the *syndrome* of $\mathbf{y}$ with respect to $\mathbf{H}$, *i.e.* $\mathbf{s}^\intercal = \mathbf{H}\mathbf{y}^\intercal$. This can be verified by observing that if there exists $\mathbf{c} \in \mathscr{C}$ and $\mathbf{e}$ such that $\mathbf{y} = \mathbf{c} + \mathbf{e}$ then $\mathbf{H}\mathbf{y}^\intercal = \mathbf{H}(\mathbf{c}+\mathbf{e})^\intercal = \mathbf{H}\mathbf{c}^\intercal + \mathbf{H}\mathbf{e}^\intercal = \mathbf{H}\mathbf{e}^\intercal$.

The decoding problem has been studied for a long time and despite many efforts on this issue [24, 26, 12, 2, 14, 4, 19, 3, 20] the best algorithms [3, 20, 5, 6] are exponential in the number of errors that have to be corrected: correcting t errors in a binary linear code of length n with the aforementioned algorithms has a cost of $2^{\alpha n(1+o(1))}$ where $\alpha = \alpha(R, \tau)$ is a constant depending of the code rate $R \triangleq \frac{k}{n}$, the error rate $\tau \triangleq \frac{t}{n}$ and the algorithm which is used. All the efforts that have been spent on this problem have only managed to decrease slightly this exponent α. Let us emphasize that this exponent is the key for estimating the security level of any code-based cryptosystem. We expect that this problem is the hardest at the Gilbert-Varshamov relative distance $\tau = \delta_{\text{GV}}$ where $\delta_{\text{GV}} \triangleq h^{-1}(1-R)$, with h being the binary entropy function $h(x) \triangleq -x \log_2 x - (1-x)\log_2(1-x)$ and $h^{-1}(x)$ its inverse ranging over $[0, \frac{1}{2}]$. This corresponds in the case of random linear codes to the largest relative weight below which there is typically just one solution of the decoding problem assuming that there is one. Above this bound, the number of solutions becomes exponential (at least as long as $\tau < 1-\delta_{\text{GV}}$) and this helps to devise more efficient decoders. Furthermore, all the aforementioned algorithms become polynomial in the regime $\frac{1-R}{2} \leq \tau \leq \frac{1+R}{2}$ (see an illustration of this behavior in Figure 1.1).

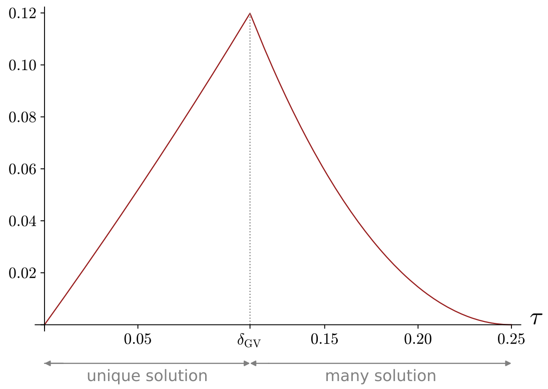


Fig. 1.1. Complexity exponent α of the Prange ISD algorithm [24] as a function of the error ratio $\tau \triangleq \frac{t}{n}$ at rate $R = \frac{1}{2}$. The peak corresponds to the normalized Gilbert-Varshamov distance $\delta_{\text{GV}} = h^{-1}(1-R)$.

There are code-based cryptographic primitives whose security relies precisely on the difficulty of decoding at the Gilbert-Varshamov relative distance (something which is also called *full distance decoding* [20, 5, 6]), for instance the Stern code-based identification schemes or associated signatures schemes [27, 16, 1, 13]. In the light of the upcoming NIST second call for new quantum resistant signature algorithms, it is even more important to have a stable and precise assessment of what we may expect about the complexity of solving this problem. For much smaller distances, say sub-linear, which is relevant for cryptosystems like [22, 21], the situation seems much more stable/well understood, since the complexity exponent of all the above-mentioned algorithms is the same in this regime [7].

1.2 ISD Algorithms and Beyond: Statistical Decoding

All the aforementioned algorithms can be viewed as a refinement of the original Prange algorithm [24] and are actually all referred to as Information Set Decoding (ISD) algorithms. Basically, they all use a common principle, namely making the bet that in a certain set of about k positions (the "information set") there are only very few errors and using this bet to speed-up decoding. The parameters of virtually all code-based cryptographic algorithms (for the Hamming metric) have been chosen according to the running time of this family of algorithms. Apart from these algorithms, there is one algorithm which is worth mentioning, namely statistical decoding. It was first proposed by Al Jabri in [17] and improved a little bit by Overbeck in [23]. Later on, [15] proposed an iterative version of this algorithm.

It is essentially a two-stage algorithm, the first step consisting in computing an exponentially large number of parity-check equations of the smallest possible weight w, and then from these parity-check equations the error is recovered by some kind of majority voting based on these equations. This majority voting is based on the following principle, take a parity-check equation $\mathbf{h}$ for the code $\mathscr{C}$ we want to decode, *i.e.* a binary vector $\mathbf{h} = (h_i)_{1 \leq i \leq n}$ such that $\langle \mathbf{h}, \mathbf{c} \rangle = 0$ for every $\mathbf{c}$ in $\mathscr{C}$. Assume that the i-th bit of the parity-check is 1, then since $\langle \mathbf{h}, \mathbf{y} \rangle = \langle \mathbf{h}, \mathbf{e} \rangle = e_i + \sum_{j \neq i} h_j e_j$, the i-th bit e_i of the error $\mathbf{e}$ we want to recover satisfies

$$e_i + \sum_{j \neq i} h_j e_j = \langle \mathbf{h}, \mathbf{y} \rangle . \tag{1.1}$$

The sum $\sum_{j \neq i} h_j e_j$ is biased, say it is equal to 1 with probability $\frac{1-\varepsilon}{2}$ with a bias ε which is (essentially) a decreasing function of the weight w of the parity-check $\mathbf{h}$. This allows to recover e_i with about $\Theta\left(1/\varepsilon^2\right)$ parity-checks. However the bias is exponentially small in the minimum weight of $\mathbf{h}$ and $\mathbf{e}$ and the complexity of such an algorithm is exponential in the codelength. An asymptotic analysis of this algorithm was performed in [9] and it turns out that even if we had a way to obtain freely the parity-check equations we need, this kind of algorithm could not even outperform the simplest ISD algorithm: the Prange algorithm. This is done in [9] by showing that there is no loss in generality if we just care about getting the best exponent to restrict ourselves to a single parity-check weight w (see Section 5 in [9]) and then analyse the complexity of such a putative algorithm for a single weight by using the knowledge of the typical number of parity-check equations of a given weight in a random linear code. The complexity exponent we get is a lower bound on the complexity of statistical decoding. We call such a putative statistical decoding algorithm, *genie-aided statistical decoding*: we are assisted by a genie which gives for free all the parity-check equations we require (but of course we can only get as much parity-check equations of some weight w as there exists in the code we want to decode). The analysis of the exponent we obtain with such genie-aided statistical decoding is given in [9, §7] and shows that it is outperformed very significantly by the Prange algorithm (see [9, §7.2]).

1.3 Contributions

In this work, we modify statistical decoding so that each parity-check yields now an LPN sample which is a noisy linear combination involving part of the error vector. This improves significantly statistical decoding, since the new decoding algorithm outperforms significantly all ISD's for code rates smaller than 0.3. It gives for the first time after 60 years, a better decoding algorithm that does not belong to the ISD family, and this for a very significant range of rates. The only other example where ISD algorithms have been beaten was in 1986, when Dumer introduced his collision technique. This improved the Prange decoder only for rates in the interval $[0.98, 1]$ and interestingly enough it gave birth to all the modern improvements of ISD algorithms starting from Stern's algorithm [26].

A New Approach : Using Parity-Checks to Reduce Decoding to LPN. Our approach for solving the decoding problem reduces it to the so-called Learning Parity with Noise Problem (LPN).

Problem 1.2 (LPN). *Let $\mathcal{O}_{\mathbf{s},\tau}(\cdot)$ be an oracle parametrized by $\mathbf{s} \in \mathbb{F}_2^s$ and $\tau \in [0,1]$ such that on a call it outputs $(\mathbf{a}, \langle \mathbf{s}, \mathbf{a} \rangle + e)$ where $\mathbf{a} \in \mathbb{F}_2^s$ is uniformly distributed and e is distributed according to a Bernoulli of parameter τ. We have access to $\mathcal{O}_{\mathbf{s},\tau}(\cdot)$ and want to find $\mathbf{s}$.*

(1.1) can be interpreted as an LPN sample with an $\mathbf{s}$ of size 1, namely e_i. However, if instead of splitting the support of the parity-check with one bit on one side and the other ones on the other side, but choose say s positions on the first part (say the s first ones) and $n - s$ on the other, we can write

$$\langle \mathbf{h}, \mathbf{y} \rangle = \underbrace{\sum_{i=1}^{s} h_i e_i}_{\text{linear comb.}} + \underbrace{\sum_{j>s} h_j e_j}_{\text{LPN noise}} .$$

We may interpret such a scalar product as an LPN sample where the secret is $(e_1, \cdots, e_s)$; *i.e.* we have a noisy information on a linear combination $\sum_{i=1}^{s} h_i e_i$ on the s first bits of the error where the noise is given by the term $\sum_{j>s} h_j e_j$

and the information is of the form $\sum_{i=1}^{s} h_i e_i + \text{noise} = \langle \mathbf{h}, \mathbf{y} \rangle$. Again the second linear combination is biased, say $\mathbb{P}\left(\sum_{j>s} h_j e_j = 1\right) = \frac{1-\varepsilon}{2}$ and information theoretic arguments show that again $\Theta\left(1/\varepsilon^2\right)$ samples are enough to determine $(e_1, \cdots, e_s)$. It seemed that we gained nothing here since we still need as many samples as before and it seems that now recovering $(e_1, \cdots, e_s)$ is much more complicated than performing majority voting.

However with this new approach, we just need parity-check equations of low weight on $n - s$ positions (those that determine the LPN noise) whereas in statistical decoding algorithm we have to compute parity-check equations of low weight on $n - 1$ positions. This brings us to the main advantage of our new method: the parity-checks we produce have much lower weight on those $n - s$ positions than those we produce for statistical decoding. This implies that the bias ε in the LPN noise is much bigger with the new method and the number $N = \Theta\left(1/\varepsilon^2\right)$ of parity-check equations much lower. Secondly, by using the fast Fourier transform, we can recover $(e_1, \cdots, e_s)$ in time $O(s2^s)$. Therefore, as long as the number of parity-checks we need is of order $\Omega\left(2^s\right)$, there is no exponential extra cost of having to recover $(e_1, \cdots, e_s)$. This new approach will be called from now on *Reduction to LPN* decoding (RLPN).

Subset Sum Techniques and Bet on the Error Distribution. As just outlined, our RLPN decoder needs an exponential number $N = \Theta\left(1/\varepsilon^2\right)$ of parity-checks of small weight on $n - s$ positions. This can be achieved efficiently by using collision/subset techniques used in the inner loop of ISD's. Recall that all ISD's proceed in two steps, (*i*) first they pick an augmented information set and (*ii*) then have an inner loop computing low weight codewords of some sort. Step (*ii*) uses advanced techniques to solve subset-sum problems like birthday paradox [10, 12], Wagner algorithm [28] or representations techniques [19, 3]. All these techniques can also be used in a natural way in our RLPN decoder to compute the low weight parity-checks we need.

Furthermore, another idea of ISD's can be used in our RLPN decoder. All ISD's are making, in a fundamental way, a bet on the error weight distribution in several zones related to the information set picked up in (*i*). There are two zones: the potentially augmented information set and the rest of the positions. ISD algorithms assume that the (augmented) information set contains only very few errors. A similar bet can be made in our case. We have two different zones: on one hand the s positions determining s error bits and on the other $n - s$ bits which determine the LPN noise. It is clearly favorable to have an error ratio which is smaller on the second part. The probability that this unlikely event happens is largely outweighed by the gain in the bias of the LPN noise.

Our Results. Using all the aforementioned ingredients results in dramatically improving statistical decoding (see Figure 1.2), especially in the low rate regime $(R \leq \frac{1}{2})$ where ISD algorithms are known to perform slightly worse than in the high rate regime $(R > \frac{1}{2})$. Indeed, the complexity exponent $\alpha(R) \stackrel{\triangle}{=} \alpha(R, \delta_{\mathrm{GV}}(R))$

of ISD's for full decoding (*a.k.a.* the GV bound decoding) which could be expected to be symmetric in R is actually bigger in the low rate regime than in the high rate regime: $\alpha(R) > \alpha(1-R)$ for $0 < R < \frac{1}{2}$. This results in an exponent curve which is slightly tilted towards the left, the maximum exponent being always obtained for $R < \frac{1}{2}$. Even worse, the behavior for very small rates (*i.e.* $R \to 0^+$) is fundamentally different in the very high rate regime ($R \to 1^-$). The complexity curve behaves like $\alpha(R) \approx R$ in the first case and like $\alpha(R) \approx \frac{1-R}{2}$ in the second (at least for all later improvements of the Prange decoder incorporating collision techniques). This behavior at 0 for full distance decoding has never been changed by *any* decoder. It should be noted that $\alpha(R) = R(1+o(1))$ around 0 means that the complexity behaves like $2^{\alpha(R)n} = 2^{R(1+o(1))n} = 2^{k(1+o(1))}$, so in essence ISD's are not performing really better than trivial enumeration on all codewords. This fundamental barrier is still unbroken by our RLPN decoder, but it turns out that $\alpha(R)$ approaches R much more slowly with RLPN. For instance, for $R = 0.02$ we have $\alpha(R) \approx \frac{R}{2}$. This behavior in the very low regime is instrumental for the improvement we obtain on ISD's. In essence, this improvement is due in this regime to the conjunction of RLPN decoding with a collision search of low weight parity-checks. This method can be viewed as the dual (*i.e.* operating on the dual code) of the collision search performed in advanced ISD's which are successful for lowering the complexity exponent down to $\alpha(R) \approx \frac{1-R}{2}$ in the high rate regime. In some sense, the RLPN strategy allows us to *dualize* advanced ISD techniques for working in the low rate regime.

All in all, using [3] (one of the most advanced ISD techniques) to compute low weight codewords of some shape we are able to outperform significantly even the latest improvements of ISD algorithms for code rates R smaller than 0.3 as shown in Figure 1.2. This is a breakthrough in this area, given the dominant role that ISD algorithms have played during all those years for assessing the complexity of decoding a linear code. Note however that the correctness of this algorithm relies on the LPN error model (Assumption 3.7) for which some recent experiments have found out not to be completely accurate (see https://github.com/tillich/RLPNdecoding/tree/master/verification_heuristic/histogram). However, experimental results seem to indicate that this LPN modeling can be replaced by the weaker Conjecture 3.11 which is compatible with the experiments we have made and for which there is a clear path to demonstrate its validity (see Subsection 3.4).

Proving the Standard Assumption of Statistical Decoding. In analyzing the new decoding algorithm, we also put statistical decoding on a much more rigorous foundation. We show that the basic condition that has to be met for both statistical decoding and RLPN decoding, namely that the number N of parity-check equations that are available is at least of order $\Omega\left(1/\varepsilon^2\right)$ in the case of statistical decoding and $\Omega\left(s/\varepsilon^2\right)$ in the case of RLPN decoding where ε is the bias of the LPN noise, is also essentially the condition which ensures that the bias is well approximated by the standard assumption made for statistical decoding which assumes that

$$\text{bias}\left(\langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle\right) \approx \text{bias}\left(\langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}'_{\mathscr{N}} \rangle\right), \tag{1.2}$$

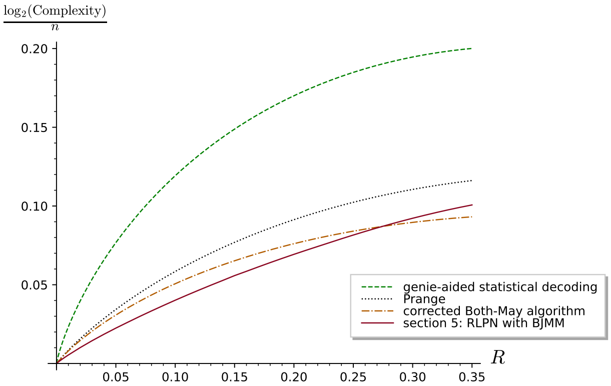


Fig. 1.2. Complexity exponent for full distance decoding of genie-aided statistical decoding [9, §7] (recall that this is a *lower bound* on the complexity exponent of statistical decoding), the basic Prange ISD algorithm [24], the best state-of-the-art algorithm of [6] (with a correction in the exponent, see the full version of this paper [8, Ap. B]) and our RLPN decoder as a function of R.

where bias(X) is defined for a binary random variable as bias$(X) \stackrel{\triangle}{=} \mathbb{P}(X=0) - \mathbb{P}(X=1)$, $\mathscr{N}$ is a subset of $n-s$ positions (those which are involved in the LPN noise), $\mathbf{h}$ is chosen uniformly at random among the parity-checks of weight w on $\mathscr{N}$ of the code $\mathscr{C}$ we decode whereas $\mathbf{h}'$ is chosen uniformly at random among the words of weight w on $\mathscr{N}$. We will namely prove that as soon as the parameters are chosen such that $N = \omega\left(1/\operatorname{bias}\left(\langle \mathbf{e}_{\mathscr{N}}, \mathbf{h'}_{\mathscr{N}}\rangle\right)^2\right)$, we have that for all but a proportion $o(1)$ of codes $\mathscr{C}$ (as proved in Proposition 3.1 in Subsection 3.1): $\operatorname{bias}\left(\langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}}\rangle\right) = (1+o(1))\operatorname{bias}\left(\langle \mathbf{e}_{\mathscr{N}}, \mathbf{h'}_{\mathscr{N}}\rangle\right)$.

2 Notation and Coding Theory Background

Vectors and matrices. Vectors and matrices are respectively denoted in bold letters and bold capital letters such as $\mathbf{a}$ and $\mathbf{A}$. The entry at index i of the vector $\mathbf{x}$ is denoted by x_i. The canonical scalar product $\sum_{i=1}^{n} x_i y_i$ between two vectors $\mathbf{x}$ and $\mathbf{y}$ of $\mathbb{F}_2^n$ is denoted by $\langle \mathbf{x}, \mathbf{y} \rangle$. Let $\mathscr{I}$ be a list of indexes. We denote by $\mathbf{x}_{\mathscr{I}}$ the vector $(x_i)_{i \in \mathscr{I}}$. In the same way, we denote by $\mathbf{A}_{\mathscr{I}}$ the sub-matrix made of the columns of $\mathbf{A}$ which are indexed by $\mathscr{I}$. The concatenation of two vectors $\mathbf{x}$ and $\mathbf{y}$ is denoted by $\mathbf{x}||\mathbf{y}$. The Hamming weight of a vector $\mathbf{x} \in \mathbb{F}_2^n$ is defined as the number of its non-zero coordinates, namely $|\mathbf{x}| \stackrel{\triangle}{=} \#\{i \in [\![1,n]\!] : x_i \neq 0\}$ where $\#\mathscr{A}$ stands for the cardinality of a finite set $\mathscr{A}$ and $[\![a,b]\!]$ stands for the set of the integers between a and b.

Probabilistic notation. For a finite set $\mathscr{S}$, we write $X \stackrel{\$}{\leftarrow} \mathscr{S}$ when X is an element of $\mathscr{S}$ drawn uniformly at random in it. For a Bernoulli random variable X, denote by $\mathrm{bias}(X)$ the quantity $\mathrm{bias}(X) \stackrel{\triangle}{=} \mathbb{P}(X=0) - \mathbb{P}(X=1)$. For a Bernoulli random variable X of parameter $p = \frac{1-\varepsilon}{2}$, *i.e.* $\mathbb{P}(X=1) = \frac{1-\varepsilon}{2}$, we have $\mathrm{bias}(X) = \varepsilon$.

Soft-O notation. For real valued functions defined over $\mathbb{R}$ or $\mathbb{N}$ we define $o()$, $O()$, $\Omega()$, $\Theta()$, in the usual way and also use the less common notation $\widetilde{O}()$ and $\widetilde{\Omega}()$, where $f = \widetilde{O}(g)$ means that $f(x) = O\Big(g(x)\log^k g(x)\Big)$ and $f = \widetilde{\Omega}(g)$ means that $f(x) = \Omega\left(g(x)\log^k g(x)\right)$ for some k. We will use this for functions which have an exponential behavior, say $g(x) = e^{\alpha x}$, in which case $f(x) = \widetilde{O}(g(x))$ means that $f(x) = O(P(x)g(x))$ where P is a polynomial in x. We also use $f = \omega(g)$ when f dominates g asymptotically; that is when $\lim\limits_{x \to \infty} \frac{|f(x)|}{g(x)} = \infty$.

Coding theory. A binary linear code $\mathscr{C}$ of length n and dimension k is a subspace of the vector space $\mathbb{F}_2^n$ of dimension k. We say that it has parameters $[n,k]$ or that it is an $[n,k]$-code. Its *rate* R is defined as $R \stackrel{\triangle}{=} \frac{k}{n}$. A generator matrix $\mathbf{G}$ for $\mathscr{C}$ is a full rank $k \times n$ matrix over $\mathbb{F}_2$ such that $\mathscr{C} = \left\{\mathbf{u}\mathbf{G} : \mathbf{u} \in \mathbb{F}_2^k\right\}$. In other words, the rows of $\mathbf{G}$ form a basis of $\mathscr{C}$. A parity-check matrix $\mathbf{H}$ for $\mathscr{C}$ is a full-rank $(n-k) \times n$ matrix over $\mathbb{F}_2$ such that $\mathscr{C} = \{\mathbf{c} \in \mathbb{F}_2^n : \mathbf{H}\mathbf{c}^\intercal = \mathbf{0}\}$. In other words, $\mathscr{C}$ is the null space of $\mathbf{H}$. The code whose generator matrix is the parity-check matrix of $\mathscr{C}$ is called the dual code of $\mathscr{C}$. It might be seen as the subspace of parity-checks of $\mathscr{C}$ and is defined equivalently as

Definition 2.1 (dual code). *The* dual code $\mathscr{C}^\perp$ *of an* $[n,k]$*-code* $\mathscr{C}$ *is an* $[n, n-k]$*-code which is defined by* $\mathscr{C}^\perp \stackrel{\triangle}{=} \{\mathbf{h} \in \mathbb{F}_2^n : \forall \mathbf{c} \in \mathscr{C},\ \langle \mathbf{c}, \mathbf{h} \rangle = 0\}$.

It will also be very convenient to consider the operation of puncturing a code, *i.e.* keeping only a subset of entries in a codeword.

Definition 2.2 (punctured code). *For a code* $\mathscr{C}$ *and a subset* $\mathscr{I}$ *of code positions, we denote by* $\mathscr{C}_\mathscr{I}$ *the punctured code obtained from* $\mathscr{C}$ *by keeping only the positions in* $\mathscr{I}$*,* i.e. $\mathscr{C}_\mathscr{I} = \{\mathbf{c}_\mathscr{I} : \mathbf{c} \in \mathscr{C}\}$.

We will also use several times that random binary linear codes can be decoded successfully, with a probability of error going to 0, as the codelength goes to infinity as long as the code rate is below the capacity, and this of any binary input symmetric channel whose definition is

Definition 2.3 (binary input memoryless symmetric channel). *A binary input memoryless symmetric channel (BIMS) with output a finite alphabet* $\mathscr{Y}$*, is an error model on* $\{0,1\}^*$ *assuming that when a bit* $b \in \{0,1\}$ *is sent, it gets mapped to* $y \in \mathscr{Y}$ *with probability denoted by* $p(y|b)$ *(these are the transition probabilities of the channel). Being symmetric means that there is an involution* f *such that* $p(y|0) = p(f(y)|1)$*. Being memoryless means that the outputs of the channel are independent conditioned on the inputs: when* $b_1 \cdots b_n \in \{0,1\}^n$ *is sent, the probability that the output is* $y_1 \cdots y_n$ *is given by* $p(y_1|b_1) \cdots p(y_n|b_n)$*.*

We use here this rather general formulation to analyze what is going on when we have several different LPN samples corresponding to the same parity-check $\mathbf{h}$. The error model that we have in this case will be more complicated than the standard binary symmetric channel (see Definition 2.6 below). The capacity of such a channel is given by

Definition 2.4 (capacity of a BIMS channel). *The capacity*[4] *C of a BIMS channel with transition probabilities $(p(y|b))_{\substack{y\in\mathscr{Y}\\ b\in\{0,1\}}}$ is given by*

$$C \stackrel{\triangle}{=} \sum_{y\in\mathscr{Y}} \sum_{b\in\{0,1\}} \frac{p(y|b)}{2} \log_2 \frac{p(y|b)}{\frac{1}{2}p(y|0)+\frac{1}{2}p(y|1)}.$$

LPN samples correspond to the binary symmetric channel (BSC) given by

Definition 2.5 (binary symmetric channel). *BSC(p) is a BIMS channel with output alphabet $\mathscr{Y}=\{0,1\}$ and transition probabilities $p(0|0)=p(1|1)=1-p$, $p(1|0)=p(0|1)=p$, where p is the crossover probability of the channel.*

In other words, this means that a bit b is transformed into its opposite $1-b$ with probability p when sent through the channel. It is readily verified that

Definition 2.6 (binary symmetric channel). *The capacity C of BSC(p) is given by $C=1-h(p)$.*

We will also talk about maximum likelihood decoding a code (under the assumption that the input codeword is chosen uniformly at random) for a given channel, meaning the following

Definition 2.7 (maximum likelihood decoding). *Maximum likelihood decoding of a binary code $\mathscr{C}\subset\{0,1\}^n$ over a BIMS channel with transitions probabilities $(p(y|b))_{\substack{y\in\mathscr{Y}\\ b\in\{0,1\}}}$ corresponds, given a received word $\mathbf{y}\in\mathscr{Y}^n$, to output the (or one of them if there are several equally likely candidates) codeword $\mathbf{x}$ which maximizes $p(\mathbf{y}|\mathbf{x})$. Here $p(\mathbf{y}|\mathbf{x})\stackrel{\triangle}{=}p(y_i|x_i)\cdots p(y_n|x_n)$ denotes the probability of receiving $\mathbf{y}$ given that $\mathbf{x}$ was sent.*

In a sense, this is the best possible decoding algorithm for a given channel model. There is a variation of Shannon's theorem (see for instance [25, Th. 4.68 p. 203]) which says that a family of random binary linear codes $(\mathscr{C}_n)_n$ attain the capacity of a BIMS channel.

Theorem 2.8. *Consider a BIMS channel of capacity C. Let $\delta>0$ and consider a family of random binary linear codes $\mathscr{C}_n$ of length n and rate smaller than $(1-\delta)C$ obtained by choosing their generator matrix uniformly at random. Then under maximum likelihood decoding, the probability of error after decoding goes to 0 as n tends to infinity.*

[4] The formula given here is strictly speaking the symmetric capacity of a channel, but these two notions coincide in the case of a BIMS channel.

3 Reduction to LPN and the Associated Algorithm

The purpose of this section is (i) to explain in detail the reduction to LPN, (ii) to give a high level description of the algorithm which does not specify the method for finding the dual codewords we need, and then (iii) to give its complexity. We assume from now on that we are given $\mathbf{y}$ which is equal to a sum of a codeword $\mathbf{c}$ of the code $\mathscr{C}$ we want to decode plus an error vector $\mathbf{e}$ of Hamming weight t:

$$\mathbf{y} = \mathbf{c} + \mathbf{e}, \;\; \mathbf{c} \in \mathscr{C}, \;\; |\mathbf{e}| = t.$$

We will start this section by explaining how we reduce decoding to an LPN problem and also show how the LPN noise can be estimated accurately.

3.1 Reduction to LPN

Recall that in RLPN decoding we first randomly select a subset $\mathscr{P}$ of s positions

$$\mathscr{P} \subseteq [\![1,n]\!] \quad \text{such that} \quad \#\mathscr{P} = s$$

where s is a parameter that will be chosen later. $\mathscr{P}$ corresponds to the entries of $\mathbf{e}$ we aim to recover and is the secret in the LPN problem. We denote by $\mathscr{N} \stackrel{\triangle}{=} [\![1,n]\!] \setminus \mathscr{P}$ the complementary set, with a choice of the letter $\mathscr{N}$ standing for "noise" for reasons that will be clear soon. Given $\mathbf{h} \in \mathscr{C}^{\perp}$, we compute,

$$\langle \mathbf{y}, \mathbf{h} \rangle = \langle \mathbf{e}, \mathbf{h} \rangle = \sum_{j \in \mathscr{P}} h_j e_j + \sum_{j \in \mathscr{N}} h_j e_j = \langle \mathbf{e}_{\mathscr{P}}, \mathbf{h}_{\mathscr{P}} \rangle + \langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle$$

It gives access to the following LPN sample:

$$(\mathbf{a}, \langle \mathbf{s}, \mathbf{a} \rangle + e) \quad \text{where} \quad \mathbf{s} \stackrel{\triangle}{=} \mathbf{e}_{\mathscr{P}}, \; \mathbf{a} \stackrel{\triangle}{=} \mathbf{h}_{\mathscr{P}} \; \text{ and } \; e \stackrel{\triangle}{=} \langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle.$$

Here e follows a Bernoulli distribution that is a function of n, s and u (*resp.* w) the weight of $\mathbf{e}$ (*resp.* $\mathbf{h}$) restricted to $\mathscr{N}$, namely

$$u \stackrel{\triangle}{=} |\mathbf{e}_{\mathscr{N}}| \quad \text{and} \quad w \stackrel{\triangle}{=} |\mathbf{h}_{\mathscr{N}}|\,.$$

The probability that e is equal to 1 is estimated through the following proposition which gives for the first time a rigorous statement for the standard assumption (1.2) made for statistical decoding.

Proposition 3.1. *Assume that the code $\mathscr{C}$ is chosen by picking for it an $(n-k) \times n$ binary parity-check matrix uniformly at random. Let $\mathscr{N}$ be a fixed set of $n-s$ positions in $[\![1,n]\!]$ and $\mathbf{e}$ be some error of weight u on $\mathscr{N}$. Choose $\mathbf{h}$ uniformly at random among the parity-checks of $\mathscr{C}$ of weight w on $\mathscr{N}$ and $\mathbf{h}'$ uniformly at random among the words of weight w on $\mathscr{N}$. Let $\delta \stackrel{\triangle}{=} \operatorname{bias}(\langle \mathbf{e}, \mathbf{h}' \rangle)$. If the parameters k, s, u, w are chosen as functions on n so that for n going to infinity, the expected number N of parity-checks of $\mathscr{C}$ of weight w on $\mathscr{N}$ satisfies $N = \omega\left(1/\delta^2\right)$ then for all but a proportion $o(1)$ of codes we have*

$$\operatorname{bias}(\langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle) = (1 + o(1))\delta.$$

Proof. Let us define for $b \in \{0,1\}$:

$$E_b \overset{\triangle}{=} \#\{\mathbf{h} \in \mathscr{C}^\perp : \quad |\mathbf{h}_{\mathscr{N}}| = w, \ \langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle = b\} \tag{3.1}$$

$$E'_b \overset{\triangle}{=} \#\{\mathbf{h}' \in \mathbb{F}_2^n : \quad |\mathbf{h}'_{\mathscr{N}}| = w, \ \langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}'_{\mathscr{N}} \rangle = b\} \tag{3.2}$$

By using [2, Lemma 1.1 p.10][5], we obtain

$$\mathbb{E}(E_b) = \frac{E'_b}{2^k} \quad \text{and} \quad \mathbf{Var}\left(E_b\right) \leq \frac{E'_b}{2^k}.$$

By using now the Bienaymé-Tchebychev inequality, we obtain for any function f mapping the positive integers to positive real numbers:

$$\mathbb{P}_{\mathscr{C}}\left(|E_b - \mathbb{E}(E_b)| \geq \sqrt{f(n)\mathbb{E}(E_b)}\right) \leq \frac{1}{f(n)}. \tag{3.3}$$

Since $\text{bias}\left(\langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle\right) = \frac{E_0 - E_1}{E_0 + E_1}$ we have with probability greater than $1 - \frac{2}{f(n)}$

$$\frac{\mu_0 - \mu_1 - \sqrt{2f(n)}\sqrt{\mu_0 + \mu_1}}{\mu_0 + \mu_1 + \sqrt{2f(n)}\sqrt{\mu_0 + \mu_1}} \leq \text{bias}\left(\langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle\right) \leq \frac{\mu_0 - \mu_1 + \sqrt{2f(n)}\sqrt{\mu_0 + \mu_1}}{\mu_0 + \mu_1 - \sqrt{2f(n)}\sqrt{\mu_0 + \mu_1}} \tag{3.4}$$

where $\mu_i \overset{\triangle}{=} \mathbb{E}(E_i)$ and where we used that for all positive x and y, $\sqrt{x} + \sqrt{y} \leq \sqrt{2(x+y)}$. We let $f(n) = \delta\sqrt{N}/2$. Since $N = \mu_0 + \mu_1$ this implies $f(n) = \delta\sqrt{\mu_0 + \mu_1}/2$. By the assumptions made in the proposition, note that $f(n)$ tends to infinity as n tends to infinity. We notice that

$$\sqrt{2f(n)}\sqrt{\mu_0 + \mu_1} = \delta^{1/2}(\mu_0 + \mu_1)^{3/4} = o\left(\delta(\mu_0 + \mu_1)\right) \tag{3.5}$$

because

$$\frac{\delta^{1/2}(\mu_0 + \mu_1)^{3/4}}{\delta(\mu_0 + \mu_1)} = \frac{1}{\sqrt{\delta\sqrt{\mu_0 + \mu_1}}} = \frac{1}{\sqrt{2f(n)}} \to 0 \text{ as } n \to \infty.$$

Equation (3.4) can now be rewritten as

$$\frac{\mu_0 - \mu_1 - o\left(\delta(\mu_0 + \mu_1)\right)}{\mu_0 + \mu_1 + o\left(\delta(\mu_0 + \mu_1)\right)} \leq \text{bias}\left(\langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle\right) \leq \frac{\mu_0 - \mu_1 + o\left(\delta(\mu_0 + \mu_1)\right)}{\mu_0 + \mu_1 - o\left(\delta(\mu_0 + \mu_1)\right)} \tag{3.6}$$

Now, on the other hand

$$\delta = \text{bias}\left(\langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}'_{\mathscr{N}} \rangle\right) = \frac{E'_0 - E'_1}{E'_0 + E'_1} = \frac{\frac{E'_0}{2^k} - \frac{E'_1}{2^k}}{\frac{E'_0}{2^k} + \frac{E'_1}{2^k}} = \frac{\mu_0 - \mu_1}{\mu_0 + \mu_1} \ \text{(by (3.1))}.$$

From this it follows that we can rewrite (3.6) as

$$\frac{\delta}{1 + o(\delta)} - o(\delta) \leq \text{bias}\left(\langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle\right) \leq \frac{\delta}{1 - o(\delta)} + o(\delta) \tag{3.7}$$

from which it follows immediately that $\text{bias}\left(\langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle\right) = \delta(1 + o(1))$. □

[5] Note that there is an additional condition "Suppose Lq^{-r} grows exponentially in n" in the statement of this lemma, but it is readily seen that this condition is neither necessary nor used in the proof.

Remark 3.2. Note that the condition $N = \Omega\left(1/\delta^2\right)$, respectively $N = \Omega\left(s/\delta^2\right)$ is the condition we need in order that statistical decoding, respectively RLPN decoding succeed. This means that if we just have slightly more equations than the ratio $\frac{1}{\delta^2}$, then the standard assumption (1.2) made for statistical decoding holds. The point of this assumption is that it allows easily to estimate the bias as the following lemma shows.

Lemma 3.3. *Under the same assumptions made in Proposition 3.1, we have that for all but a proportion* $o(1)$ *of codes,*

$$\mathrm{bias}(\langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle) = \delta(1+o(1)) \quad \textit{with} \quad \delta \stackrel{\triangle}{=} \frac{K_w^{n-s}(u)}{\binom{n-s}{w}}$$

where $u \stackrel{\triangle}{=} |\mathbf{e}_{\mathscr{N}}|$ *and* K_w^n *stands for the Krawtchouk polynomial of order* n *and degree* $w \in [\![0,n]\!]$ *which is defined as:*

$$K_w^n(X) \stackrel{\triangle}{=} \sum_{j=0}^{w} (-1)^j \binom{X}{j}\binom{n-X}{w-j}.$$

Proof. By using Proposition 3.1 (and the same notation as the one used there) we have that for all but a proportion $o(1)$ of codes $\mathrm{bias}\left(\langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle\right) = (1 + o(1))\,\mathrm{bias}\left(\langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}'_{\mathscr{N}} \rangle\right)$. Now by definition of u, we have

$$\begin{aligned}
\mathrm{bias}\left(\langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}'_{\mathscr{N}} \rangle\right) &= \frac{1}{\binom{n-s}{w}} \sum_{j \text{ even}} \binom{u}{j}\binom{n-s-u}{w-j} - \frac{1}{\binom{n-s}{w}} \sum_{j \text{ odd}} \binom{u}{j}\binom{n-s-u}{w-j} \\
&= \frac{1}{\binom{n-s}{w}} \sum_{j} (-1)^j \binom{u}{j}\binom{n-s-u}{w-j} \\
&= \frac{K_w^{n-s}(u)}{\binom{n-s}{w}}.
\end{aligned}$$

□

We will now repeatedly denote by *bias* of the LPN sample the quantity ε appearing in the previous lemma and the *estimated bias* the quantity namely

Definition 3.4 (bias of the LPN samples). *The* bias ε *of the LPN samples is defined by*

$$\varepsilon \stackrel{\triangle}{=} \mathrm{bias}(\langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle)$$

when $\mathbf{e}_{\mathscr{N}}$ *has Hamming weight* u *and* $\mathbf{h}$ *is drawn uniformly at random among the parity-check equations of weight* w *restricted on* $\mathscr{N}$*. The* estimated bias *is the quantity* δ *defined by*

$$\delta \stackrel{\triangle}{=} \mathrm{bias}(\langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}'_{\mathscr{N}} \rangle)$$

when $\mathbf{e}_{\mathscr{N}}$ *has Hamming weight* u *and* $\mathbf{h}'$ *is drawn uniformly at random among the binary words of weight* w *restricted on* $\mathscr{N}$*. This quantity is equal to*

$$\delta = \frac{K_w^{n-s}(u)}{\binom{n-s}{w}}.$$

The point of introducing Krawtchouk polynomials is that we can bring in asymptotic expansions of Krawtchouk polynomials. Most of the relevant properties we need about Krawtchouk polynomials are given in [18, §II.B]. They can be summarized by

Proposition 3.5. *1.* Value at 0. *For all* $0 \leq w \leq n$, $K_w^n(0) = \binom{n}{w}$.

2. Reciprocity. *For all* $0 \leq t, w \leq n$, $\binom{n}{t} K_w^n(t) = \binom{n}{w} K_t^n(w)$.

3. Roots. *The polynomials* K_w^n *have* w *distinct roots which lie in the interval*

$$\left\llbracket n/2 - \sqrt{w(n-w)}, n/2 + \sqrt{w(n-w)} \right\rrbracket .$$

The distance between roots is at least 2 *and at most* $o(n)$.

4. Magnitude outside the root region. *We set* $\tau \triangleq \frac{t}{n}$, $\omega \triangleq \frac{w}{n}$. *We assume* $w \leq n/2$ *and* $t \leq n/2 - \sqrt{w(n-w)}$. *Let* $z \triangleq \frac{1-2\tau-\sqrt{D}}{2(1-\omega)}$ *where* $D \triangleq (1-2\tau)^2 - 4\omega(1-\omega)$. *We have*

$$K_w^n(t) = 2^{n(\tau \log_2(1-z) + (1-\tau)\log_2(1+z) - \omega \log_2 z + o(1))}. \tag{3.8}$$

5. Magnitude in the root region. *Between any two consecutive roots of* K_w^n, *where* $1 \leq w \leq \frac{n}{2}$, *there exists* t *such that:*

$$K_w^n(t) = 2^{n\left(\frac{1+h(\omega)-h(\tau)}{2} + o(1)\right)} \text{ where } \omega \triangleq \frac{w}{n} \text{ and } \tau \triangleq \frac{t}{n}. \tag{3.9}$$

By using this proposition, we readily obtain

Proposition 3.6 (exponential behavior of δ^2). *Let* τ *and* ω *be two reals in the interval* $\left[0, \frac{1}{2}\right]$. *Let* $\omega^\perp \triangleq \frac{1}{2} - \sqrt{\omega(1-\omega)}$ *and* $z \triangleq \frac{1-2\tau-\sqrt{D}}{2(1-\omega)}$ *where* $D \triangleq (1-2\tau)^2 - 4\omega(1-\omega)$. *There exists a sequence of positive integers* $(t_n)_{n\in\mathbb{N}}$ *and* $(w_n)_{n\in\mathbb{N}}$, *such that* $\frac{t_n}{n} \underset{n\to\infty}{\to} \tau$, $\frac{w_n}{n} \underset{n\to\infty}{\to} \omega$ *and* $\frac{\log_2(K_{w_n}^n(t_n)^2/\binom{n}{w_n}^2)}{n}$ *has a limit which we denote* $\widetilde{\delta}(\tau, \omega)$ *with*

$$\widetilde{\delta}(\tau,\omega) = \begin{cases} 2\left(\tau \log_2(1-z) + (1-\tau)\log_2(1+z) - \omega \log_2 z - h(\omega)\right) & \text{if } \tau \in [0, \omega^\perp] \\ 1 - h(\tau) - h(\omega) & \text{otherwise.} \end{cases}$$

Proof. In the case $\tau \in [0, \omega^\perp]$ we just let $t_n = \lceil \tau n \rceil$, $w_n = \lceil \omega n \rceil$ and use directly the asymptotic expansion (3.8). In the case $\tau \in \left[\omega^\perp, \frac{1}{2}\right]$ we still define w_n with $w_n \triangleq \lceil \omega n \rceil$ but define t_n differently. For n large enough, we know from Proposition 3.5 that $\lceil \tau n \rceil$ lies between two zeros of the Krawtchouk polynomial and that there exists an integer t_n in this interval such that $\frac{\log_2(K_{w_n}^n(t_n))}{n} = \frac{1+h(\omega)-h(\tau_n)}{2} + o(1)$ where $\tau_n = \frac{t_n}{n}$. Now since the size of this interval is an $o(n)$ we necessarily have $\tau_n = \tau + o(1)$ and therefore $\frac{\log_2(K_{w_n}^n(t_n))}{n} = \frac{1+h(\omega)-h(\tau)}{2} + o(1)$. □

The point of this proposition is that the term $2\log_2(K_w^{n-s}(u)/\binom{n-s}{w})$ quantifies the exponential behaviour of the square ε^2 of the bias ε (see Lemma 3.3) and $1/\varepsilon^2$ is up to polynomial terms the number of parity-checks we need for having enough information to solve the LPN problem as will be seen. This is because the capacity of the BSC($\frac{1-\varepsilon}{2}$) is $1 - h\left(\frac{1-\varepsilon}{2}\right) = \theta(\varepsilon^2)$ and that solving an LPN-problem with a secret of size s and N samples amounts to be able to decode a random linear code of rate $\frac{s}{N}$ over the BSC($\frac{1-\varepsilon}{2}$). It is therefore doable as soon as the rate is below the capacity (see Theorem 2.8). The reason why the Shannon capacity appears here is because of the following heuristic/assumption we will make here:

Assumption 3.7 (LPN modelling). *We will assume that the $\langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle$ are i.i.d Bernoulli random variables of parameter $\frac{1-\varepsilon}{2}$.*

Strictly speaking, the corresponding random variables are not independent. However, note that similar heuristics are also used to analyze a related lattice decoder making use of short dual lattice vectors (they are called dual attacks in the literature). We will discuss this assumption in more depth in Subsection 3.4. Assumption 3.7 models the LPN noise as a binary symmetric channel BSC($\frac{1-\varepsilon}{2}$) of crossover probability $\frac{1-\varepsilon}{2}$. A straightforward application of Theorem 2.8 together with the fact that the capacity of a binary symmetric BSC($\frac{1-\varepsilon}{2}$) is $1 - h\left(\frac{1-\varepsilon}{2}\right) = \Omega(\varepsilon^2)$ implies

Fact 3.8. *With Assumption 3.7, the number N of LPN samples is such that $s/N = O(\varepsilon^2)$ for a small enough constant in the O, performing maximum-likelihood decoding of the corresponding $[N, s]$ binary code recovers the secret $\mathbf{e}_{\mathscr{P}}$ with probability $1 - o(1)$.*

Performing maximum likelihood decoding of the corresponding code can be achieved by a fast Fourier transform on a relevant vector. Indeed, for a given received word $\mathbf{y}$ and a set $\widetilde{\mathscr{H}}$ of N parity-checks so that their restriction to $\mathscr{P}$ leads to a set $\mathscr{H}$ of N different vectors of $\mathbb{F}_2^s$, we let for $\mathbf{a} \in \mathscr{H}$, $\widetilde{\mathbf{a}}$ be the unique parity-check in $\widetilde{\mathscr{H}}$ such that $\widetilde{\mathbf{a}}_{\mathscr{P}} = \mathbf{a}$ and define $f_{\mathbf{y},\mathscr{H}}$ as

$$f_{\mathbf{y},\mathscr{H}} : \mathbf{a} \in \mathbb{F}_2^s \mapsto \begin{cases} (-1)^{\langle \mathbf{y}, \widetilde{\mathbf{a}} \rangle} & \text{if } \mathbf{a} \in \mathscr{H} \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

We define the Fourier transform of such a function by $\widehat{f}(\mathbf{x}) \stackrel{\triangle}{=} \sum_{\mathbf{u} \in \mathbb{F}_2^s} f(\mathbf{u})(-1)^{\langle \mathbf{x}, \mathbf{u} \rangle}$. The code $\mathscr{D}$ we want to decode (obtained via our LPN samples) is described as

$$\mathscr{D} \stackrel{\triangle}{=} \{\mathbf{c}_{\mathbf{x}},\ \mathbf{x} \in \mathbb{F}_2^s\} \text{ where } \mathbf{c}_{\mathbf{x}} \stackrel{\triangle}{=} (\langle \mathbf{x}, \mathbf{a} \rangle)_{\mathbf{a} \in \mathscr{H}}, \tag{3.11}$$

and the word $\mathbf{u}_{\mathbf{y},\mathscr{H}}$ we want to decode is given by $\mathbf{u}_{\mathbf{y},\mathscr{H}} = (\langle \mathbf{y}, \widetilde{\mathbf{a}} \rangle)_{\mathbf{a} \in \mathscr{H}}$. It is readily seen that

$$\widehat{f_{\mathbf{y},\mathscr{H}}}(\mathbf{x}) = \sum_{\mathbf{a} \in \mathbb{F}_2^s} f(\mathbf{a})(-1)^{\langle \mathbf{x}, \mathbf{a} \rangle} = \sum_{\mathbf{a} \in \mathscr{H}} (-1)^{\langle \mathbf{x}, \mathbf{a} \rangle + \langle \mathbf{y}, \tilde{\mathbf{a}} \rangle} = \#\mathscr{H} - 2|\mathbf{u}_{\mathbf{y},\mathscr{H}} - \mathbf{c}_{\mathbf{x}}|.$$

In other words, finding the closest codeword to $\mathbf{u}_{\mathbf{y},\mathscr{H}}$ is nothing but finding the $\mathbf{x}$ which maximizes $\widehat{f_{\mathbf{y},\mathscr{H}}}(\mathbf{x})$. This is achieved in time $O(s2^s)$ by performing a fast Fourier transform. Notice that an exhaustive search would cost $O\big(2^{2s}\big)$.

3.2 Sketch of the whole algorithm

Algorithm 3.1. RLPN decoder

```
    Input: $\mathbf{y}$, $t$, $\mathscr{C}$ an $[n,k]$-code
    Output: $\mathbf{e}$ such that $|\mathbf{e}| = t$ and $\mathbf{y} - \mathbf{e} \in \mathscr{C}$.
  function RLPNdecode($\mathbf{y}$, $\mathscr{C}$, $t$)
      $s, u \leftarrow$ Optim($t$, $k$, $n$)
              ▷ $s$ and $u$ in order to minimize the complexity of the following procedure.
      for $i$ from 1 to $\mathrm{N}_{\mathrm{iter}}$ do              ▷ $\mathrm{N}_{\mathrm{iter}}$ is a certain function of $n$, $s$, $t$ and $u$.
          $\mathscr{P} \xleftarrow{\$} \{\mathscr{I} \subseteq [\![1,n]\!] \ : \ \#\mathscr{I} = s\}$
          $\mathscr{N} \leftarrow [\![1,n]\!] \setminus \mathscr{P}$
          $\mathscr{H} \leftarrow$ Create$(N, w, \mathscr{P})$
          $\widehat{f_{\mathbf{y},\mathscr{H}}} \leftarrow$FFT$(f_{\mathbf{y},\mathscr{H}})$
          $\mathbf{x}_0 \leftarrow \arg\max \widehat{f_{\mathbf{y},\mathscr{H}}}$
          if $\widehat{f_{\mathbf{y},\mathscr{H}}}(\mathbf{x}_0) \geq \frac{\delta N}{2}$ then              ▷ $\delta \triangleq K_w^{n-s}(u)/\binom{n-s}{w}$.
              return $\mathbf{e}$ such that $\mathbf{e}_{\mathscr{P}} = \mathbf{x}_0$ and $\mathbf{e}_{\mathscr{N}} =$ RLPNdecode$(\mathbf{y}_{\mathscr{N}}, \mathscr{C}_{\mathscr{N}}, t - |\mathbf{x}_0|)$)
          end if
      end for
  end function
```

Besides, the fast Fourier transform solving the LPN problem, Algorithm 3.1 uses two other ingredients:

- A routine Create$(N, w, \mathscr{P})$ creating a set $\mathscr{H}$ of N parity-check equations $\mathbf{h}$ such that $|\mathbf{h}_{\mathscr{N}}| = w$ where $\mathscr{N} \triangleq [\![1,n]\!] \setminus \mathscr{P}$. We will not specify how this function is realized here: this is done in the following sections. This procedure together with an FFT for decoding the code associated to the parity-check equations in $\mathscr{H}$ (see Equation (3.11)) form the inner loop of our algorithm.
- An outer loop making a certain number $\mathrm{N}_{\mathrm{iter}}$ of calls to the inner procedure, checking each time a new set $\mathscr{P}$ of s positions with the hope of finding an $\mathscr{N}$ containing an unusually low number u of errors in it. The point is that with a right u, the number of times we will have to check a new $\mathscr{P}$ is outweighed by the decrease in N because the bias δ is much higher for such a u.

3.3 Analysis of the RLPN decoder

We need to show now that our RLPN decoder returns what we expect. It is what the following proposition shows (a proof can be found in the full paper [8]).

Proposition 3.9 (acceptation criteria). *Under Assumption 3.7, by choosing* $\mathrm{N}_{\mathrm{iter}} = \omega\left(\frac{1}{P_{succ}}\right)$ *(where* P_{succ} *is the probability over the choice of* $\mathscr{N}$ *that there are exactly* u *errors in* $\mathscr{N}$*),* $s = \omega(1)$ *and* $N = \omega\left(\frac{n}{\delta^2}\right)$*, we have with probability* $1 - o(1)$ *that at least one iteration is such that* $\mathbf{e}_{\mathscr{P}}$ *meets the acceptation criteria*

$\widehat{f_{\mathbf{y},\mathscr{H}}}(\mathbf{e}_{\mathscr{P}}) \geq \frac{\delta N}{2}$. *Moreover, the probability that there exists* $\mathbf{x} \neq \mathbf{e}_{\mathscr{P}}$ *which meets this acceptation criteria is* $o(1)$*.*

The space and time complexity of this method are given by

Proposition 3.10. *Assume that* Create*(*N*,*w*,*$\mathscr{P}$*) produces* N *parity-check equations in space* S_{eq} *and time* T_{eq}*. The probability* P_{succ} *(over the choice of* $\mathscr{N}$*) that there are exactly* u *errors in* $\mathscr{N}$ *is given by* $P_{succ} = \frac{\binom{s}{t-u}\binom{n-s}{u}}{\binom{n}{t}}$*. The space complexity* S *and the time complexity* T *of the RLPN-decoder are given by*

$$\textbf{Space: } S = O(S_{eq} + 2^s)\,, \quad \textbf{Time: } T = \widetilde{O}\left(\frac{T_{eq} + 2^s}{P_{succ}}\right).$$

The parameters s*,* u *and* w *have to meet the following constraints*

$$N \leq 2^s \tag{3.12}$$

$$N \leq \frac{\binom{n-s}{w}}{2^{k-s}}. \tag{3.13}$$

Under Assumption 3.7 the algorithm outputs the correct $\mathbf{e}_{\mathscr{P}}$ *with probability* $1 - o(1)$ *if in addition we choose* N *and* N_{iter} *such that*

$$N = \omega\left(n\left(\frac{\binom{n-s}{w}}{K_w^{n-s}(u)}\right)^2\right) \tag{3.14}$$

$$N_{iter} = \omega\left(\frac{1}{P_{succ}}\right). \tag{3.15}$$

Proof. All the points are straightforward here, with the exception of the constraints. The first constraint is that the number of parity-checks should not be bigger than the total number of different LPN samples we can possibly produce. The second one is that the number of parity-checks needed is smaller than the number of available parity-checks. The conditions ensuring the correctness of the algorithm follow immediately from Proposition 3.9. □

3.4 On the validity of Assumption 3.7

The proof of the correctness of the algorithm relies on the validity of the LPN modelling (Assumption 3.7). We have programmed this algorithm and have verified that for several parameters it gives the correct answer. The corresponding experiments with the programs that have been used for running them can be found on https://github.com/tillich/RLPNdecoding. However, we have also found out (see https://github.com/tillich/RLPNdecoding/tree/master/verification_heuristic/histogram) that the second largest Fourier coefficient (the one which corresponds to the second nearest codeword, besides $\mathbf{e}_{\mathscr{P}}$) does not behave in the same way in the LPN model as in practice with the noise given by the $\langle \mathbf{h}_{\mathscr{N}}, \mathbf{e}_{\mathscr{N}} \rangle$'s.

This can be traced back to the fact that $\langle \mathbf{h}_{\mathscr{N}}, \mathbf{e}_{\mathscr{N}} \rangle$ and $\langle \mathbf{h}'_{\mathscr{N}}, \mathbf{e}_{\mathscr{N}} \rangle$ are positively correlated when $\mathbf{h}_{\mathscr{N}}$ and $\mathbf{h}'_{\mathscr{N}}$ are close to each other in Hamming distance. Actually these correlations have an effect on the tails of the largest Fourier coefficients as demonstrated in Figure 3.1 which display longer tails corresponding to the largest Fourier coefficients in the case of a noise produced by $\langle \mathbf{h}_{\mathscr{N}}, \mathbf{e}_{\mathscr{N}} \rangle$'s (called parity-checks in the figure) instead of Fourier coefficients produced by decoding a code with a BSC($\frac{1-\varepsilon}{2}$) noise (called BSC in the figure). This phenomenon vanishes when k gets larger as can be verified in Figure 3.1 or on https://github.com/tillich/RLPNdecoding/tree/master/verification_heuristic/histogram. From our experiments (see more details on https://github.com/tillich/RLPNdecoding) this phenomenon is not severe enough to prevent Algorithm 3.1 from working but needs some adjustments about how larger N has to be in terms of $\frac{1}{\delta^2}$. This experimental evidence leads us to conjecture

Conjecture 3.11. Algorithm 3.1 is successful if we replace in Proposition 3.10 the condition $N = \omega\left(n\left(\frac{\binom{n-s}{w}}{K_w^{n-s}(u)}\right)^2\right)$ by the slightly stronger condition $N = \omega\left(n^{\alpha}\left(\frac{\binom{n-s}{w}}{K_w^{n-s}(u)}\right)^2\right)$ for a certain $\alpha \geq 1$.

If this conjecture is true, then obviously the asymptotic exponent of the complexity is unchanged if we replace Assumption 3.7 by Conjecture 3.11. A semi-heuristic way to verify this conjecture could be to proceed as follows

1. Let W be the weight of the vector $\left(\left\langle \widetilde{\mathbf{h}}_{\mathscr{N}}, \mathbf{e}_{\mathscr{N}} \right\rangle\right)_{\mathbf{h} \in \widetilde{\mathscr{H}}}$. Compute $\mathbf{Var}\,(W)$ and prove that $\mathbf{Var}\,(W)$ is of order $O\big(n^{\beta} N\big)$ where β is some constant.
2. Use this computation to bound heuristically the tails of the Fourier coefficients and use this computation of $\mathbf{Var}\,(W)$ to give an estimation for the second largest Fourier coefficient when decoding the $[N, s]$-code which agrees with the experimental evidence.
3. Use this to prove that the second largest Fourier coefficient is typically far away enough from the first one to prove the validity of Conjecture 3.11.

4 Collision techniques for finding low weight parity-checks

4.1 Using the [10] method

A way for creating parity-checks with a low weight on $\mathscr{N}$ is simply to use subset-sum/collision techniques [10, 26, 11]. We start here with the simplest method for performing such a task pioneered by Dumer in [10]. Consider a parity-check matrix $\mathbf{H}$ for the code $\mathscr{C}$ we want to decode and keep only the columns belonging to $\mathscr{N}$ to obtain an $(n-k) \times (n-s)$ matrix $\mathbf{H}_{\mathscr{N}}$. The row-space of $\mathbf{H}_{\mathscr{N}}$ generates the restrictions $\mathbf{h}_{\mathscr{N}}$ to $\mathscr{N}$ of the parity-checks $\mathbf{h}$ of $\mathscr{C}$. This row-space is nothing but the dual code $\mathscr{C}^{\perp}$ punctured in $\mathscr{P}$, *i.e.* we keep only the positions in $\mathscr{N}$. With our notation, this is $\mathscr{C}^{\perp}_{\mathscr{N}}$ and is an $[n-s, n-k]$-code. Therefore if we want

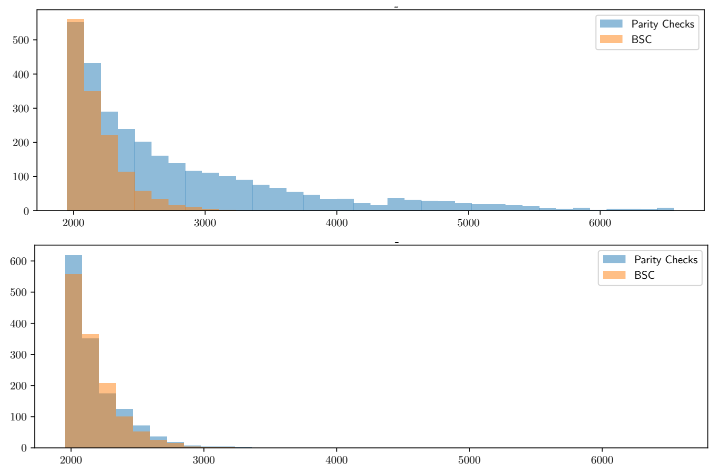


Fig. 3.1. Tails of the largest Fourier coefficients when decoding the $[N, s]$-code either with the noise produced by the $\langle \mathbf{h}_{\mathscr{N}}, \mathbf{e}_{\mathscr{N}} \rangle$'s or by the ideal LPN noise model (the $\text{BSC}(\frac{1-\varepsilon}{2})$ noise model). Both figures correspond to parity-checks $\mathbf{h}_{\mathscr{N}}$ of weight 6 and to $s = 19$. However they differ in the value for k. k equals 26 in the first figure and displays rather heavy tails for the largest Fourier coefficients corresponding to the parity-checks $\mathbf{h}_{\mathscr{N}}$ whereas $k = 40$ corresponds to rather similar tails in both cases. This is a general trend that can be verified on https://github.com/tillich/RLPNdecoding/tree/master/verification_heuristic/histogram, when k gets larger, the heavy tail phenomenon vanishes.

to find parity-checks $\mathbf{h}$ of $\mathscr{C}$ such that $|\mathbf{h}_{\mathscr{N}}| = w$, this amounts to find codewords of $\mathscr{C}_{\mathscr{N}}^{\perp}$ of weight w. For this, we compute a parity-check matrix $\mathbf{H}'$ of $\mathscr{C}_{\mathscr{N}}^{\perp}$ *i.e.* a $(k-s) \times (n-s)$ matrix such that $\mathscr{C}_{\mathscr{N}}^{\perp} = \{\mathbf{c} \in \mathbb{F}_2^{n-s} : \mathbf{H}'\mathbf{c}^{\intercal} = 0\}$. We split such a matrix in two parts randomly chosen and of the same size $\mathbf{H}' = \begin{pmatrix}\mathbf{H}_1 \; \mathbf{H}_2\end{pmatrix}$. We obtain an algorithm of time and space complexity, T and S respectively, producing N codewords of weight w, with $N = \frac{\binom{\frac{n-s}{2}}{\frac{w}{2}}^2}{2^{k-s}}(1 + o(1))$ and $\mathrm{S} = \mathrm{T} = O\left(\binom{\frac{n-s}{2}}{\frac{w}{2}} + N\right)$. The algorithm for producing such codewords is to set up two lists,

$$\mathscr{L}_1 \triangleq \left\{(\mathbf{H}_1\mathbf{h}_1^{\intercal}, \mathbf{h}_1) : |\mathbf{h}_1| = \frac{w}{2}\right\} \text{ and } \mathscr{L}_2 \triangleq \left\{(\mathbf{H}_2\mathbf{h}_2^{\intercal}, \mathbf{h}_2) : |\mathbf{h}_2| = \frac{w}{2}\right\}$$

and looking for collisions $\mathbf{H}_1\mathbf{h}_1^{\intercal} = \mathbf{H}_2\mathbf{h}_2^{\intercal}$ in the lists. It yields vectors $\mathbf{h}' = \mathbf{h}_1||\mathbf{h}_2$ of weight w which are in $\mathscr{C}_{\mathscr{N}}^{\perp}$ since $\mathbf{H}'\mathbf{h}'^{\intercal} = \mathbf{H}_1\mathbf{h}_1^{\intercal} + \mathbf{H}_2\mathbf{h}_2^{\intercal} = \mathbf{0}$. These vectors in $\mathbb{F}_2^{n-s}$ can be completed to give vectors $\mathbf{h} \in \mathbb{F}_2^n$ such that $\mathbf{h}_{\mathscr{N}} = \mathbf{h}'$. The number of collisions is expected to be of order $\binom{\frac{n-s}{2}}{\frac{w}{2}}^2 / 2^{k-s}$ since $2^{-(k-s)}$ is the collision probability of two vectors in $\mathbb{F}_2^{k-s}$. The algorithm for performing this task is given by Algorithm 4.1.

Algorithm 4.1. Creating low weight parity-checks by collisions

```
    Input $\mathscr{C}$, $w$, $\mathscr{P}$
    Output a list of parity-check equations $\mathbf{h}$ of $\mathscr{C}$ such that $|\mathbf{h}_{\mathscr{N}}| = w$ where $\mathscr{N} \triangleq [\![1,n]\!] \setminus \mathscr{P}$.
  function Create($\mathscr{C}$,$w$, $\mathscr{P}$)
      $\mathbf{H} \leftarrow$ Parity-check-matrix($\mathscr{C}^{\perp}, \mathscr{P}$)
              ▷ returns a parity-check matrix for $\mathscr{C}^{\perp}$ with an identity corresponding to the positions in $\mathscr{P}$: $\mathbf{H} = \begin{pmatrix} \mathbf{I} & \mathbf{P} \\ \mathbf{0} & \mathbf{H}' \end{pmatrix}$ where we assume that the first block corresponds to the positions of $\mathscr{P}$.
      $\mathscr{L}_1 \leftarrow \{(\mathbf{H}_1\mathbf{h}_1^{\intercal}, \mathbf{h}_1) : |\mathbf{h}_1| = w/2, \mathbf{h}_1 \in \mathbb{F}_2^{\frac{n-s}{2}}\}$
      $\mathscr{L}_2 \leftarrow \{(\mathbf{H}_2\mathbf{h}_2^{\intercal}, \mathbf{h}_2) : |\mathbf{h}_2| = w/2, \mathbf{h}_2 \in \mathbb{F}_2^{\frac{n-s}{2}}\}$
              ▷ We assume $\mathbf{H}' = \begin{pmatrix} \mathbf{H}_1 & \mathbf{H}_2 \end{pmatrix}$, with $\mathbf{H}_1$ and $\mathbf{H}_2$ of the same size.
      $\mathscr{L} \leftarrow \{\mathbf{h}_1 || \mathbf{h}_2 \in \mathscr{L}_1 \times \mathscr{L}_2 : \mathbf{H}_1\mathbf{h}_1^{\intercal} = \mathbf{H}_2\mathbf{h}_2^{\intercal}\}$
      return $\{\mathbf{h}'\mathbf{P}^{\intercal} || \mathbf{h}' : \mathbf{h}' \in \mathscr{L}\}$
              ▷ It is straightforward to check that $\mathbf{h}'\mathbf{P}^{\intercal} || \mathbf{h}'$ belongs to $\mathscr{C}^{\perp}$.
  end function
```

We have represented in Figure 4.1 the form of the parity-checks output by this method, together with the bet we make on the error.

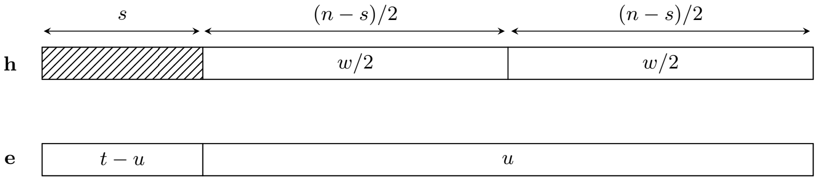


Fig. 4.1. The form of the parity-checks produced by this method, *vs.* the bet made on the error. The hatched rectangle of size s for $\mathbf{h}$ indicates that the weight is arbitrary on this part.

The amortized cost for producing a parity-check equation of weight w is $O(1)$ as long as $N \geq \Omega\left(\binom{\frac{n-s}{2}}{\frac{w}{2}}\right)$. It is insightful to consider the smallest value of w for which $\binom{\frac{n-s}{2}}{\frac{w}{2}} \leq \binom{\frac{n-s}{2}}{\frac{w}{2}}^2 / 2^{k-s}$. This is roughly speaking the smallest value (up to negligible terms) of w for which the amortized cost for producing parity-check equations of weight w is $O(1)$ per equation. In such a case, we roughly have $N \approx \binom{\frac{n-s}{2}}{\frac{w}{2}} \approx \frac{\binom{\frac{n-s}{2}}{\frac{w}{2}}^2}{2^{k-s}} \approx 2^{k-s}$. In other words with this choice we have $\mathrm{T}_{\mathrm{eq}} = O\big(2^{k-s}\big)$. Let us choose now u as the "typical error weight" when restricted to $\mathscr{N}$, namely $u \approx t\frac{n-s}{n}$ and s such that the decoding complexity of the $[N, s]$-code is also of order the codelength, *i.e.* $N = \widetilde{\Theta}\left(2^s\right)$. This would imply $2^s \approx 2^{k-s}$, which means that we are going to choose $s = \frac{k}{2}$. By using Proposition 3.10, all these choices would yield a time complexity $\mathrm{T}_{\mathrm{Dumer86}}$ for decoding $\mathscr{C}$ which would be of order

$$\mathrm{T}_{\mathrm{Dumer86}} = \widetilde{O}\left(2^{k/2}\right), \tag{4.1}$$

if the constraint $N = \widetilde{\Omega}\left(\left(\frac{\binom{n-s}{w}}{K_w^{n-s}(u)}\right)^2\right)$ for successful decoding the $[N, s]$-code is met. This amounts to $2^{Rn/2} = \widetilde{\Omega}\left(\left(\frac{\binom{n(1-R/2)}{w}}{K_w^{n(1-R/2)}(t(1-R/2))}\right)^2\right)$, where R is the code rate, *i.e.* $R = \frac{k}{n}$. By using Proposition 3.1, we can give an asymptotic formula for this constraint. It translates into $R/2 \geq 2(1-R/2)\,\widetilde{\delta}\left(\tau, \omega/(1-R/2)\right)$, where $\widetilde{\delta}$ is the function defined in Proposition 3.6. Amazingly enough this constraint is met up to very small values of R, it is only below $R \approx 0.02$ that this condition is not met anymore. This innocent looking remark has actually very concrete consequences. This means that above the range $R \gtrapprox 0.02$ the asymptotic complexity exponent, *i.e.* $\alpha_{\mathrm{Dumer86}} \overset{\triangle}{=} \limsup_n \log_2 \mathrm{T}_{\mathrm{Dumer86}}/n$ where $\mathrm{T}_{\mathrm{Dumer86}}$ is the time complexity, satisfies

$$\alpha_{\mathrm{Dumer86}} \leq \frac{R}{2}. \tag{4.2}$$

This is very surprising, since in the vicinity of $R \approx 0$ the asymptotic time complexity of *all known* decoding methods approach quickly R. In other words, in this regime, the complexity is of order $\mathrm{T} \approx 2^{Rn} = 2^k$ for full distance (*a.k.a.* GV) decoding, meaning that they are not better than exhaustive search. Unfortunately this is also the case for our method. It can namely be proved that even by optimizing on the value of s, w and u we can not do better than this with our method, since $\alpha_{\mathrm{Dumer86}}(R) \sim R$ as R approaches 0. However, as can be guessed from the fact that $\alpha_{\mathrm{Dumer86}} \leq \frac{R}{2}$ for $R \gtrapprox 0.02$, the behaviour of the complexity is much better for our RLPN decoder. This can be verified in Figure 4.2.

It is worthwhile to recall that ISD algorithms in the regime of the rate *close to* 1 precisely use this collision method to find low weight codewords in order to reduce significantly the complexity of decoding. In a sense, we have a dual version of the birthday/collision decoder of [10] with reduced complexity for rates close to 0.

4.2 Improving [10] by puncturing as in [11]

There is a simple way of improving the generation of dual codewords of low weight on $\mathscr{N}$. It consists in partitioning $\mathscr{N}$ in two sets $\mathscr{N}_1$ and $\mathscr{N}_2$ with $\mathscr{N}_2$ being a subset of positions of size just a little bit above $n-k$ (which is the dimension of the dual code $\mathscr{C}^{\perp}$), say $n-k+\ell$ and then to use the collision method to get dual codewords of weight w_2 on $\mathscr{N}_2$. The same method is used in the improvement

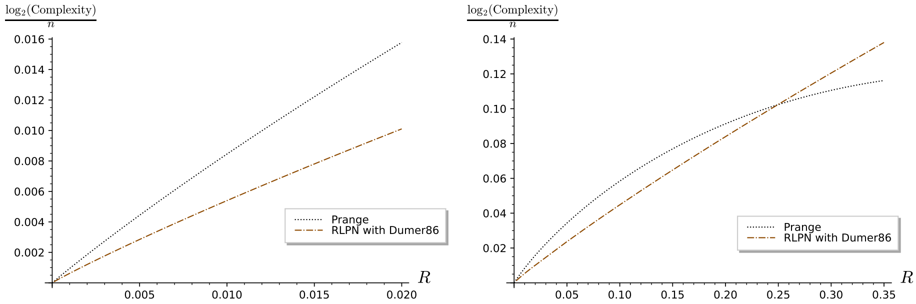


Fig. 4.2. The complexity of the RLPN-decoder for very small rates *vs.* the simplest information set decoder, namely the ISD Prange decoder [24]. For small R, there is no much difference between the ISD Prange decoder and much more evolved decoders like [3, 20, 5, 6]. The RLPN-decoder with the very simple [10] technique performs much better for small rates than ISD decoders. It is only outperformed by the Prange decoder for rates above 0.25 approximately.

[11] of the simple collision decoder [10] or in a slightly less efficient way in [26]. It just consists in finding codewords in $\mathscr{C}^\perp$ which have weight w_1 on $\mathscr{N}_1$ and w_2 on $\mathscr{N}_2$ instead of simply weight w on $\mathscr{N}$. We have represented in Figure 4.3 the form of the parity-checks we produce with this method. Note that the weight w_1 is expected to be half the size $k - \ell - s$ of $\mathscr{N}_1$.

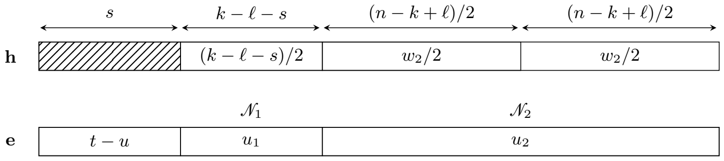


Fig. 4.3. The form of the parity-checks produced by this method, *vs.* the bet made on the error. The hatched rectangle of size s for $\mathbf{h}$ indicates that the weight is arbitrary on this part.

To understand the bias we get in this case, the proof of Proposition 3.1 can be readily adapted to yield

Proposition 4.1. *Assume that the code $\mathscr{C}$ is chosen by picking for it an $(n-k) \times n$ binary parity-check matrix uniformly at random. Let $\mathscr{N}$ be a fixed set of $n-s$ positions in $[\![1, n]\!]$ which is partitioned in two sets $\mathscr{N}_1$ and $\mathscr{N}_2$ and $\mathbf{e}$ be some error of weight u_i on $\mathscr{N}_i$ for $i \in \{1,2\}$. For $i \in \{1,2\}$, choose $\mathbf{h}$ uniformly at random among the parity-checks of $\mathscr{C}$ of weight w_i on the $\mathscr{N}_i$'s and $\mathbf{h}'$ uniformly at random among the words of weight w_i on the $\mathscr{N}_i$'s. For $i \in \{1,2\}$, let*

$$\delta_i \stackrel{\triangle}{=} \text{bias}\left(\langle \mathbf{e}_{\mathscr{N}_i}, \mathbf{h}'_{\mathscr{N}_i} \rangle\right) \text{ and } \delta \stackrel{\triangle}{=} \delta_1 \delta_2$$

If the parameters k, s, u_i, w_i are chosen as functions on n so that for n going to infinity, the expected number N of parity-checks of $\mathscr{C}$ of respective weight w_i on $\mathscr{N}_i$ for $i \in \{1,2\}$, satisfies $N = \omega\left(1/\delta^2\right)$ then for all but a proportion $o(1)$ of codes we have

$$\text{bias}\left(\langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle\right) = (1+o(1))\delta.$$

With the collision method we use, the parity-checks we produce have actually a slightly more specific form, since $\mathscr{N}_2$ is partitioned in two sets of (almost) the same size on which $\mathbf{h}$ has weight $w_2/2$. It is not difficult to turn such a generation of parity-checks at the cost of a polynomial overhead into a generation of uniformly distributed parity-checks of weight w_2 on $\mathscr{N}_2$. We leave out the details for doing this here. Under such an assumption, we have

Lemma 4.2. *With the same assumptions as in Proposition 4.1,*

$$\mathbb{P}_{\mathbf{h}}(\langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} = 1 \rangle) = \frac{1-\varepsilon}{2} \quad \textit{where} \quad \varepsilon = \delta_1\delta_2(1-o(1))$$

$\delta_1 \triangleq \frac{K_{w_1}^{k-\ell-s}(u_1)}{\binom{k-\ell-s}{w_1}}$, $\delta_2 \triangleq \frac{K_{w_2}^{n-k+\ell}(u_2)}{\binom{n-k+\ell}{w_2}}$, $u_1 \triangleq |\mathbf{e}_{\mathscr{N}_1}|$, $u_2 \triangleq |\mathbf{e}_{\mathscr{N}_2}|$, $w_1 \triangleq |\mathbf{h}_{\mathscr{N}_1}|$ *and* $w_2 \triangleq |\mathbf{h}_{\mathscr{N}_2}|$.

Proof. This is an application of the previous proposition and Lemma 3.3. □

All these considerations lead to a slight variation of the RLPN decoder given in Algorithm 3.1. Let us make now a bet on the weight u_i of the error restricted to $\mathscr{N}_i$ for $i \in \{1,2\}$ and use Dumer's [11] collision low-weight codeword generator to produce N parity-checks $\mathbf{h}$ such that $|\mathbf{h}_{\mathscr{N}_i}| = w_i$ for $i \in \{1,2\}$. We call the associated function $\textsc{Create}(N, w_1, w_2, \mathscr{P})$.

Proposition 4.3. *If Assumption 3.7 holds and assuming that* $\textsc{Create}(N, w_1, w_2, \mathscr{P})$ *produces N parity-check equations in space S_{eq} and time T_{eq} that are of weight w_i on $\mathscr{N}_i$ for $i \in \{1,2\}$. The probability P_{succ} (over the choice of $\mathscr{N}_1$ and $\mathscr{N}_2$) that there are exactly u_1 errors in $\mathscr{N}_1$ and u_2 errors in $\mathscr{N}_2$ is given by*

$$P_{succ} = \frac{\binom{s}{t-u_1-u_2}\binom{k-\ell-s}{u_1}\binom{n-k+\ell}{u_2}}{\binom{n}{t}}.$$

The space complexity $\mathrm{S}_{Dumer89}$ and time complexity $\mathrm{T}_{Dumer89}$ of the RLPN-decoder are given by

$$\textbf{Space: } \mathrm{S}_{Dumer89} = O(S_{eq} + 2^s), \quad \textbf{Time: } \mathrm{T}_{Dumer89} = \widetilde{O}\left(\frac{T_{eq} + 2^s}{P_{succ}}\right).$$

under the constraint on the parameters s, ℓ, u_1, u_2, w_1 and w_2 given by

$$N \leq 2^s \tag{4.3}$$

$$N \leq \frac{\binom{k-\ell-s}{w_1}\binom{n-k+\ell}{w_2}}{2^{k-s}} \tag{4.4}$$

$$N = \omega\left(\left(\frac{\binom{k-\ell-s}{w_1}\binom{n-k+\ell}{w_2}}{K_{w_1}^{k-\ell-s}(u_1)K_{w_2}^{k-\ell-s}(u_2)}\right)^2\right). \tag{4.5}$$

We have found out that choosing w_1 carefully is unnecessary and simply setting it to it its expected value is sufficient, *i.e.* $w_1 = \frac{k-\ell-s}{2}$. Again, the same discussion as in the previous section applies and if Conjecture 3.11 applies then the asymptotic form of the complexity is the same as if we use Proposition 4.3 and we get the following asymptotic form

Proposition 4.4. *If Conjecture 3.11 holds, the asymptotic complexity exponent of the RLPN decoder based on Dumer's collision low weight dual codeword generator is given by*

$$\alpha_{Dumer89}(R) \stackrel{\triangle}{=} \min_{(\sigma,\nu_1,\nu_2,\lambda,\omega_1,\omega_2)\in\mathscr{R}} \beta(R,\sigma,\nu_1,\nu_2,\lambda,\omega_1,\omega_2) \tag{4.6}$$

$$\beta \stackrel{\triangle}{=} \max(\sigma,\nu') + \pi,$$
$$\nu' \stackrel{\triangle}{=} \max\left(\tfrac{(1-R+\lambda)}{2} h\left(\tfrac{\omega_2}{1-R+\lambda}\right), \nu\right), \quad \nu \stackrel{\triangle}{=} (1-R+\lambda) h\left(\tfrac{\omega_2}{1-R+\lambda}\right) - \lambda,$$
$$\pi \stackrel{\triangle}{=} 1 - R - \sigma h\left(\tfrac{\tau-\nu_1-\nu_2}{\sigma}\right) - (R-\lambda-\sigma) h\left(\tfrac{\nu_1}{R-\lambda-\sigma}\right) - (1-R+\lambda) h\left(\tfrac{\nu_2}{1-R+\lambda}\right),$$
$$\tau \stackrel{\triangle}{=} \delta_{\mathrm{GV}}(R) = h^{-1}(1-R)$$

and the constraint region $\mathscr{R}$ is defined by the subregion of non-negative tuples $(\sigma,\nu_1,\nu_2,\lambda,\omega_1,\omega_2)$ such that $\omega_1 = \frac{R-\lambda-\sigma}{2}$ and

$$\sigma \le R-\lambda, \;\; \nu_1 \le R-\lambda-\sigma, \;\; \nu_2 \le 1-R+\lambda, \;\; \tau-\sigma \le \nu_1+\nu_2 \le \tau, \;\; \nu \le \sigma,$$
$$\nu = -(R-\lambda-\sigma)\widetilde{\delta}\left(\tfrac{\nu_1}{R-\lambda-\sigma}, \tfrac{\omega_1}{R-\lambda-\sigma}\right) - (1-R+\lambda)\widetilde{\delta}\left(\tfrac{\nu_2}{1-R+\lambda}, \tfrac{\omega_2}{1-R+\lambda}\right)$$

where $\widetilde{\delta}$ is the function defined in Proposition 3.6.

5 Using advanced collision techniques

ISD techniques have evolved [26, 11, 4, 19, 3] by first introducing [26] collision techniques whose purpose is to produce for codes of rate close to 1, all codewords of some small weight, and later on by substantially improving them by using on top of that for instance representation techniques [19]. These algorithms come very handy in our case for devising the function Create$(N, w, \mathscr{P})$ that we need. In the previous section, we have explored what could be achieved by the very first techniques of this type taken from [10, 11]. We are going to explain now what can be gained by using [19, 3]. It is convenient here to formalize the basic step used in the previous section which can be explained by the function of Algorithm 5.1. It creates codewords of weight w in a code of parity-check matrix $\mathbf{H}$ as sums $\mathbf{x}_1 + \mathbf{x}_2$ of two lists $\mathscr{L}_1$ and $\mathscr{L}_2$ with a complexity which is of the form $O\left(\max\left(\#\mathscr{L}_1, \#\mathscr{L}_2, \frac{\#\mathscr{L}_1 \cdot \#\mathscr{L}_2}{2^\ell}\right)\right)$ if the $\mathbf{H}\mathbf{x}_i^\intercal$'s are distributed uniformly at random and independently (we will make this assumption from now on). It is clear that [10] and [11] is more or less a direct application of this method. [19] and [3] use several layers of this function. [19] starts by partitioning the set of

Algorithm 5.1. Merging two lists for producing low weight codewords

```
    Input: $\mathscr{L}_1 \subseteq \mathbb{F}_2^n$, $\mathscr{L}_2 \subseteq \mathbb{F}_2^n$, $w \in [\![1, n]\!]$, $\mathbf{H} \in \mathbb{F}_2^{\ell \times n}$
    Output: a list $\mathscr{L} = \{\mathbf{x} = \mathbf{x}_1 + \mathbf{x}_2 : \mathbf{x}_i \in \mathscr{L}_i,\ i \in \{1,2\},\ |\mathbf{x}| = w,\ \mathbf{H}\mathbf{x}^\intercal = \mathbf{0}\}$ of
elements of the form $\mathbf{x}_1 + \mathbf{x}_2$ with $\mathbf{x}_i$ belonging to $\mathscr{L}_i$ of weight $w$ belonging to the
code of parity-check matrix $\mathbf{H}$
  function Merge($\mathscr{L}_1, \mathscr{L}_2, w, \mathbf{H}$)
      $\mathscr{L} \leftarrow \emptyset$
      for all $\mathbf{x}_1 \in \mathscr{L}_1$ do
          Store $\mathbf{x}_1$ in a hashtable $\mathscr{T}$ at address $\mathbf{H}\mathbf{x}_1^\intercal$
      end for
      for all $\mathbf{x}_2 \in \mathscr{L}_2$ do
          if $\exists \mathbf{x}_1$ in $\mathscr{T}$ at address $\mathbf{H}\mathbf{x}_2^\intercal$ and $|\mathbf{x}_1 + \mathbf{x}_2| = w$ then
              Put $\mathbf{x}_1 + \mathbf{x}_2$ in $\mathscr{L}$
          end if
      end for
      return $\mathscr{L}$
  end function
```

positions of the vectors of $\mathbb{F}_2^n$ which are considered in two sets $\mathscr{I}_1$ and $\mathscr{I}_2$ of about the same size. Then it starts with two lists $\mathscr{L}_1^0$ and $\mathscr{L}_2^0$ of all elements of weight p_0 and support $\mathscr{I}_1$ and $\mathscr{I}_2$ respectively. It merges them in a list $\mathscr{L}^1$ of elements of weight p_1 in the kernel of a parity-check matrix $\mathbf{H}_1$. Since the elements of $\mathscr{L}_1^0$ and $\mathscr{L}_2^0$ have disjoint supports by construction, we necessarily have that $p_1 = 2p_0$. List $\mathscr{L}^1$ is then merged with itself to yield elements which are in the kernel of another matrix $\mathbf{H}_2$ (see Figure 5.1). Since these are sums of elements of $\mathscr{L}^1$ they are also in the kernel of $\mathbf{H}_1$, so that the elements of the final list are of weight p_2 and belong to the code of parity-check $\mathbf{H} = \begin{pmatrix} \mathbf{H}_1 \\ \mathbf{H}_2 \end{pmatrix}$. The size of $\mathbf{H}_1$ is chosen such that an element $\mathbf{x}$ of weight p_2 and $\mathbf{H}_1\mathbf{x}^\intercal = \mathbf{0}$ is typically the sum of only two elements of $\mathscr{L}^1$ (this is the point of the representation technique). [3] is similar to [19] with one layer which is added. In this case, we create at the end a list of elements of weight p_3 which are in the code of parity-check matrix

$$\mathbf{H} \overset{\triangle}{=} \begin{pmatrix} \mathbf{H}_1 \\ \mathbf{H}_2 \\ \mathbf{H}_3 \end{pmatrix}. \tag{5.1}$$

The sizes of $\mathbf{H}_1$ in the [19] case, and of $\mathbf{H}_1$ and $\mathbf{H}_2$ in [3] are chosen to ensure unicity of the representation of an element of a list as the sum of two elements of the lists used for the merge (this is the representation technique).

We use these two techniques as we used the [10] technique inside the [11] technique, namely to generate codewords of $\mathscr{C}^\perp$ (*i.e.* $\mathbf{H}\mathbf{x}^\intercal = \mathbf{0}$ for $\mathbf{H}$ given by (5.1)) which are of weight p_3 on a set of indices of size $n - k + \ell$ (see Figure 5.2).

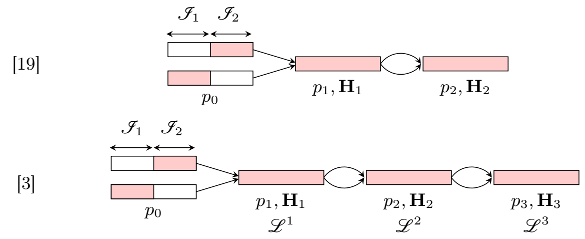


Fig. 5.1. This figure represents the successive lists obtained in [19] and [3]. The support of the elements of the list are represented in pink. Arrows point from the lists which are merged to the result of the merge and if two arrows depart from a list and arrive at another list, this means that the departure list is merged with itself. The weights of the elements are indicated for each level and the matrix $\mathbf{H}_i$ used for the merge is also given at the level of the result of the merge.

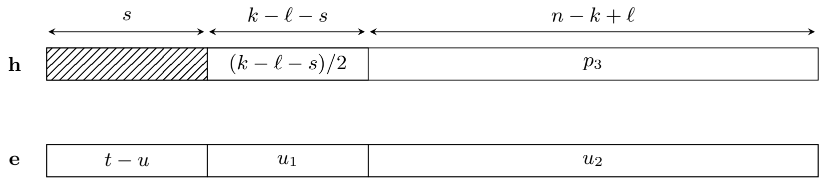


Fig. 5.2. The form of the parity-checks produced by this method [3], *vs.* the bet made on the error. The hatched rectangle of size s for $\mathbf{h}$ indicates that the weight is arbitrary on this part.

If we let ℓ_1 be the number of rows of $\mathbf{H}_1$, ℓ_2 be the number of rows of the matrix of $\mathbf{H}'_2 \stackrel{\triangle}{=} \begin{pmatrix} \mathbf{H}_1 \\ \mathbf{H}_2 \end{pmatrix}$, then the fact that the elements of $\mathscr{L}^2$ should have a unique representation in terms of a sum of a pair of elements of $\mathscr{L}^1$ respectively and that they are all elements $\mathbf{x}$ of weight p_1 and p_2 respectively which satisfy $\mathbf{H}'_2\mathbf{x}^\intercal = \mathbf{0}$ and $\mathbf{H}\mathbf{x}^\intercal = \mathbf{0}$ respectively, imposes conditions (5.2) which follow. The S_i represent the space complexity of the successive lists (*i.e.* $\mathscr{L}^0$, $\mathscr{L}^1$, $\mathscr{L}^2$ and $\mathscr{L}^3$) used in the [3] algorithm, whereas the T_i's denote the complexity of each merge and T_{eq} is the final complexity.

$$2^{\ell_1} = \binom{p_2}{p_2/2}\binom{n-k+\ell-p_2}{p_1-p_2/2}, \qquad 2^{\ell_2} = \binom{p_3}{p_3/2}\binom{n-k+\ell-p_3}{p_2-p_3/2} \tag{5.2}$$

$$S_0 = \binom{\frac{n-k+\ell}{2}}{\frac{p_1}{2}}, \quad S_1 = \frac{\binom{n-k+\ell}{p_1}}{2^{\ell_1}}, \quad S_2 = \frac{\binom{n-k+\ell}{p_2}}{2^{\ell_2}}, \quad S_3 = \frac{\binom{n-k+\ell}{p_3}}{2^{\ell}} \tag{5.3}$$

$$T_0 = S_0,\ T_1 = S_0 + \frac{S_0^2}{2^{\ell_1}},\ T_2 = S_1 + \frac{S_1^2}{2^{\ell_2-\ell_1}},\ T_3 = S_2 + \frac{S_2^2}{2^{\ell-\ell_2}},\ \mathrm{T}_{\mathrm{eq}} = \sum_i T_i \tag{5.4}$$

There is a similar proposition as Proposition 4.4 which gives the asymptotic complexity of the RLPN decoder used in conjunction with the [19] or [3] techniques for producing low weight codewords. For [19] it is given by

Proposition 5.1. *If conjecture 3.11 applies, the asymptotic complexity exponent of the RLPN decoder based on [19] is given by*

$$\alpha_{MMT}(R) \triangleq \min_{(\sigma,\nu_1,\nu_2,\lambda,\lambda_1,\omega_1,\omega_2,\pi_1)\in\mathscr{R}} \beta(R,\sigma,\nu_1,\nu_2,\lambda,\lambda_1,\omega_1,\omega_2,\pi_1) \tag{5.5}$$

$$\beta \triangleq \max(\sigma,\nu') + \pi, \quad \nu' \triangleq \max(\gamma_1,\gamma_2), \quad \nu \triangleq (1-R+\lambda)\,h\left(\tfrac{\omega_2}{1-R+\lambda}\right) - \lambda$$
$$\gamma_1 \triangleq \max\left(\tfrac{1-R+\lambda}{2} h\left(\tfrac{\pi_1}{1-R+\lambda}\right), (1-R+\lambda)h\left(\tfrac{\pi_1}{1-R+\lambda}\right) - \lambda_1\right),$$
$$\gamma_2 \triangleq 2(1-R+\lambda)h\left(\tfrac{\pi_1}{1-R+\lambda}\right) - \lambda_1 - \lambda,$$
$$\rho \triangleq 1 - R - \sigma h\left(\tfrac{\tau-\nu_1-\nu_2}{\sigma}\right) - (R-\lambda-\sigma)h\left(\tfrac{\nu_1}{R-\lambda-\sigma}\right) - (1-R+\lambda)h\left(\tfrac{\nu_2}{1-R+\lambda}\right),$$
$$\tau \triangleq \delta_{\mathrm{GV}}(R) = h^{-1}(1-R)$$

and the constraint region $\mathscr{R}$ *is defined by the subregion of non-negative tuples* $(\sigma,\nu_1,\nu_2,\lambda,\lambda_1,\pi,\omega_1,\omega_2)$ *such that*

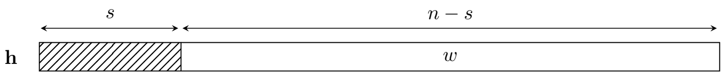
$$\sigma \le R-\lambda, \quad \lambda_1 \le \lambda, \quad \pi_1 \le \omega_2, \quad \nu_1 \le R-\lambda-\sigma, \quad \nu_2 \le 1-R+\lambda, \quad \nu \le \sigma,$$
$$\tau - \sigma \le \nu_1 + \nu_2 \le \tau, \quad \omega_1 = \tfrac{R-\lambda-\sigma}{2}, \quad \omega_2 < 1-R+\lambda,$$
$$\tfrac{\omega_2}{2} < \pi_1 < 1-R+\lambda, \quad \lambda_1 = \omega_2 + (1-R+\lambda-\omega_2)h\left(\tfrac{\pi_1-\omega_2/2}{1-R+\lambda-\omega_2}\right),$$
$$\nu = -(R-\lambda-\sigma)\widetilde{\delta}\left(\tfrac{\nu_1}{R-\lambda-\sigma}, \tfrac{\omega_1}{R-\lambda-\sigma}\right) - (1-R+\lambda)\widetilde{\delta}\left(\tfrac{\nu_2}{1-R+\lambda}, \tfrac{\omega_2}{1-R+\lambda}\right)$$

where $\widetilde{\delta}$ *is the function defined in Proposition 3.6.*

A proposition for the asymptotic behavior of RLPN decoding used together with [3] can be found in the full version of the paper [8, Ap. A]. We have used them for producing the complexity curves given in Figure 6.1 which display the various complexities of the RLPN decoders we have presented. Even if there is a tiny improvement by using [3] instead of [19] the two curves are nearly indistinguishable. A perspective of improvement of our algorithm could be to produce the parity-check equations by using more recent ISD techniques than [3], in particular [20, 5] or [6] which all use nearest-neighbor search. Our preliminary results using in particular [20] do not provide significant improvement, we have only been able to achieve a very slightly better complexity for rates close to 0.2.

6 A Lower bound on the complexity of RLPN decoders

As pointed out all along the paper, RLPN decoding needs a large number N of parity-check equations to work *but* of some shape as indicated below



where the hatched area indicates that the weight is arbitrary on this part while **h** restricted on the other positions needs to have Hamming weight w. The number N of such parity-checks has to verify (see Proposition 3.10)

$$N = \omega\left(n\left(\frac{\binom{n-s}{w}}{K_w^{n-s}(u)}\right)^2\right) \tag{6.1}$$

in order to be able to solve the underlying LPN problem. It can be verified that the smaller w is (the bigger is the bias ε), the smaller is N and the more efficient is our algorithm. Obviously if w is too small, there are not enough such parity-checks. It can be verified that the expected number of parity-checks of the aforementioned shape is given by $2^s\binom{n-s}{w}/2^k$ in a random code (which is our assumption). Therefore we need

$$N = O\left(\frac{2^s\binom{n-s}{w}}{2^k}\right). \tag{6.2}$$

Given this picture it is readily seen that the complexity of RLPN decoding is always lower-bounded by N (which is at least the cost to produce N parity-checks) but we can be more accurate on the lower-bound over the complexity. Recall that we first need to solve an underlying LPN problem and that we make a bet on the number of errors u in $\mathscr{N}$. Therefore, *assuming* that we can compute a parity-check of the aforementioned shape in time $O(1)$, the complexity of this genie-aided RLPN decoding is given by

$$\widetilde{O}\left(\frac{1}{\mathrm{P}_{\mathrm{succ}}}\max\left(2^s, N\right)\right) \tag{6.3}$$

where $\mathrm{P}_{\mathrm{succ}}$ is given in Proposition 3.10. Our only constraints are given by (6.1) and (6.2). By optimizing (6.3) over s, u and w, we can give a lower-bound on the complexity of RLPN decoding. However notice that our lower-bound applies to a partition of parity-checks in two parts (s and $n-s$). We do not consider here finer partitions. This method for lower bounding the complexity of RLPN decoding is very similar to the technique used in [9, §7] to lower bound the complexity of statistical decoding. All in all, we give in Figure 6.1 this lower-bound of the complexity. The optimal parameters computed for each RLPN algorithms can be found on https://github.com/tillich/RLPNdecoding. As we see our RLPN decoders meet this lower-bound for small rates and we can hope to outperform significantly ISD's for code rates smaller than ≈ 0.45.

7 Concluding remarks

Since Prange's seminal work [24] in 1962, ISD algorithms have played a predominant role for assessing the complexity of code-based cryptographic primitives.

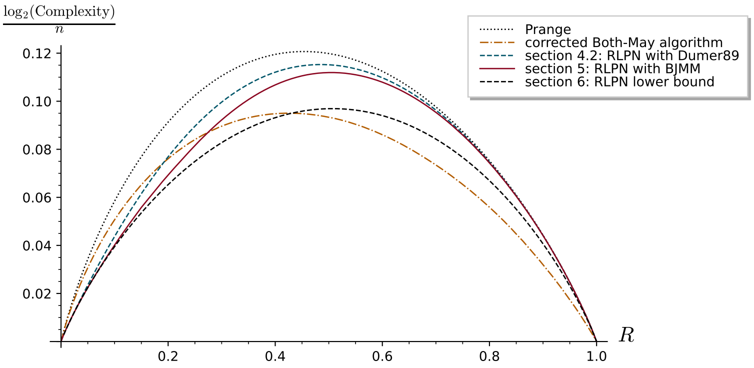


Fig. 6.1. Complexity exponents of our different RLPN decoders, ISD's and the genie-aided RLPN algorithm when splitting parity-checks in two parts.

In the fixed rate regime, they have been beaten only once in [10] with the help of collision techniques, and this only for a tiny code rate range ($R \in (0.98, 1)$) and for a short period of time [26, 11] until these collision techniques were merged with the collision techniques to yield modern ISD's. Surprisingly enough, these improved ISD have resulted in decoding complexity curves tilting more and more to the left (*i.e.* with a maximum which is attained more and more below $\frac{1}{2}$) instead of being symmetric around $\frac{1}{2}$ as it could have been expected. It is precisely for rates below $\frac{1}{2}$ that RLPN decoding is able to outperform the best ISD's. This seems to point to the fact that it is precisely for this regime of parameters that we should aim for improving them. Interestingly enough, even if there is some room of improvement for RLPN decoding by using better strategies for producing the needed low weight parity-checks, there is a ceiling that this technique can not break (at least if we just split the parity-checks in two parts) and which is extremely close at rate $R = 0.45$ to the best ISD algorithm [6]. The RLPN decoding algorithm presented here has not succeeded in changing the landscape for very tiny code rates (R going to 0), since the complexity exponent of RLPN decoding approaches the one of exhaustive search on codewords, but the speed at which this complexity approaches exhaustive search is much smaller than for ISD's in the full decoding regime (*i.e.* at the GV distance). The success of RLPN decoding for $R < 0.3$ could be traced back precisely to this behaviour close to 0. An interesting venue for research could be to try to explore if there are other decoding strategies that would be candidate for beating exhaustive search in the tiny code rate regime.

Note however that like dual attacks in lattice based cryptography, the success of this algorithm relies on assumptions of the noise model we get from the low weight parity-check equations we produce (which is similar to the vectors in the dual lattice of small norm we use for dual attacks). The strict LPN model for this noise (Assumption 3.7) has been found out not to be completely accurate for the large Fourier coefficients obtained during decoding the $[N, s]$-code with

Fourier techniques (see Subsection 3.4). However, a weaker conjecture, namely Conjecture 3.11, is enough for guaranteeing the success of this decoding method and is compatible with the experiments we have made. There is a rather clear path for verifying at least semi-heuristically this conjecture and this will be the object of further studies about this algorithm.

Acknowledgement. We would like to express our warmest gratitude to Elena Kirshanova for all her work and comments on an earlier version of this paper which helped us a great deal in improving the quality of this submission. We would also thank the Asiacrypt 22' reviewers for all their comments and their scientific rectitude. We thank Ilya Dumer for his very insightful thoughts about decoding linear codes in the low rate regime. The work of TDA was funded by the French Agence Nationale de la Recherche through ANR JCJC COLA (ANR-21-CE39-0011). The work of CMH was funded by the French Agence de l'innovation de Défense and by Inria.

References

1. Aguilar, C., Gaborit, P., and Schrek, J. A new zero-knowledge code based identification scheme with reduced communication. In *Proc. IEEE Inf. Theory Workshop- ITW 2011* (Oct. 2011), IEEE, pp. 648–652.
2. Barg, A. Complexity issues in coding theory. *Electronic Colloquium on Computational Complexity* (Oct. 1997).
3. Becker, A., Joux, A., May, A., and Meurer, A. Decoding random binary linear codes in $2^{n/20}$: How $1+1 = 0$ improves information set decoding. In *Advances in Cryptology - EUROCRYPT 2012* (2012), LNCS, Springer.
4. Bernstein, D. J., Lange, T., and Peters, C. Smaller decoding exponents: ball-collision decoding. In *Advances in Cryptology - CRYPTO 2011* (2011), vol. 6841 of *LNCS*, pp. 743–760.
5. Both, L., and May, A. Optimizing BJMM with Nearest Neighbors: Full Decoding in $2^{2/21n}$ and McEliece Security. In *WCC Workshop on Coding and Cryptography* (Sept. 2017).
6. Both, L., and May, A. Decoding linear codes with high error rate and its impact for LPN security. In *Post-Quantum Cryptography 2018* (Fort Lauderdale, FL, USA, Apr. 2018), T. Lange and R. Steinwandt, Eds., vol. 10786 of *LNCS*, Springer, pp. 25–46.
7. Canto-Torres, R., and Sendrier, N. Analysis of information set decoding for a sub-linear error weight. In *Post-Quantum Cryptography 2016* (Fukuoka, Japan, Feb. 2016), LNCS, pp. 144–161.
8. Carrier, K., Debris-Alazard, T., Meyer-Hilfiger, C., and Tillich, J. Statistical decoding 2.0: Reducing decoding to LPN. Cryptology ePrint Archive, Report 2022/1000, 2022.
9. Debris-Alazard, T., and Tillich, J.-P. Statistical decoding. preprint, Jan. 2017. arXiv:1701.07416.
10. Dumer, I. On syndrome decoding of linear codes. In *Proceedings of the 9th All-Union Symp. on Redundancy in Information Systems, abstracts of papers (in russian), Part 2* (Leningrad, 1986), pp. 157–159.

11. Dumer, I. Two decoding algorithms for linear codes. *Probl. Inf. Transm. 25*, 1 (1989), 17–23.

12. Dumer, I. On minimum distance decoding of linear codes. In *Proc. 5th Joint Soviet-Swedish Int. Workshop Inform. Theory* (Moscow, 1991), pp. 50–52.

13. Feneuil, T., Joux, A., and Rivain, M. Shared permutation for syndrome decoding: New zero-knowledge protocol and code-based signature. *IACR Cryptol. ePrint Arch.* (2021), 1576.

14. Finiasz, M., and Sendrier, N. Security bounds for the design of code-based cryptosystems. In *Advances in Cryptology - ASIACRYPT 2009* (2009), M. Matsui, Ed., vol. 5912 of *LNCS*, Springer, pp. 88–105.

15. Fossorier, M. P. C., Kobara, K., and Imai, H. Modeling bit flipping decoding based on nonorthogonal check sums with application to iterative decoding attack of McEliece cryptosystem. *IEEE Trans. Inform. Theory 53*, 1 (2007), 402–411.

16. Gaborit, P., and Girault, M. Lightweight code-based authentication and signature. In *Proc. IEEE Int. Symposium Inf. Theory - ISIT* (Nice, France, June 2007), pp. 191–195.

17. Jabri, A. A. A statistical decoding algorithm for general linear block codes. In *Cryptography and coding. Proceedings of the 8^{th} IMA International Conference* (Cirencester, UK, Dec. 2001), B. Honary, Ed., vol. 2260 of *LNCS*, Springer, pp. 1–8.

18. Kirshner, N., and Samorodnitsky, A. A moment ratio bound for polynomials and some extremal properties of krawchouk polynomials and hamming spheres. *IEEE Trans. Inform. Theory 67*, 6 (2021), 3509–3541.

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

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

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

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

23. Overbeck, R. Statistical decoding revisited. In *Information security and privacy : 11^{th} Australasian conference, ACISP 2006* (2006), R. S.-N. Lynn Batten, Ed., vol. 4058 of *LNCS*, Springer, pp. 283–294.

24. Prange, E. The use of information sets in decoding cyclic codes. *IRE Transactions on Information Theory 8*, 5 (1962), 5–9.

25. Richardson, T., and Urbanke, R. *Modern Coding Theory.* Cambridge University Press, 2008.

26. Stern, J. A method for finding codewords of small weight. In *Coding Theory and Applications* (1988), G. D. Cohen and J. Wolfmann, Eds., vol. 388 of *LNCS*, Springer, pp. 106–113.

27. Stern, J. A new identification scheme based on syndrome decoding. In *Advances in Cryptology - CRYPTO'93* (1993), D. Stinson, Ed., vol. 773 of *LNCS*, Springer, pp. 13–21.

28. Wagner, D. A generalized birthday problem. In *Advances in Cryptology - CRYPTO 2002* (2002), M. Yung, Ed., vol. 2442 of *LNCS*, Springer, pp. 288–303.