Post-Quantum Cryptography - Codes

Lecture 3: Information Set Decoding Algorithms

Lecturer: Thomas Debris-Alazard

Introduction

The aim of any code-based cryptosystem is to rely its security on the hardness of the decoding problem when the input code is random. It is therefore crucial to study the best algorithms, usually called *generic* decoding algorithms, for solving this problem. Despite many efforts on this issue the best ones [BJMM12, MO15, BM17] are exponential in the number of errors that have to be corrected and all can be viewed as a refinement of the original Prange's algorithm [Pra62]. They are actually referred to as *Information Set Decoding* (ISD). The aim of these lecture notes is to describe the "first" ISD algorithms. Our description will be mainly algorithmic with the use of parity-check matrices. However in this (too long) introduction we try to give another point of view, more related to the inherent mathematical structure of codes: linear subspaces of some $\mathbb{F}_q^n$.

Prange's approach: using linearity. Given an $[n,k]_q-$code $\mathcal{C}$ and one word $\mathbf{y} \in \mathbb{F}_q^n$, we are looking for some codeword $\mathbf{c} \in \mathcal{C}$ at distance t from $\mathbf{y}$, namely $|\mathbf{y}-\mathbf{c}| = t$. Here $\mathcal{C}$ is defined as a *linear* subspace of $\mathbb{F}_q^n$ of dimension k. Therefore one can check that there exists some set of positions $\mathscr{I} \subseteq [\![1,n]\!]$ of size k, *called an information set*, which uniquely determines every codewords, more precisely

$$\forall \mathbf{x} \in \mathbb{F}_q^k : \quad \exists! \mathbf{c} \in \mathcal{C} \text{ (that we can easily compute by linear algebra) such that } \mathbf{c}_{\mathscr{I}} = \mathbf{x}$$

where $\mathbf{c}_{\mathscr{I}}$ denotes the vector whose coordinates are those of $\mathbf{c} = (c_i)_{1\leq i\leq n}$ which are indexed by $\mathscr{I}$, *i.e.* $\mathbf{c}_{\mathscr{I}} = (c_i)_{i\in\mathscr{I}}$.

Exercise 1. *Let $\mathcal{C}$ be an $[n,k]$-code and $\mathscr{I} \subseteq [\![1,n]\!]$ be of size k. Show that,*

$$\begin{aligned} \mathscr{I} \text{ is an information set for } \mathcal{C} &\iff \forall \mathbf{G} \text{ generator matrix of } \mathcal{C},\ \mathbf{G}_{\mathscr{I}} \text{ is invertible} \\ &\iff \forall \mathbf{H} \text{ parity-check matrix of } \mathcal{C},\ \mathbf{H}_{\overline{\mathscr{I}}} \text{ is invertible} \end{aligned}$$

where given $\mathbf{M} \in \mathbb{F}_q^{r\times n}$, $\mathbf{M}_{\mathscr{I}}$ denotes the matrix whose columns *are those of $\mathbf{M}$ which are indexed by $\mathscr{I}$.*

Prange's idea to recover some solution $\mathbf{c}^{\text{sol}} \in \mathcal{C}$, where $\mathbf{y} = \mathbf{c}^{\text{sol}} + \mathbf{e}^{\text{sol}}$ and $|\mathbf{e}^{\text{sol}}| = t$, is as follows. First we pick some random information set $\mathscr{I}$ and we hope that it contains no error positions, namely:

$$\mathbf{c}^{\text{sol}}_{\mathscr{I}} = \mathbf{y}_{\mathscr{I}} \quad \left(\iff \mathbf{e}^{\text{sol}}_{\mathscr{I}} = \mathbf{0} \right). \tag{3.1}$$

If this is true, we are done. It remains to compute *the unique* codeword $\mathbf{c}$ such that $\mathbf{c}_{\mathscr{I}} = \mathbf{y}_{\mathscr{I}}$ as by uniqueness we get $\mathbf{c} = \mathbf{c}^{\text{sol}}$. In other words, Prange's idea (when looking for a close codeword) simply consists in picking some information set $\mathscr{I}$, computing the unique codeword $\mathbf{c}$ equal to $\mathbf{y}$ on those coordinates, and then to check if the constraint (3.1) is verified, namely if $|\mathbf{y}-\mathbf{c}| = t$. The average number of times we have to pick a set $\mathscr{I}$ in Prange's algorithm until finding a solution is therefore given by $1/p_{\text{pr}}$ where p_{pr} is the probability that Equation (3.1) is verified. As we will see, $1/p_{\text{pr}}$ is

- polynomial for $t/n = \left(\frac{q-1}{q}\right)\left(1-\frac{k}{n}\right)$,
- exponential in t when $t/n \in \left]0, \left(\frac{q-1}{q}\right)\left(1-\frac{k}{n}\right)\right[$

as long as $n \to +\infty$ and k/n is some constant. Interestingly, all improvements of Prange's algorithm, since sixty years, have the same behaviour with respect to t/n. Even though Prange's algorithm is quite naive, it really shows where decoding is easy, where it is not.

Let us now describe how these improvements, in particular ISD algorithms, were obtained as they start from the same key idea. For this let us back up a bit.

Dumer's approach: a collision search. The simplest way to find a codeword $\mathbf{c}$ at distance t from $\mathbf{y}$ is basically enumerating all the errors $\mathbf{e}$ of weight t until finding one that reaches $\mathbf{y} - \mathbf{e} \in \mathcal{C}$. This naive approach will obviously cost $\binom{n}{t}(q-1)^t$. However, taking advantage of the birthday paradox, this exhaustive enumeration can be improved as Dumer showed [Dum86]. Dumer's idea was to notice that if one splits in two parts [(1)] of the same size some parity-check matrix $\mathbf{H} = \begin{pmatrix}\mathbf{H}_1 & \mathbf{H}_2\end{pmatrix}$ of $\mathcal{C}$, then solving the decoding problem boils down to finding $\mathbf{e}_1$ and $\mathbf{e}_2$ of Hamming weight $t/2$ such that $\mathbf{H}_1\mathbf{e}_1^\intercal + \mathbf{H}_2\mathbf{e}_2^\intercal = \mathbf{H}\mathbf{y}^\intercal$. A natural strategy to compute a solution $\mathbf{e} = (\mathbf{e}_1, \mathbf{e}_2)$ reduces to compute the following lists of $\mathbb{F}_q^{n-k}$

$$\mathscr{L}_1 \stackrel{\text{def}}{=} \left\{\mathbf{H}_1\mathbf{e}_1^\intercal : |\mathbf{e}_1| = \frac{t}{2}\right\} \quad \text{and} \quad \mathscr{L}_2 \stackrel{\text{def}}{=} \left\{-\mathbf{H}_2\mathbf{e}_2^\intercal + \mathbf{H}\mathbf{y}^\intercal : |\mathbf{e}_2| = \frac{t}{2}\right\}$$

and then to compute their "collision"

$$\mathscr{L}_1 \bowtie \mathscr{L}_2 \stackrel{\text{def}}{=} \left\{(\mathbf{e}_1, \mathbf{e}_2) \in \mathscr{L}_1 \times \mathscr{L}_2, \quad \mathbf{H}_1\mathbf{e}_1^\intercal = -\mathbf{H}_2\mathbf{e}_2^\intercal + \mathbf{H}\mathbf{y}^\intercal\right\}.$$

This new list trivially leads to solutions of the decoding problem. However, what is the cost of this procedure? By using classical techniques such as hash tables or sorting lists, computing $\mathscr{L}_1 \bowtie \mathscr{L}_2$ costs, up to a polynomial factor, $\max(\sharp\mathscr{L}_1, \sharp\mathscr{L}_2) + \sharp(\mathscr{L}_1 \bowtie \mathscr{L}_2)$. Notice now that both lists $\mathscr{L}_1$ and $\mathscr{L}_2$ have the same size, namely

$$\binom{n/2}{t/2}(q-1)^{t/2} = \widetilde{O}\left(\sqrt{\binom{n}{t}(q-1)^t}\right).$$

To estimate the cost of this procedure it remains to estimate the size of $\mathscr{L}_1 \bowtie \mathscr{L}_2$. One can check that Dumer's approach finds all the solutions of the decoding problem $\left(\max\left(1, \binom{n}{t}(q-1)^t/q^{n-k}\right)\right.$ as shown in lecture notes 2) but up to some polynomial loss, given by the probability that a solution $\mathbf{e}$ is not split into two equal parts. Then, $\mathscr{L}_1 \bowtie \mathscr{L}_2$ is the set of solution(s) of the considered decoding problem. To summarize, Dumer's approach enables to roughly find (up to polynomial factors)

$$\max\left(1, \frac{\binom{n}{t}(q-1)^t}{q^{n-k}}\right) \quad \text{solutions in time} \quad \sqrt{\binom{n}{t}(q-1)^t} + \frac{\binom{n}{t}(q-1)^t}{q^{n-k}}. \tag{3.2}$$

Notice in the case where t is equal to the Gilbert-Varshamov distance, namely when $\binom{n}{t}(q-1)^t \approx q^{n-k}$, Dumer's algorithm has a quadratic gain compared to the exhaustive search. However, it is even better, as shown by the following proposition.

Proposition 1. *The running time of Prange's algorithm for solving* $\mathsf{DP}(n, q, R, \tau)$ *when* $\tau = h^{-1}(1-R)$ [(2)] *and* $R \to 1$ *is given by:*

$$q^{n\,(1-R)(1+o(1))}$$

while Dumer's algorithm will cost:

$$q^{n\,\frac{1-R}{2}(1+o(1))}.$$

[(1)] To simplify the presentation, the cut is explained by taking the first $\frac{n-k}{2}$ positions for the first part and the other $\frac{n-k}{2}$ else for the second part, but of course in general these two sets of positions are randomly chosen.

[(2)] The relative Gilbert-Varshamov distance.

Dumer's algorithm has therefore a quadratic gain over Prange when the code rate tends to one and decoding at the Gilbert-Varhsamov distance. Though, the primary interest of this approach is not here. First, Dumer's algorithm finds (almost) all solutions of the decoding problem even if there are many of them. Furthermore, the distance t can be chosen such that it finds (almost) all of them in *amortized time one*.

Definition 1 (Amortized time one)**.** *An algorithm that outputs S solutions in time T of some problem is said to be in amortized time one if $S = \frac{T}{P(n)}$ for some polynomial P. In the sequel we will always neglect this polynomial factor.*

Dumer's algorithm works in amortized time one when t is beyond the Gilbert-Varshamov bound and verifies:

$$\sqrt{\binom{n}{t}(q-1)^t} = \frac{\binom{n}{t}(q-1)^t}{q^{n-k}} \iff \binom{n}{t}(q-1)^t = \left(q^{n-k}\right)^2. \tag{3.3}$$

As we are going to explain, most of the ideas to improve Prange's algorithm were based on these two remarks. The key idea is to reduce the initial decoding problem to a "denser" decoding problem where there are an exponential number of solutions but which can be found in amortized time one.

A mixed approach: ISD. The key point to improve Prange's algorithm starts from the following idea. Given some set of positions $\mathscr{J} \subseteq [\![1, n]\!]$ of size $k+\ell$ where $\ell > 0$, compute first a set $\mathscr{S}$ of *decoding candidates* which are some vectors at distance p from the target $\mathbf{y}$ when their coordinates are restricted to $\mathscr{J}$, namely:

$$\mathscr{S} \subseteq \left\{\mathbf{c}_{\mathscr{J}} : |\mathbf{c}_{\mathscr{J}} - \mathbf{y}_{\mathscr{J}}| = p \text{ and } \mathbf{c} \in \mathcal{C}\right\}. \tag{3.4}$$

Notice that $\mathscr{S}$ is a subset of the solutions of a decoding problem at distance p when it is given as input the target $\mathbf{y}_{\mathscr{J}}$ and the code

$$\mathscr{D} \stackrel{\text{def}}{=} \left\{\mathbf{c}_{\mathscr{J}} \in \mathbb{F}_q^{k+\ell} : \mathbf{c} \in \mathcal{C}\right\}. \tag{3.5}$$

It turns out that $\mathscr{D}$ is a code known as the *punctured* code of $\mathcal{C}$ at the positions $\overline{\mathscr{J}}$. Its length is $k+\ell$ and its dimension is k if $\mathscr{J}$ is an *augmented information set*, namely it contains some information set of $\mathcal{C}$, which will be assumed in what follows. Under this condition, $\mathbf{c}_{\mathscr{J}}$ uniquely determines its "lift" $\mathbf{c} \in \mathcal{C}$ which can be easily computed by linear algebra.

Exercise 2. *Let $\mathcal{C}$ be an $[n, k]$-code and $\mathscr{J} \subseteq [\![1, n]\!]$ be of size $k + \ell$. Show that,*

$$\mathscr{J} \text{ is an augmented information set for } \mathcal{C} \iff \mathscr{D} \text{ defined in Equation (3.5) has dimension } k.$$

Now, for the codeword $\mathbf{c} \in \mathcal{C}$, such that $\mathbf{c}_{\mathscr{J}} \in \mathscr{S}$, to be a solution of the original decoding problem, it has necessarily to verify

$$\left|\mathbf{c}_{\overline{\mathscr{J}}} - \mathbf{y}_{\overline{\mathscr{J}}}\right| = t - p. \tag{3.6}$$

This condition is weaker than of Prange algorithm (see Equation (3.1)): by picking our set $\mathscr{J}$ we do not hope to remove all the errors but only some fraction of it. Furthermore, contrary to Prange's approach we have many decoding candidates for each draw of the augmented information set $\mathscr{J}$. However, notice that smaller is p, harder it will be to compute even one decoding candidate. Therefore we cannot reasonably hope to choose p too small if we seek to test many decoding candidates at each draw of $\mathscr{J}$. It also turns out that if p is too small (below the Gilbert-Varshamov bound of the punctured code $\mathscr{D}$) no solutions are expected while on the other hand, if p is just above the Gilbert-Varshamov distance, we expect an exponential number of solutions.

So all in all, we have reduced our problem to decode a code of length n and dimension k to the bet made in (3.6) *and* the computation of $\mathscr{S}$ (*i.e.* the decoding candidates) which is nothing else than decoding a

"sub"-code of length $k+\ell$ and dimension k. This whole approach is known as Information Set Decoding (ISD). Note that we are completely free to choose our favourite algorithm to compute $\mathscr{S}$. Each ISD is then "parametrized" by the algorithm used as a subroutine for computing this set and, the better the algorithm, the better the ISD. However one may ask our meaning of a "better" algorithm for computing $\mathscr{S}$. To understand this let us introduce the probability $\alpha_{p,\ell}$ that a fixed $\mathbf{c}_{\mathscr{J}} \in \mathscr{S}$ leads to $\mathbf{c} \in \mathcal{C}$ which verifies Equation (3.6). We will show that the overall probability (after computing $\mathscr{S}$) to get a solution is given by $\min(1, \sharp\mathscr{S}\ \alpha_{p,\ell})$. It will lead to the following proposition that gives the running time of the whole algorithm to solve DP[3].

Proposition 2. *Assume that, given a random code $\mathscr{D}$ of length $k+\ell$, dimension k and a target $\mathbf{z} \in \mathbb{F}_q^{k+\ell}$, we can compute in time T a set of size S of codewords $\mathbf{d} \in \mathscr{D}$ at distance p from $\mathbf{z}$. Then, we can solve $\mathsf{DP}(n, q, R, \tau)$ in average time (up to a polynomial factor in n)*

$$T\ \max\left(1, \frac{1}{S\ \alpha_{p,\ell}}\right). \tag{3.7}$$

The overall cost for solving DP is therefore crucially parametrized by the cost for decoding a code $\mathscr{D}$ of rate $k/(k+\ell)$ at distance p, but notice that we need to find S solutions in time T and a priori not only one. If we want to design algorithms achieving this task such that the ISD improves original Prange's algorithm we have first to understand how parameters p, ℓ and quantities T, S interact.

Let us admit that $p \mapsto \alpha_{p,\ell}$ is a decreasing function. Notice now that, the larger p, the easier the decoding of $\mathscr{D}$ at distance p. Therefore we can reasonably suppose that $p \mapsto T$ is also a decreasing function. These two facts lead to a contradictory situation to minimize the ISD cost, we need to choose p as small as possible for minimizing $1/\alpha_{p,\ell}$ while at the same time we need to choose a large p to decrease T. Notice now, as $T \geq S$, that we have

$$T\ \max\left(1, \frac{1}{S\ \alpha_{p,\ell}}\right) \geq \frac{1}{\alpha_{p,\ell}}$$

Therefore we do not really have the choice, to minimize the cost of the ISD we have in the best case to design a sub-routine such that for parameters p and ℓ we have above an equality instead of an inequality. In particular it shows that our decoding algorithm at distance p (as small as possible) needs to find solutions in amortized time one, *i.e.* $S = T$. If this can be done we would get an improvement over Prange. Indeed, we have to remember that $\alpha_{p,\ell}$, the probability that our decoding candidate verifies Equation (3.6), is exponentially larger than the probability to verify Equation 3.1 as in Prange's algorithm (our bet is weaker).

Our discussion has just shown that it is theoretically possible to improve Prange's algorithm if we succeed, given a code $\mathscr{D}$ of length $k+\ell$, dimension k and any target $\mathbf{z}$, to compute in amortized time one many codewords $\mathbf{d} \in \mathscr{D}$ at distance p (as small as possible) from $\mathbf{z}$. The fundamental remark here is that $\mathscr{D}$ has a rate given by $k/(k+\ell) \approx 1$ when ℓ is not too large. It corresponds exactly to the range of parameters where Dumer's algorithm (that we have described earlier) can decode in amortized time one and can also have a quadratic gain over the original Prange algorithm. However parameters p and ℓ have to be carefully chosen as in Equation (3.3) (where replace t by p and n by $k+\ell$). In particular p cannot be chosen too small. Even though this choice of parameters is extremely constrained, the ISD using Dumer's algorithm improves Prange algorithm. But the better was yet to come. More sophisticated algorithms were designed, enabling to change the balance of parameters between p and ℓ by still decoding in amortized time one (in particular decreasing $p/(k+\ell)$ but also increasing ℓ to move away the rate $k/(k+\ell)$ from one). In these lecture notes we will restrict our study to the improvement given by the generalized birthday algorithm [Wag02]. But nowadays there exists far better techniques, such as "representations technique" (originally used for solving subset-sum problems) [BJMM12] or nearest neighbours search [MO15, BM17] but this is out of scope of

[3] The following proposition is the equivalent of Proposition 8 with the "noisy codeword" point of view.

these lecture notes.

Basic notation. Given $\mathbf{H} \in \mathbb{F}_q^{r \times n}$ and $\mathscr{I} \subseteq [\![1, n]\!]$ we will denote by $\mathbf{H}_{\mathscr{I}}$ the matrix whose *columns* are those of $\mathbf{H}$ which are indexed by $\mathscr{I}$.

During all these lecture notes both $R \in (0, 1)$ and the field size q will be supposed to be constants.

Described algorithms to solve DP. In these lecture notes we will describe three ISD algorithms to solve $\mathsf{DP}(n, q, R, \tau)$ (problem 5 in lecture notes 1). In each case we will show that their running time (over the input distribution) is of the form $2^{n\,\alpha(n,q,R,\tau)(1+o(1))}$. For all of them (and all known algorithms), their exponent $\alpha(n, q, R, \tau)$ is > 0 as long as τ does not belong to $\left[\frac{q-1}{q}(1-R), R + \frac{q-1}{q}(1-R)\right]$ as roughly described in Figure 3.1. Our aim during this lecture will be to compute the exponents of the three described algorithms. We draw them in Figures 3.2, 3.2 and 3.4 as function of τ for some rates R and field sizes q.

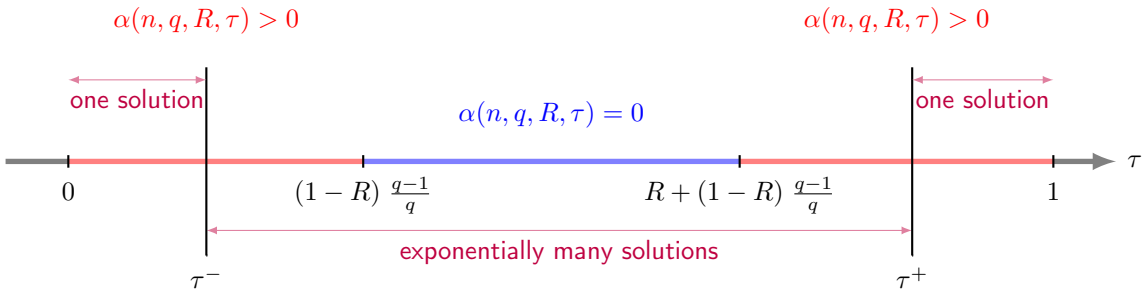


FIGURE 3.1. Exponents of the best generic decoding algorithms and expected number of solutions of $\mathsf{DP}(n, q, R, \tau)$ as function of τ.

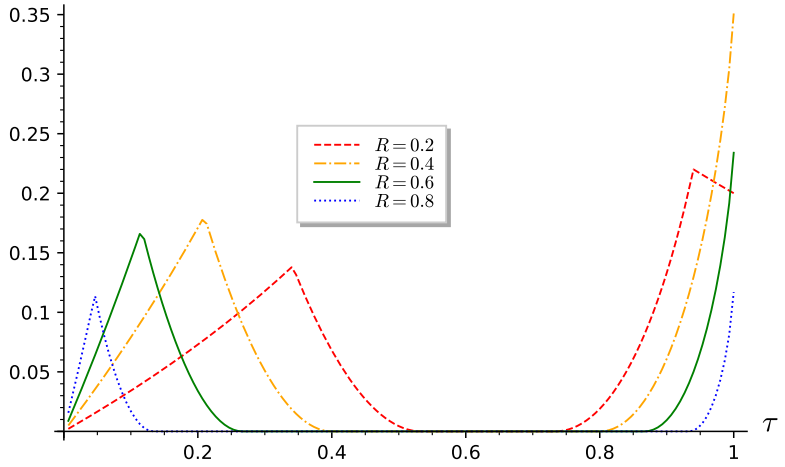


FIGURE 3.2. Exponent of Prange's algorithm (in base 2) to solve $\mathsf{DP}(n, q, R, \tau)$ for $q = 3$ and different rates R as function of τ.

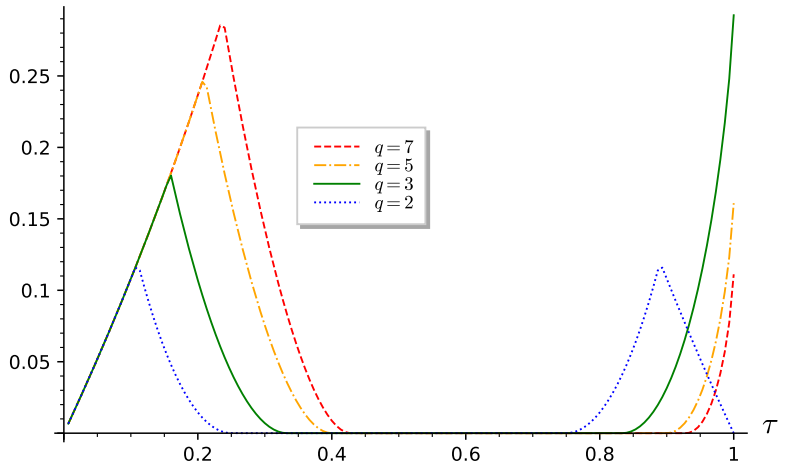


FIGURE 3.3. Exponent of Prange's algorithm (in base 2) to solve $\mathsf{DP}(n, q, R, \tau)$ for $R = 0.5$ and different field sizes q as function of τ.

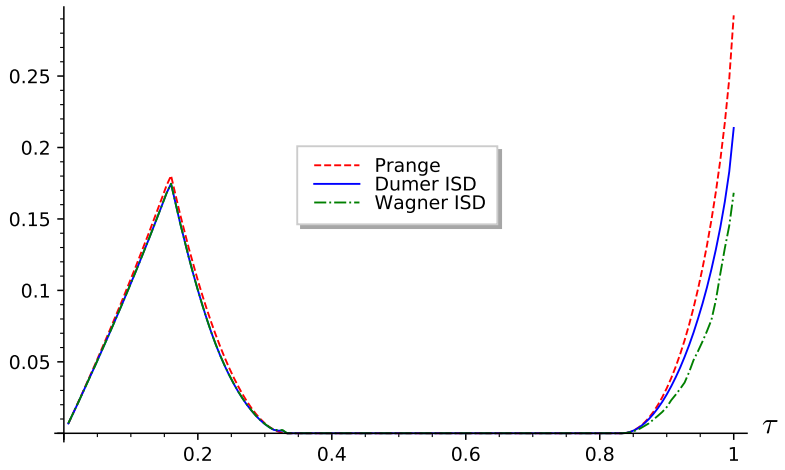


FIGURE 3.4. Exponents of Prange's algorithm and ISD with Dumer and Wagner' algorithms (in base 2) to solve $\mathsf{DP}(n, q, R, \tau)$ for $R = 0.5$ and $q = 3$ as function of τ.

1. Prange Algorithm

From now on, let us fix some instance $(\mathbf{H}, \mathbf{s}) \in \mathbb{F}_q^{(n-k)\times n} \times \mathbb{F}_q^{n-k}$ of the decoding problem $\mathsf{DP}(n, q, R, \tau)$ where recall that $k = Rn$. Our aim is to find $\mathbf{e} \in \mathbb{F}_q^n$ of Hamming weight $t = \tau n$ such that $\mathbf{e}\mathbf{H}^\intercal = \mathbf{s}$ and we know that by definition there is at least one solution.

It corresponds to solving a linear system with $n-k$ equations and n unknowns with a constraint on the Hamming weight of the solution. Prange's idea simply consists in "fixing" k unknowns and then solving a square linear system of size $(n-k)\times(n-k)$ hoping that the solution will have the correct Hamming weight; if not, we just repeat by fixing k other unknowns[4]. More precisely, Prange's algorithm is as follows. Let us first introduce the following distribution $\mathscr{D}_t$ over vectors of $\mathbb{F}_q^k$, for reasons that will be clear in the sequel.

The distribution $\mathscr{D}_t$.

- If $t < \frac{q-1}{q}(n-k)$, $\mathscr{D}_t$ only outputs $\mathbf{0} \in \mathbb{F}_q^k$,
- if $t \in [\![\frac{q-1}{q}(n-k), k+\frac{q-1}{q}(n-k)]\!]$, $\mathscr{D}_t$ outputs uniform vectors of weight $t-\frac{q-1}{q}(n-k)$,
- if $t > k+\frac{q-1}{q}(n-k)$, $\mathscr{D}_t$ outputs uniform vectors of weight k.

The algorithm.

1. *Picking the information set.* Let $\mathscr{I} \subseteq [\![1,n]\!]$ be a random set of size k. If $\mathbf{H}_{\overline{\mathscr{I}}} \in \mathbb{F}_q^{(n-k)\times(n-k)}$ is not of full-rank, pick another set $\mathscr{I}$.
2. *Linear algebra.* Perform a Gaussian elimination to compute a non-singular matrix $\mathbf{S} \in \mathbb{F}_q^{(n-k)\times(n-k)}$ such that $\mathbf{S}\mathbf{H}_{\overline{\mathscr{I}}} = \mathbf{1}_{n-k}$.
3. *Test Step.* Pick $\mathbf{x} \in \mathbb{F}_q^k$ according to the distribution $\mathscr{D}_t$ and let $\mathbf{e} \in \mathbb{F}_q^n$ be such that

$$\mathbf{e}_{\overline{\mathscr{I}}} = \left(\mathbf{s} - \mathbf{x}\mathbf{H}_{\mathscr{I}}^{\intercal}\right)\mathbf{S}^{\intercal} \quad ; \quad \mathbf{e}_{\mathscr{I}} = \mathbf{x}. \tag{3.8}$$

   If $|\mathbf{e}| \neq t$ go back to Step 1, otherwise it is a solution.

Correction of the algorithm. It easily follows from the following computation,

$$\begin{aligned}
\mathbf{S}\mathbf{H}\mathbf{e}^{\intercal} &= \mathbf{S}\mathbf{H}_{\overline{\mathscr{I}}}\,\mathbf{e}_{\overline{\mathscr{I}}}^{\intercal} + \mathbf{S}\mathbf{H}_{\mathscr{I}}\,\mathbf{e}_{\mathscr{I}}^{\intercal} \\
&= \mathbf{e}_{\overline{\mathscr{I}}}^{\intercal} + \mathbf{S}\mathbf{H}_{\mathscr{I}}\,\mathbf{e}_{\mathscr{I}}^{\intercal} \qquad \text{(by definition } \mathbf{S}\mathbf{H}_{\overline{\mathscr{I}}} = \mathbf{1}_{n-k}) \\
&= \mathbf{S}\left(\mathbf{s}^{\intercal} - \mathbf{H}_{\mathscr{I}}\,\mathbf{x}^{\intercal}\right) + \mathbf{S}\mathbf{H}_{\mathscr{I}}\,\mathbf{x}^{\intercal} \qquad \text{(by Equation (3.8))} \\
&= \mathbf{S}\mathbf{s}^{\intercal}
\end{aligned}$$

which corresponds to $\mathbf{H}\mathbf{e}^{\intercal} = \mathbf{s}^{\intercal}$ as $\mathbf{S}$ is non-singular. Furthermore the end of Step 3 is here to ensure that $\mathbf{e}$ will have the correct Hamming weight once the algorithm terminates.

Remark 1. *Let $\mathbf{y} \in \mathbb{F}_q^n$ be such that $\mathbf{y}\mathbf{H}^{\intercal} = \mathbf{s}$. Notice that $\mathbf{y} - \mathbf{e} = \mathbf{c} \in \mathcal{C}$ and by definition of $\mathbf{e}$ we have $\mathbf{c}_{\mathscr{I}} = \mathbf{y}_{\mathscr{I}}$ when $\mathbf{x} = \mathbf{0}$. In other words, when $\mathbf{x} = \mathbf{0}$, we recover the interpretation of Prange's algorithm to find a close codeword given in introduction.*

Exercise 3. *Describe Prange's algorithm with the generator matrix formalism in the same fashion as above (with also three steps and the distribution $\mathscr{D}_t$).*

Far or close codeword? One may ask why did we pick some vector $\mathbf{x}$ in Step 3 of the algorithm? Notice that it corresponds to fixing k unknowns to the value that we want. Suppose now that we would like to find a solution $\mathbf{e}$ of small Hamming weight. Obviously fixing $\mathbf{x}$ to be a non-zero vector is both needless and counterproductive as it would increase the weight of the decoding candidate. It is therefore better to choose $\mathbf{x}$ as $\mathbf{0}$ if we seek a solution of small Hamming weight. But now what happens if someone is looking for an error $\mathbf{e}$ of large Hamming weight, let us say close to n? The exact opposite: we need to choose $\mathbf{x}$ as a non-zero Hamming weight vector to increase the weight of the potential solution and therefore improving our success probability.

[4] Another interpretation of Prange's algorithm.

In summary, the vector $\mathbf{x}$ that we pick relies on what we want to do, finding a "short" or a "large" solution. The distribution $\mathscr{D}_t$, upon which $\mathbf{x}$ is picked, is precisely chosen according to the aforementioned aim. Equivalently, if one takes the generator matrix point of view, the distribution $\mathscr{D}_t$ enables to find a close or a far away codeword from a given target.

Rough analysis of the algorithm. Before giving a precise analysis of the running time of Prange's algorithm let us start by a rough analysis about what we "expect". First, let us assume that $\mathbf{s}$ *is uniformly distributed over* $\mathbb{F}_q^{n-k}$. Notice that it is, according to Proposition 6 in lecture notes 2, equivalent to assuming that t/n belongs to $[\tau^-, \tau^+]$. Therefore, according to Equation (3.8) the expected Hamming weight of the decoding candidate $\mathbf{e}$ is given by ($\mathbf{S}$ is non-singular hence it keeps invariant the uniform distribution of $\mathbf{s}$)

$$\mathbb{E}\left(|\mathbf{e}|\right) = |\mathbf{x}| + \frac{q-1}{q}\,(n-k).$$

By choosing $\mathbf{x} = \mathbf{0}$ we expect $\mathbf{e}$ to have a Hamming weight equal to $\frac{q-1}{q}\,(n-k)$. In other words, if one seeks a solution of DP with the aforementioned weight, its probability of success (in Step 3) is roughly $p_{\mathrm{pr}} \approx 1$ and the number of repetitions of the whole algorithm will be given by $1/p_{\mathrm{pr}} \approx 1$. On the other hand, if one wants a weight smaller than $(1-\varepsilon)\,\frac{q-1}{q}\,(n-k)$ or larger than $(1+\varepsilon)\,\frac{q-1}{q}\,(n-k)$, its probability of success will be exponentially small in $\varepsilon\,(n-k)$ since $\mathbf{s}$ is uniformly distributed. In that case we will need to repeat the three steps an exponential number of times before succeeding. However we can turn the above strategy into a stronger one: by carefully choosing $|\mathbf{x}| \in [\![0, k]\!]$ (recall that $\mathbf{x}$ is a vector of length k, the co-dimension of our "constrained" linear system to solve), we can easily reach any weight in

$$\left[\!\!\left[\frac{q-1}{q}\,(n-k),\ k + \frac{q-1}{q}\,(n-k) \right]\!\!\right].$$

It explains why there is a whole interval in which DP is claimed to be easy to solve (as drawn in Figure 3.1). Let us stress once again that no algorithm is known to solve DP in polynomial time outside this range of parameters (up to an additive logarithmic factors in the above interval).

Precise analysis of the algorithm. All the challenge in the analysis of Prange's algorithm running time relies on the computation of the success probability in Step 3. From now on we will we make the following assumption concerning Step 1 of the algorithm

Assumption 1. *The success probability of Prange's algorithm is equal (up to a constant factor) to the probability of success when $\mathscr{I}$ is supposed to be uniformly distributed in Step* 1.

It is an usual assumption (or heuristic) to make when studying the complexity of Prange's algorithm. Notice that we did not suppose that $\mathscr{I}$ is uniformly distributed, but that our probabilities will be well approximated by making this assumption. It would be obviously false to suppose directly that $\mathscr{I}$ is uniformly distributed as $\mathbf{H}_{\overline{\mathscr{I}}}$ will be non-singular for some sets $\mathscr{I}$ (at the exception of very particular cases). However, when $\mathbf{H}$ is random, $\mathbf{H}_{\overline{\mathscr{I}}}$ is typically non-singular.

In the following lemma we give the success probability of Prange's algorithm. Our proof is written with a lot of details. In what follows we will not repeat this and we will process in a simpler way. The idea is that we study algorithms to solve DP *in average* and from a cryptographic point of view we are on the cryptanalysis side. We can suppose to live in the best world for a cryptanalyst. For instance, an event that is expected or that occurs with a probability given by the inverse of a polynomial, *always* happens and we are not concerned with approximation factors (although some heuristics may be hidden). The rationale behind the following proof is to show that what follows during these lecture notes could be stated and proved very precisely but

at the price of significantly increasing the complexity of statements and their proofs while at the same time without changing conclusions.

Proposition 3. *Let p_{pr} be the success probability in Step 3 of the above algorithm. Under Assumption 1, we have*

$$p_{\mathrm{pr}} = \Theta\left(\frac{\binom{n-k}{t-j}(q-1)^{t-j}}{\min\left(q^{n-k}, \binom{n}{t}(q-1)^t\right)}\right)$$

for a density $1-2^{-\Omega(n)}$ of matrices $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$, where

$$j \stackrel{def}{=} \begin{cases} 0 & \text{if } t < \frac{q-1}{q}(n-k) \\ t - \frac{q-1}{q}(n-k) & \text{if } t \in [\![\frac{q-1}{q}(n-k), k + \frac{q-1}{q}(n-k)]\!] \\ k & \text{otherwise.} \end{cases}$$

Proof. Notice that input $(\mathbf{H}, \mathbf{s})$ is fixed, the randomness of the algorithm comes from $\mathbf{x}$ picked according to $\mathscr{D}_t$ and the drawing of the information set $\mathscr{I}$. Under Assumption 1, our probability computations over $\mathscr{I}$ are up to a constant given by the case where $\mathscr{I}$ is uniformly distributed (we will not write the constant during the computations).

Let us fix $\mathbf{e}^{(1)}$ to be a solution of our decoding problem (we know that there is at least one). To compute the success probability of Prange's algorithm let us first notice that an iteration will succeed if $\mathbf{x} = \mathbf{e}^{(1)}_{\mathscr{I}}$, namely

$$\mathbb{P}\left(\text{an iteration of Prange finds } \mathbf{e}^{(1)}\right) = \mathbb{P}_{\mathscr{I},\mathbf{x}}\left(\mathbf{x} = \mathbf{e}^{(1)}_{\mathscr{I}}\right) \tag{3.9}$$

It comes from the fact that $\mathbf{e}^{(1)}$ is uniquely determined by $\mathbf{e}^{(1)}_{\mathscr{I}}$ as necessarily $\mathbf{e}^{(1)}_{\overline{\mathscr{I}}} = \left(\mathbf{s} - \mathbf{e}^{(1)}_{\mathscr{I}}\mathbf{H}^{\intercal}_{\mathscr{I}}\right)\mathbf{S}^{\intercal}$. Furthermore, $\mathscr{D}_t$ only outputs vectors $\mathbf{x}$ of Hamming weight j. To find $\mathbf{e}^{(1)}$ it is necessary that during an iteration we have $\left|\mathbf{e}^{(1)}_{\overline{\mathscr{I}}}\right| = t - j$ as $|\mathbf{e}^{(1)}| = t$. Therefore, using the law of total probability, we obtain the following computation

$$\mathbb{P}_{\mathscr{I},\mathbf{x}}\left(\mathbf{x} = \mathbf{e}^{(1)}_{\mathscr{I}}\right) = \mathbb{P}_{\mathscr{I},\mathbf{x}}\left(\mathbf{x} = \mathbf{e}^{(1)}_{\mathscr{I}} \mid \left|\mathbf{e}^{(1)}_{\mathscr{I}}\right| = t - j\right)\mathbb{P}_{\mathscr{I},\mathbf{x}}\left(\left|\mathbf{e}^{(1)}_{\mathscr{I}}\right| = t - j\right) \tag{3.10}$$

The probability to find $\mathbf{e}^{(1)}$ in one iteration is given by the probability that

(i) $\mathscr{I}$ is such that $\left|\mathbf{e}^{(1)}_{\overline{\mathscr{I}}}\right| = t - j$
(ii) $\mathbf{x} = \mathbf{e}^{(1)}_{\mathscr{I}}$ supposing (i).

Under Assumption 1, the probability of (i) is (up to a constant) $\frac{\binom{t}{j}\binom{n-t}{k-j}}{\binom{n}{k}}$ while the probability of (ii) is given by $\frac{1}{\binom{k}{j}(q-1)^j}$ as $\mathbf{x} \in \mathbb{F}_q^k$ picked according to $\mathscr{D}_t$ is uniformly distributed among words of Hamming weight j. Plugging this in Equation (3.9) and using Equation (3.10) we obtain the following computation

$$\begin{aligned} \mathbb{P}\left(\text{an iteration of Prange finds } \mathbf{e}^{(1)}\right) &= \mathbb{P}_{\mathscr{I},\mathbf{x}}\left(\mathbf{x} = \mathbf{e}^{(1)}_{\mathscr{I}} \mid |\mathbf{e}_{\mathscr{I}}| = t - j\right)\mathbb{P}_{\mathscr{I}}\left(|\mathbf{e}_{\mathscr{I}}| = t - j\right) \\ &= \frac{1}{\binom{k}{j}(q-1)^j}\,\frac{\binom{t}{j}\binom{n-t}{k-j}}{\binom{n}{k}} \\ &= \frac{\binom{n-k}{t-j}(q-1)^{t-j}}{\binom{n}{t}(q-1)^t} \end{aligned}$$

where the last equality follows from a simple computation.

Recall now that we are sure that there is at least one solution of the decoding problem. However, depending on t, it may happen that there are more. Let us denote by N the number of solutions. According to the above equation, the probability to find none of these in one iteration of the algorithm is given by

$$\left(1-\frac{\binom{n-k}{t-j}(q-1)^{t-j}}{\binom{n}{t}(q-1)^t}\right)^N = 1-\Theta\left(N\,\frac{\binom{n-k}{t-j}(q-1)^{t-j}}{\binom{n}{t}(q-1)^t}\right)$$

Here we used that the randomness $\mathscr{I}$, $\mathbf{x}$ of the algorithm is independent of the solutions. Therefore, the probability p_{pr} that Prange's algorithm succeeds is

$$p_{\mathrm{pr}} = \Theta\left(N\,\frac{\binom{n-k}{t-j}(q-1)^{t-j}}{\binom{n}{t}(q-1)^t}(1+o(1))\right) \tag{3.11}$$

But now recall from lecture notes 2[5] that for any constant C,

$$\mathbb{P}_{\mathbf{H}}\left(\left|N-\max\left(1,\frac{\binom{n}{t}(q-1)^t}{q^{n-k}}\right)\right| > C\max\left(1,\frac{\binom{n}{t}(q-1)^t}{q^{n-k}}\right)\right) = 2^{-\Omega(n)}$$

Therefore, since $\mathbf{H}$ is uniformly distributed in the above probability, we have for a density $1-2^{-\Omega(n)}$ of matrices $\mathbf{H}$,

$$p_{\mathrm{pr}} = \Theta\left(\max\left(1,\frac{\binom{n}{t}(q-1)^t}{q^{n-k}}\right)\,\frac{\binom{n-k}{t-j}(q-1)^{t-j}}{\binom{n}{t}(q-1)^t}\right) = \Theta\left(\frac{\binom{n-k}{t-j}(q-1)^{t-j}}{\min\left(q^{n-k},\binom{n}{t}(q-1)^t\right)}\right)$$

where we used Equation (3.11). It concludes the proof. □

We are now ready to give the running-time of Prange's algorithm to solve DP. It will be a simple consequence of the above proposition.

Corollary 1. *Under Assumption 1, the complexity $C_{\mathrm{Prange}}(n,q,R,\tau)$ of Prange's algorithm to solve $\mathsf{DP}(n,q,R,\tau)$ is up to a polynomial factor (in n) given by*

$$\frac{\min\left(q^{n-k},\binom{n}{t}(q-1)^t\right)}{\binom{n-k}{t-j}(q-1)^{t-j}}$$

where $k \overset{def}{=} Rn$, $t \overset{def}{=} \tau n$ and

$$j \overset{def}{=} \begin{cases} 0 & \text{if } t < \frac{q-1}{q}(1-R) \\ t-\frac{q-1}{q}(n-k) & \text{if } t \in [\![\frac{q-1}{q}(n-k), k+\frac{q-1}{q}(n-k)]\!] \\ k & \text{otherwise.} \end{cases}$$

Proof. The cost of an iteration of Prange's algorithm is dominated by the time to perform a Gaussian elimination. Let p_{pr} be the probability of success of an iteration which is given in Proposition 3. The number of iterations is (up to a polynomial in n) $1/p_{\mathrm{pr}}$ with a probability exponentially close to one (in n). The latter affirmation comes from the fact that the number of iterations is a geometric distribution. □

To conclude this section let us briefly study the asymptotic complexity (in n) of Prange's algorithm.

[5] Depending on which term achieves the maximum, we use from lecture notes 2, Proposition 2 or 3.

Asymptotic complexity: use the entropy function. When studying the asymptotic complexity of ISD algorithms it will be important to be familiar with the q-ary entropy function and its properties. Recall from lecture notes 2 that it is defined as (and extended by continuity)

$$h_q : x \in [0,1] \longmapsto -x \log_q \left(\frac{x}{q-1}\right) - (1-x)\log_q(1-x).$$

The q-ary entropy is an increasing function over $\left[0, \frac{q-1}{q}\right]$ and a decreasing function over $\left[\frac{q-1}{q}, 1\right]$. It reaches its maximum 1 in $\frac{q-1}{q}$.
This function has the nice property to describe the asymptotic behaviour of binomials, namely (Lemma 1 in lecture notes 2)

$$\frac{1}{n}\log_q \binom{n}{t}(q-1)^t \underset{n\to+\infty}{=} h_q(\tau) + O\left(\frac{\log_q n}{n}\right) \tag{3.12}$$

where $\tau = t/n$. From this we easily deduce the exponent of Prange's algorithm

$$\frac{1}{n}\log_q C_{\text{Prange}}(n,q,R,\tau) \underset{n\to+\infty}{=} \min\left(1-R, h_q(\tau)\right) - (1-R)\, h_q\left(\frac{\tau-\gamma}{1-R}\right) + O\left(\frac{\log_q n}{n}\right) \tag{3.13}$$

where,

$$\gamma \stackrel{\text{def}}{=} \begin{cases} 0 & \text{if } \tau < \frac{q-1}{q}(n-k) \\ \tau - \frac{q-1}{q}(1-R) & \text{if } \tau \in \left[\frac{q-1}{q}(1-R), R + \frac{q-1}{q}(1-R)\right] \\ R & \text{otherwise.} \end{cases}$$

There are particular ranges of parameters for which Equation (3.12) simplifies. First, in the case where $\tau \in \left[\frac{q-1}{q}(1-R), R + \frac{q-1}{q}(1-R)\right]$, this exponent is a $O\left(\frac{\log_q n}{n}\right)$ (it is easily verified using that $h_q((q-1)/q) = 1$). It corresponds to what we have expected as we claimed that Prange's algorithm is polynomial in this range of parameters.

We used Equation (3.13) multiplied by a factor $\log_2(q)$ (exponents are drawn in base 2) to draw Figures 3.2, 3.3 and 3.4. Notice in the case $q = 2$ that the complexity of Prange's algorithm is symmetric around $1/2$ which is normal in that case. Considering short or large weight in the binary case is equivalent as you have to show in the following exercise. In particular, in the sequel we will not compute the exponents of algorithms for solving $\mathsf{DP}(n, 2, R, \tau)$ with $\tau > 1/2$.

Exercise 4. *Let $\tau \in [0, 1/2]$. Show how from an algorithm solving $\mathsf{DP}(n, 2, R, \tau)$ we can deduce an algorithm solving $\mathsf{DP}(n, 2, R, 1-\tau)$ in the same running-time (and reciprocally).*

Let us consider now the case $\tau = o(1)$ (and therefore $\gamma = 0$). It corresponds to parameters of all code-based public-key encryption schemes (for instance [McE78, Ale03, MTSB13, AAB+17]). Using

$$h_q(x) \underset{x\to 0}{=} -x\log_q\left(\frac{x}{q-1}\right) + x + o(x) \tag{3.14}$$

and Equation (3.13) (with $\gamma = 0$) we obtain the following computation,

$$\begin{aligned}
\frac{1}{n}\log_q C_{\text{Prange}}(n,q,R,\tau) &= \min\left(1-R, h_q(\tau)\right) - (1-R)\, h_q\left(\frac{\tau}{1-R}\right) + O\left(\frac{\log_q n}{n}\right) \\
&= h_q(\tau) - (1-R)\, h_q\left(\frac{\tau}{1-R}\right) + O\left(\frac{\log_q n}{n}\right) \\
&= -\tau \log_q\left(\frac{\tau}{q-1}\right) + \tau \log_q\left(\frac{\tau}{1-R}\,\frac{1}{q-1}\right) + o(\tau) + O\left(\frac{\log_q n}{n}\right) \text{ (see Eq. (3.14))} \\
&= -\tau \log_q(1-R) + o(\tau) + O\left(\frac{\log_q n}{n}\right).
\end{aligned}$$

Therefore, when $\tau = o(1)$, the complexity of Prange algorithm is given by (for some constant C)

$$C_{\text{Prange}}(n,q,R,\tau) = n^C\; q^{-t\log_q(1-R)(1+o(1))}.$$

It is even more remarkable that no algorithm is known to have a complexity $q^{ct(1+o(1))}$ with $c < -\log_q(1-R)$ as soon as $t = o(n)$. Furthermore, all known ISD (even the most sophisticated) have the same asymptotic complexity than Prange's algorithm for these parameters [CS16]. Despite its extreme simplicity, Prange's algorithm is the best known algorithm to solve asymptotically $\mathsf{DP}(n,q,R,\tau)$ when the decoding distance is sub-linear, namely $\tau = o(1)$.

2. Birthday Paradox Techniques

We present in this section two algorithms for solving $\mathsf{DP}(n,q,R,\tau)$. Both rely on the following *crucial* lemma which is essentially an average version of the birthday paradox

Lemma 1. *Let $\mathscr{L}_1, \mathscr{L}_2$ be two lists of L random and independent elements in $\mathbb{F}_q^r$. We have,*

$$\mathbb{E}\left(\sharp\, \mathscr{L}_1 \cap \mathscr{L}_2\right) = \frac{L^2}{q^r}.$$

Notice that we expect one element in the intersection of the two lists included in $\mathbb{F}_q^r$ when their size verifies $L = \sqrt{q^r}$. Recall that the birthday paradox, asserts that when there are $\sqrt{N}$ elements picked uniformly at random among a set of size N, we will get with a good probability two equal elements. It explains why we refer to the above lemma as the birthday paradox.

Proof. By definition, $\mathscr{L}_1 = \{X_1, \ldots, X_L\}$ and $\mathscr{L}_2 = \{Y_1, \ldots, Y_L\}$ where the X_i's and Y_j's are independent and uniformly distributed random variables taking their values in $\mathbb{F}_q^r$. We have

$$\sharp\, \mathscr{L}_1 \cap \mathscr{L}_2 = \sum_{i,j=1}^{L} \mathbb{1}_{\{X_i = Y_j\}}.$$

By linearity of the expectation we have the following computation,

$$\mathbb{E}\left(\sharp\, \mathscr{L}_1 \cap \mathscr{L}_2\right) = \sum_{i,j=1}^{L} \mathbb{E}\left(\mathbb{1}_{\{X_i = Y_j\}}\right) = \sum_{i,j=1}^{L} \frac{1}{q^r} = \frac{L^2}{q^r}$$

which concludes the proof. □

2.1. **Dumer's Algorithm.** Let us now quickly present Dumer's algorithm [Dum86] to solve $\mathsf{DP}(n,q,R,\tau)$. This short subsection may be skipped as the description of this algorithm has already been given in the introduction (in the same fashion).

The algorithm.

1. *Splitting in two parts.* First we randomly select a set $\mathscr{S} \subseteq [\![1,n]\!]$ of $n/2$ positions.
2. *Building lists step.* We build,
$$\mathscr{L}_1 \stackrel{\text{def}}{=} \left\{ \mathbf{H}_{\mathscr{S}} \mathbf{e}_1^\intercal : |\mathbf{e}_1| = \frac{t}{2} \right\} \quad ; \quad \mathscr{L}_2 \stackrel{\text{def}}{=} \left\{ -\mathbf{H}_{\overline{\mathscr{S}}} \mathbf{e}_2^\intercal + \mathbf{s}^\intercal : |\mathbf{e}_2| = \frac{t}{2} \right\}.$$
3. *Collisions step.* We merge the above lists (with an efficient technique like hashing or sorting)
$$\mathscr{L}_1 \bowtie \mathscr{L}_2 \stackrel{\text{def}}{=} \left\{ (\mathbf{e}_1, \mathbf{e}_2) \in \mathscr{L}_1 \times \mathscr{L}_2, \quad \mathbf{H}_{\mathscr{S}} \mathbf{e}_1^\intercal = -\mathbf{H}_{\overline{\mathscr{S}}} \mathbf{e}_2^\intercal + \mathbf{s}^\intercal \right\}.$$
and output this new list. If it is empty we go back to Step 1 and pick another set of $n/2$ positions.

Proposition 4. *The complexity $C_{\text{Dumer}}(n,q,R,\tau)$ of Dumer's algorithm to solve $\mathsf{DP}(n,q,R,\tau)$ is up to a polynomial factor (in n) given by*
$$\sqrt{\binom{n}{t}(q-1)^t} + \frac{\binom{n}{t}(q-1)^t}{q^{n-k}}$$
Furthermore, Dumer's algorithm finds $\max\left(1, \frac{\binom{n}{t}(q-1)^t}{q^{n-k}}\right)$ solutions (up to a polynomial factor in n) where $k \stackrel{def}{=} Rn$ and $t \stackrel{def}{=} \tau n$.

Proof. Let $\mathbf{e}^{(1)}$ be a solution of $\mathsf{DP}(n,q,R,\tau)$. Dumer's algorithm will find $\mathbf{e}^{(1)}$ in one iteration with probability
$$\frac{\binom{t}{t/2}\binom{n-t}{n/2-t/2}}{\binom{n}{n/2}} = 2^{th_2(1/2)+(n-t)h_2(1/2)-nh_2(1/2)+O(\log_2 n)} = 2^{O(\log_2 n)}$$
which is polynomial. Therefore the number of iterations of Dumer's algorithm will be polynomial.
The cost of one iteration is given by the time to build lists $\mathscr{L}_1, \mathscr{L}_2$, namely $\binom{n/2}{t/2}(q-1)^{t/2} = \widetilde{O}\left(\binom{n}{t}(q-1)^t\right)$ plus the time to merge them. With efficient techniques such as sorting or hashing this can be done in time $\sharp\, \mathscr{L}_1 \bowtie \mathscr{L}_2$. But, according to Lemma 1, the expected size of $\mathscr{L}_1 \bowtie \mathscr{L}_2$ is in average over $\mathbf{H}$ given by $\left(\binom{n/2}{t/2}(q-1)^{t/2}\right)^2 / q^{n-k} = \widetilde{O}\left(\binom{n}{t}(q-1)^t / q^{n-k}\right)$ as collisions are made on vectors which belong to $\mathbb{F}_q^{n-k}$. It concludes the proof. □

Remark 2. *We have presented Dumer's algorithm to find all solutions of DP. But one can also tweak this algorithm to build less solutions in one iteration.*

Exercise 5. *We have made the choice when presenting Dumer's algorithm to build lists of maximum size, namely $\binom{n/2}{t/2}(q-1)^{t/2}$. Let $(\mathbf{H},\mathbf{s}) \in \mathbb{F}_q^{(n-k)\times n} \times \mathbb{F}_q^{n-k}$ be an instance of a decoding problem that we would like to solve at distance t. We suppose that $(\mathbf{H},\mathbf{s})$ are uniformly distributed, in particular we do not suppose that there is always a solution. Show that a slight variation of Dumer's algorithm enables to compute $\frac{L^2}{q^{n-k}}$ solutions (there is no maximum in this formula, why?) in time $L + \frac{L^2}{q^{n-k}}$ (up to polynomial factors). Furthermore L has necessarily to verify $L \leq \binom{n/2}{t/2}(q-1)^{t/2}$, why? What is the condition over t and L for this algorithm to output solutions in amortized time one?*

2.2. **Wagner's Algorithm** [Wag02]**.** We have just seen that Dumer's algorithm finds all solutions of DP in roughly one iteration. It is an extremely nice property but that may be an impediment in some contexts. Suppose that one needs M solutions of DP to achieve some task. The best situation would be to find them in amortized time one. Suppose now that Dumer's algorithm is able to compute N solutions of DP in amortized time one, namely N verifies

$$N = \frac{N^2}{q^{n-k}} \iff N = q^{n-k}$$

but unfortunately $N \gg M$. In other words, Dumer's algorithm finds too many solutions. To avoid this useless situation we may be tempted to decrease the size of the built lists, namely N, to decrease the number of output solutions. However by doing this we would not produce decoding solutions in amortized time one, which would be less efficient for our purpose. To improve this situation, the fundamental remark is that Dumer's algorithm produces all its solutions with a shape $(\mathbf{e}_1, \mathbf{e}_2)$, where $|\mathbf{e}_1| = |\mathbf{e}_2| = t/2$, and by looking at collisions directly on $n-k$ symbols. The idea to produce less solutions, still in amortized time one, is to look for solutions with more constraints on their shapes and the way that collisions are built. It is precisely the idea of Wagner's algorithm [Wag02] (producing less solutions in amortized time one by decimating the search space) that we are going to present precisely in the sequel. However, as a picture is better than a long discourse, let us first describe in Figure 3.5 a simplified version of this algorithm when we try to find $\mathbf{e}$ of Hamming weight t such that $\mathbf{H}\mathbf{e}^\intercal = \mathbf{0}$.

The output of Wagner's algorithm described in Figure 3.5 is $(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \mathbf{e}_4)$. It is a solution as by construction it reaches the syndrome $\mathbf{0}$ with respect to $\mathbf{H}$ and it has the right Hamming weight since each $\mathbf{e}_i$ has weight $t/4$. Notice that this solution has a particular "shape" when compared to the output of Dumer's algorithm. During Steps 2 and 3 we do not perform collisions on all the $n-k$ symbols of the syndromes but on $(n-k)/2$ symbols. Therefore solutions have the following property: $(\mathbf{H}_1\ \mathbf{H}_2)\,(\mathbf{e}_1, \mathbf{e}_2)^\intercal$ and $(\mathbf{H}_3\ \mathbf{H}_4)\,(\mathbf{e}_3, \mathbf{e}_4)^\intercal$ are equal to 0 on the last $(n-k)/2$ positions. If one splits an output of Dumer's algorithm and the parity-check matrix in four parts, there is no reason that it verifies the above property. It will be true only for an exponentially small fraction of the solutions. It explains why Wagner's algorithm "decimates" the solutions. At the same time this algorithm has the advantage to be able to produce solutions in amortized time one. The idea in that case is to build the lists $\mathscr{L}_i$'s with size L such $L^2/q^{(n-k)/2} = L$, *i.e.* $L = \sqrt{q^{n-k}}$. Therefore, each collision step has the same cost given by the size L of the lists that are output. However it may happen that the number of solutions is still too large. If so, the next idea of Wagner's algorithm is to consider more lists at the beginning, like 8 (and not 4) and then to make collisions on $(n-k)/3$ symbols. It enables smaller lists, namely $L^2/q^{(n-k)/3} = L$, *i.e.* $L = \sqrt[3]{q^{n-k}}$. However, in that case, there will be three steps of collisions. Then we can extend this by considering a number of lists given by some 2^a and making a steps of collisions. Nonetheless, it is not possible to do this for any a. If one uses Wagner's algorithm with initially 2^a lists, all of them need to be built from vectors $\mathbf{e}_i$ of Hamming weight $t/2^a$. If a is too large it will be impossible to build lists large enough to produce collisions in amortized time one.

Let us emphasize that the above discussion is not only a thought exercise. It turns out that the above situation happens with ISD algorithms (Dumer's algorithm produces too many solutions in one iteration). It explains why the ISD with Wagner's algorithm outperforms the ISD with Dumer's algorithm for some parameters as we can see in Figure 3.4 (in particular when DP is such that there are many solutions).

Proposition 5. *Wagner's algorithm solves* $\mathsf{DP}(n, q, R, \tau)$ *by (where* $k \stackrel{def}{=} Rn$*)*

(1) finding one solution in time and space $q^{\frac{n-k}{a+1}}$ *(up to a polynomial factor in* n*) for any integer* a *such that* $q^{\frac{n-k}{a+1}} \leq \binom{n/2^a}{t/2^a}(q-1)^{t/2^a}$ *which is asymptotically,*

$$\frac{1-R}{h_q(\tau)} \leq \frac{a+1}{2^a}.$$

$$\mathbf{H} = (\mathbf{H}_1\ \mathbf{H}_2\ \mathbf{H}_3\ \mathbf{H}_4) \text{ where the } \mathbf{H}_i \in \mathbb{F}_q^{(n-k)\times n/4}$$

1st Step: compute the following lists

$$\mathscr{L}_1 \stackrel{\text{def}}{=} \{\mathbf{H}_1\mathbf{e}_1^\intercal : |\mathbf{e}_1| = t/4\} \quad \mathscr{L}_2 \stackrel{\text{def}}{=} \{\mathbf{H}_2\mathbf{e}_2^\intercal : |\mathbf{e}_2| = t/4\} \quad \mathscr{L}_3 \stackrel{\text{def}}{=} \{\mathbf{H}_3\mathbf{e}_3^\intercal : |\mathbf{e}_3| = t/4\} \quad \mathscr{L}_4 \stackrel{\text{def}}{=} \{\mathbf{H}_4\mathbf{e}_4^\intercal : |\mathbf{e}_4| = t/4\}$$

2nd Step: compute the following lists obtained by collision

$(\mathbf{H}_1\mathbf{e}_1^\intercal, \mathbf{H}_2\mathbf{e}_2^\intercal) \in \mathscr{L}_1 \times \mathscr{L}_2$ s.t

$\mathbf{H}_1\mathbf{e}_1^\intercal =$  $\mathbf{H}_2\mathbf{e}_2^\intercal =$ - 

$\longrightarrow (\mathbf{H}_1\ \mathbf{H}_2)(\mathbf{e}_1, \mathbf{e}_2)^\intercal =$ 

$(\mathbf{H}_3\mathbf{e}_3^\intercal, \mathbf{H}_4\mathbf{e}_4^\intercal) \in \mathscr{L}_3 \times \mathscr{L}_4$ s.t

$\mathbf{H}_3\mathbf{e}_3^\intercal =$  $\mathbf{H}_4\mathbf{e}_4^\intercal =$ -  $(n-k)/2$

$\longrightarrow (\mathbf{H}_3\ \mathbf{H}_4)(\mathbf{e}_3, \mathbf{e}_4)^\intercal =$ 

3rd Step: compute the following lists obtained by collision

$(\mathbf{H}_1\ \mathbf{H}_2)(\mathbf{e}_1, \mathbf{e}_2)^\intercal$ and $(\mathbf{H}_3\ \mathbf{H}_4)(\mathbf{e}_3, \mathbf{e}_4)^\intercal$ in the above lists s.t

$(\mathbf{H}_1\ \mathbf{H}_2)(\mathbf{e}_1, \mathbf{e}_2)^\intercal =$ 

$(\mathbf{H}_3\ \mathbf{H}_4)(\mathbf{e}_3, \mathbf{e}_4)^\intercal =$ - 

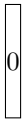
$\longrightarrow (\mathbf{H}_1\ \mathbf{H}_2\ \mathbf{H}_3\ \mathbf{H}_4)(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \mathbf{e}_4)^\intercal =$ 

FIGURE 3.5. Simplified version of Wagner's algorithm with four layers to find $\mathbf{e}$ of weight t such that $\mathbf{He} = \mathbf{0}$. The same colours on vectors means that they are equal (be careful on the minus signs).

(2) finding $q^{\frac{n-k}{a}}$ solutions in amortized time 1 *(up to a polynomial factor in n) for any integer a such that $q^{\frac{n-k}{a}} \leq \binom{n/2^a}{t/2^a}(q-1)^{t/2^a}$ which is asymptotically,*

$$\frac{1-R}{h_q(\tau)} \leq \frac{a}{2^a}.$$

During the description of Wagner's algorithm that follows (which will give the proof of the above proposition) we will use Lemma 1 to estimate the size of the lists after merging.

Wagner's algorithm. The first step is to split $\mathbf{H}$ in 2^a parts of the same size, for a parameter a that is called *depth of the algorithm*. For the sake of simplicity let us split $\mathbf{H}$ as (we can choose the partition)

$$\mathbf{H} = \begin{pmatrix}\mathbf{H}_1 & \dots & \mathbf{H}_{2^a}\end{pmatrix}$$

where for all i we have $\mathbf{H}_i \in \mathbb{F}_q^{(n-k)\times \frac{n}{2^a}}$. Then we build the following 2^a-lists for some parameter L that will be fixed later ($t = \tau n$)

$$\forall i \in [\![1, 2^a]\!], \quad \mathscr{L}_i \subseteq \left\{ \mathbf{e}\mathbf{H}_i^\intercal \,:\, \mathbf{e} \in \mathbb{F}_q^{n/2^a},\ |\mathbf{e}| = \frac{t}{2^a} \right\} \quad \textit{and} \quad \sharp\, \mathscr{L}_i = L.$$

Notice that by construction we have the following constraint,

$$L \leq \binom{n/2^a}{t/2^a} (q-1)^{t/2^a}. \tag{3.15}$$

Let $\ell \in [\![1, n-k]\!]$ be a parameter that will be chosen later. Then, Wagner's algorithm performs the collision of these lists two by two on their last ℓ symbols [6] to build the new lists $\mathscr{L}_{i,i+1}$'s, namely

$$\mathscr{L}_{i,i+1} \stackrel{\text{def}}{=} \{\mathbf{s}_i + \mathbf{s}_{i+1} \,:\, \mathbf{s}_i \in \mathscr{L}_i \text{ and the last } \ell \text{ symbols of } \mathbf{s}_i + \mathbf{s}_{i+1} \text{ are } \mathbf{0}\},$$

where by construction we have access to the errors $\mathbf{e}_i$ and $\mathbf{e}_{i+1}$ of Hamming weight $t/2^a$ that reach $\mathbf{s}_i$ and $\mathbf{s}_{i+1}$ through $\mathbf{H}_i$ and $\mathbf{H}_{i+1}$. The last list $\mathscr{L}_{2^a-1,2^a}$ is built by merging $\mathscr{L}_{2^a}$ and $\mathscr{L}_{2^a-1}$ but this time according to the last ℓ symbols of $\mathbf{s}$, namely

$$\mathscr{L}_{2^a-1,2^a} \stackrel{\text{def}}{=} \{\mathbf{s}_{2^a-1} + \mathbf{s}_{2^a} \,:\, \mathbf{s}_i \in \mathscr{L}_i \text{ and the last } \ell \text{ symbols of } \mathbf{s}_{2^a-1} + \mathbf{s}_{2^a} \text{ are equal to those of } \mathbf{s}\}.$$

Using Lemma 1, these new lists built after merging on ℓ symbols will be of the same size,

$$\frac{L^2}{q^\ell}.$$

Furthermore, we produce them at cost $L + \frac{L^2}{q^\ell}$ (up to a polynomial factor). Once this is done, we start again this process $a-2$ times by merging each time on the ℓ next new symbols. Wagner proposed to choose L such that at each step, *the time for merging is the same than the one to build lists*, namely

$$L = q^\ell. \tag{3.16}$$

This implies under Constraint (3.15) that the parameter ℓ is such that

$$q^\ell \leq \binom{n/2^a}{t/2^a} (q-1)^{t/2^a}. \tag{3.17}$$

Remark 3. *One may ask if this strategy of an amortized time one at each merge is optimal. It turns out that the answer is yes as proved in* [MS09].

Up to now we have made $a-1$ merges and we still have two lists, that we denote by $\mathscr{S}_1$ and $\mathscr{S}_2$. They are such that

$$\mathscr{S}_1 = \{\mathbf{s}_1 + \cdots + \mathbf{s}_{2^{a-1}} \,:\, \mathbf{s}_i \in \mathscr{L}_i \text{ and the last } (a-1)\ell \text{ symbols of } \mathbf{s}_1 + \cdots + \mathbf{s}_{2^{a-1}} \text{ are equal to } \mathbf{0}\}$$

$$\mathscr{S}_2 = \{\mathbf{s}_{2^{a-1}+1} + \cdots + \mathbf{s}_{2^a} \,:\, \mathbf{s}_i \in \mathscr{L}_i \text{ and the last } (a-1)\ell \text{ symbols of } \mathbf{s}_1 + \cdots + \mathbf{s}_{2^{a-1}} \text{ are equal to those of } \mathbf{s}\}$$

$$\text{where,} \qquad \sharp\, \mathscr{S}_1 = \sharp\, \mathscr{S}_2 = \frac{L^2}{q^\ell} = L = q^\ell.$$

Therefore, it remains to merge these two lists on the last $(n-k)-(a-1)\ell$ first symbols. It yields in time $\frac{q^{\ell(a+1)}}{q^{(n-k)}}$ a list of solutions of size

$$\frac{q^{2\ell}}{q^{(n-k)-(a-1)\ell}} = \frac{q^{\ell(a+1)}}{q^{(n-k)}}. \tag{3.18}$$

[6] Given a vector $\mathbf{x} \in \mathbb{F}_q^m$, it denotes $x_{m-\ell+1}, \ldots, x_m$.

Now the parameter ℓ has to be set whether one wants only one solution or many solutions in amortized time one.

Wagner to reach one solution. According to Equation (3.18), it remains to choose parameters such that

$$\ell = \frac{n-k}{a+1}.$$

All the lists in the $a-1$ first steps of the algorithm have the same size, namely $L = q^\ell$ (Equation (3.16)), therefore the algorithm has a cost given by

$$q^{\frac{n-k}{a+1}}.$$

However, we have to be careful with the depth a of the algorithm, unfortunately it cannot be chosen too large. According to Equation (3.15)

$$q^\ell = q^{\frac{n-k}{a+1}} \leq \binom{n/2^a}{t/2^a}(q-1)^{t/2^a}$$

which leads to the following asymptotic constraint,

$$\frac{1-R}{a+1} \leq \frac{1}{2^a}\, h_q(\tau) \iff \frac{1-R}{h_q(\tau)} \leq \frac{a+1}{2^a}.$$

It concludes the proof of (1).

Wagner to compute many solutions in amortized time one. In this case, according to Equation (3.18), we just need to choose ℓ such that

$$q^\ell = \frac{q^{\ell(a+1)}}{q^{n-k}} \iff \ell = \frac{n-k}{a}$$

As above we obtain the claimed constraint on a. It concludes the proof. □

We draw in Figure 3.6 the exponent of Wagner's algorithm to solve $\mathsf{DP}(n, q, R, \tau)$ as function of $\tau \geq \tau^-$ (the relative Gilbert-Varshamov distance defined in lecture notes 2). We choose parameters of the algorithm to output one solution and a being the largest integer that satisfies the constraint (1) (to have an optimal complexity). As it can be seen the exponent is a decreasing function of τ. Indeed, when τ increases, a can be chosen larger. Furthermore, there is a discontinuity in the exponent. It comes from the fact that a is an *integer*. It is possible to adapt the algorithm to "smooth" its complexity but this is out of scope of this lecture notes.

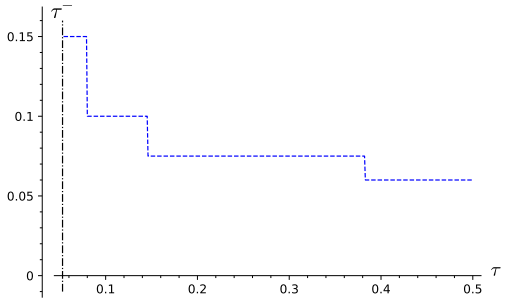


FIGURE 3.6. Exponent of Wagner's algorithm to solve $\mathsf{DP}(n, q, R, \tau)$ for $R = 0.7$ as function of τ.

3. Combining Linear Algebra and Birthday Paradox Techniques

We are now ready to present the general framework (introduced in [FS09]) of *Information Set Decoding* (ISD) algorithms.

The algorithm. Let us introduce the following parameters,

$$\ell \in [\![0, n-k]\!] \quad \text{and} \quad p \in [\![0, \min(t, k+\ell)]\!]$$

1. *Picking the augmented information set.* Let $\mathscr{J} \subseteq [\![1, n]\!]$ be a random set of size $k+\ell$. If $\mathbf{H}_{\overline{\mathscr{J}}} \in \mathbb{F}_q^{(n-k)\times(n-k)}$ is not of full-rank, pick another set $\mathscr{J}$.
2. *Linear algebra.* Perform a Gaussian elimination to compute a non-singular matrix $\mathbf{S} \in \mathbb{F}_q^{(n-k)\times(n-k)}$ such that $\mathbf{S}\mathbf{H}_{\overline{\mathscr{J}}} = \begin{pmatrix} \mathbf{1}_{n-k-\ell} \\ \mathbf{0}_\ell \end{pmatrix}$. Let $\mathbf{H}' \in \mathbb{F}_q^{(n-k-\ell)\times(k+\ell)}$, $\mathbf{H}'' \in \mathbb{F}_q^{\ell\times(k+\ell)}$, $\mathbf{s}' \in \mathbb{F}_q^{n-k-\ell}$ and $\mathbf{s}'' \in \mathbb{F}_q^{\ell}$ be such that

$$\mathbf{S}\mathbf{H}_{\mathscr{J}} = \begin{pmatrix} \mathbf{H}' \\ \mathbf{H}'' \end{pmatrix} \quad \text{and} \quad \mathbf{S}\mathbf{s}^\intercal = (\mathbf{s}', \mathbf{s}'')^\intercal \tag{3.19}$$

3. *Sub-decoding problem.* Compute a set,

$$\mathscr{S} \subseteq \left\{ \mathbf{e}'' \in \mathbb{F}_q^{k+\ell} \ : \ \mathbf{e}''\mathbf{H}''^\intercal = \mathbf{s}'' \text{ and } |\mathbf{e}''| = p \right\}. \tag{3.20}$$

4. *Test.* Find $\mathbf{e}'' \in \mathscr{S}$ such that $\left| \mathbf{s}' - \mathbf{e}'\mathbf{H}''^\intercal \right| = t - p$. If not, return to Step 1; otherwise output $\mathbf{e} \in \mathbb{F}_q^n$ such that

$$\mathbf{e}_{\overline{\mathscr{J}}} = \mathbf{s}' - \mathbf{e}''\mathbf{H}'^\intercal \quad ; \quad \mathbf{e}_{\mathscr{J}} = \mathbf{e}'' \tag{3.21}$$

Correction of the algorithm. It easily follows from the following computation,

$$\begin{aligned}
\mathbf{S}\mathbf{H}\mathbf{e}^\intercal &= \mathbf{S}\mathbf{H}_{\overline{\mathscr{J}}}\mathbf{e}_{\overline{\mathscr{J}}}^\intercal + \mathbf{S}\mathbf{H}_{\mathscr{J}}\mathbf{e}_{\mathscr{J}}^\intercal \\
&= \begin{pmatrix} \mathbf{1}_{n-k-\ell} \\ \mathbf{0}_\ell \end{pmatrix} \left(\mathbf{s}'^\intercal - \mathbf{H}''\mathbf{e}''^\intercal \right) + \begin{pmatrix} \mathbf{H}' \\ \mathbf{H}'' \end{pmatrix} \mathbf{e}''^\intercal \quad \text{(By definition of } \mathbf{S}\mathbf{H}_{\overline{\mathscr{J}}} \text{ and } \mathbf{S}\mathbf{H}_{\mathscr{J}}\text{)} \\
&= \begin{pmatrix} \mathbf{s}'^\intercal - \mathbf{H}''\mathbf{e}''^\intercal \\ \mathbf{0}_\ell \end{pmatrix} + \begin{pmatrix} \mathbf{H}'\mathbf{e}''^\intercal \\ \mathbf{H}''\mathbf{e}''^\intercal \end{pmatrix} \\
&= \begin{pmatrix} \mathbf{s}'^\intercal \\ \mathbf{s}''^\intercal \end{pmatrix} \qquad \text{(By definition of } \mathbf{e}'' \in \mathscr{S}\text{, see Equation (3.20))} \\
&= \mathbf{S}\mathbf{s}^\intercal \qquad \text{(By Equation (3.19))}
\end{aligned}$$

which corresponds to $\mathbf{H}\mathbf{e}^\intercal = \mathbf{s}^\intercal$ since $\mathbf{S}$ is non-singular. Furthermore, by definition $\mathbf{e}''$ has Hamming weight p and the test ensures that $\mathbf{s}' - \mathbf{e}''\mathbf{H}''^\intercal$ has weight $t-p$. Therefore, once the algorithm terminates, $\mathbf{e}$ reaches $\mathbf{s}$ with respect to $\mathbf{H}$ and has Hamming weight t.

Exercise 6. *Let*

$$\mathscr{D} \stackrel{def}{=} \left\{ \mathbf{c}'' \in \mathbb{F}_q^{k+\ell} \ : \ \mathbf{c}''\mathbf{H}''^\intercal = \mathbf{0} \right\}.$$

Show that $\mathscr{D}$ is a code of length $k+\ell$ and dimension k.

Remark 4. *The code of parity-check matrix* $\mathbf{H}''$ *is known as the punctured code (defined by the parity-check matrix* $\mathbf{H}$*) at the positions* $\overline{\mathscr{J}}$*. Computing* $\mathscr{S}$ *in Equation* (3.20) *amounts to solve a decoding problem at distance* p *with this input code and the syndrome* $\mathbf{s}''$*. Therefore, for each drawn of the augmented information set* $\mathscr{J}$ *we test many decoding candidates (given by elements of the list* $\mathscr{S}$ *and with associated lift defined in Equation* (3.21)*). We recover the interpretation of* ISD *algorithms given in the introduction.*

Far or close codeword? One may wonder why don't we use the distribution $\mathscr{D}_t$ in ISD algorithms to be able to produce "short" or "large" solutions? To answer this question let us take a look at the typical weight of a vector $\mathbf{e}$ that will pass the test at the end of an iteration (see Equation (3.21)). By supposing that $\mathbf{s}$ is uniformly distributed, we have

$$\mathbb{E}\left(|\mathbf{e}|\right) = p + \frac{q-1}{q}\ (n-k-\ell) \tag{3.22}$$

The $\frac{q-1}{q}\ (n-k-\ell)$ term comes from the fact that $\mathbf{s}'$ is uniformly distributed over $\mathbb{F}_q^{n-k-\ell}$ while p is here as by definition $\mathbf{e}''$ has weight p. If one wants to get a solution of small weight, the best approach is to decode the punctured code at a small as possible distance p. On the other hand, if one seeks a solution of large weight, one has to decode this punctured code at the largest as possible distance, namely $p = k + \ell$. Therefore the strategy to reach short or large error relies on how we choose the parameter p.

The above discussion hints us why we can not reasonably hope, with ISD algorithms, to solve DP in polynomial time outside the interval $[\![\frac{q-1}{q}\ (n-k), k + \frac{q-1}{q}\ (n-k)]\!]$. For instance, if one is looking for an ISD algorithm solving DP in polynomial time for some $t < \frac{q-1}{q}(n-k)$, one has according to Equation (3.22) to find a subroutine decoding in polynomial time a random code of length $k+\ell$ and dimension k at distance p such that

$$p - \frac{q-1}{q}\ell < 0.$$

But at the same time, the smaller p for which we known how to decode in polynomial a random $[k+\ell, k]$-code is precisely $\frac{q-1}{q}\ (k+\ell-k) = \frac{q-1}{q}\ \ell$ which is the above limit to get an improvement. Therefore, if one seeks an ISD enlarging the interval of weights in which Prange's algorithm is polynomial, one has to first enlarge this interval.

Analyse of the algorithm. As in Prange's algorithm, all the challenge in the analysis of ISD algorithms running time relies on the computation of the success probability in Step 4. However, contrary to Prange's algorithm it will not be necessary to make an assumption on how the augmented information sets are picked. We can suppose directly, when $\ell = \Theta(n)$ (which will be the case in our applications), that they are uniformly distributed as shown by the following proposition.

Proposition 6. *Let* $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ *and* $\mathscr{J} \subseteq [\![1, n]\!]$ *being uniformly distributed over the sets of size* $k+\ell$ *(where* $\ell > 0$*) such that* $\mathbf{H}_{\overline{\mathscr{J}}}$ *is non-singular. Let* $\mathscr{J}_{\text{unif}} \subseteq [\![1, n]\!]$ *being uniformly distributed over the sets of size* $k+\ell$*. We have,*

$$\mathbb{E}_{\mathbf{H}}\left(\Delta\left(\mathscr{J}, \mathscr{J}_{\text{unif}}\right)\right) = O\left(\frac{1}{q^\ell}\right)$$

where Δ *denotes the statistical distance.*

Proof. Let us index from 1 to $\binom{n}{k+\ell}$ the subset of size $k+\ell$ of $[\![1, n]\!]$ and let X_i be the indicator of the event "the subset $\mathscr{J}_i$ of index i is such that $\mathbf{H}_{\overline{\mathscr{J}_i}}$ has not a full rank". Let,

$$N \stackrel{\text{def}}{=} \sum_i X_i$$

It can be verified that we have

$$\mathbb{E}_{\mathbf{H}}\left(\Delta\left(\mathscr{J},\mathscr{J}_{\text{unif}}\right)\right)=\mathbb{E}_{\mathbf{H}}\left(\frac{N}{\binom{n}{k+\ell}}\right)=\frac{1}{\binom{n}{k+\ell}}\sum_{i=1}^{\binom{n}{k+\ell}}\mathbb{E}_{\mathbf{H}}(N_i) \tag{3.23}$$

where the last equality follows from the linearity of the expectation.
Notice now that $\mathbf{H}_{\overline{\mathscr{J}}}\in\mathbb{F}_q^{(n-k)\times(n-k-\ell)}$ has not a full rank with probability (over $\mathbf{H}$) given by a $O\left(\frac{1}{q^\ell}\right)$. Therefore,

$$\mathbb{P}\left(X_i=1\right)=O\left(\frac{1}{q^\ell}\right)$$

Plugging this in Equation (3.23) concludes the proof by linearity of the expectation. □

However, although the above proposition enables to avoid an assumption, there will be as for Prange, an assumption to make when studying ISD algorithms.

An important quantity. Let us use the notations of the above algorithm. Let $\alpha_{p,\ell}$ be the probability that given a fixed $\mathbf{x}\in\mathbb{F}_q^{k+\ell}$ be such that $\begin{cases}\mathbf{H}''\mathbf{x}^\intercal=\mathbf{s}''\\ |\mathbf{x}|=p\end{cases}$, the vector $\mathbf{e}'\stackrel{\text{def}}{=}\mathbf{s}'-\mathbf{x}\mathbf{H}'^\intercal$ has Hamming weight $t-p$, namely

$$\alpha_{p,\ell}\stackrel{\text{def}}{=}\mathbb{P}\left(|\mathbf{e}'|=t-p\right).$$

In other words, $\alpha_{p,\ell}$ denotes the probability that given a solution $\mathbf{x}$ of the decoding problem at distance p with input $(\mathbf{H}'',\mathbf{s}'')$, then its lift gives a solution of weight t of the initial decoding problem with input $(\mathbf{H},\mathbf{s})$. Notice that we did not suppose that $\mathbf{x}\in\mathscr{S}$.

Proposition 7. *The probability $\alpha_{p,\ell}$ is up to a polynomial factor (in n) given by,*

$$\frac{\binom{n-k-\ell}{t-p}(q-1)^{t-p}}{\min\left(q^{n-k-\ell},\binom{n}{t}(q-1)^t q^{-\ell}\right)}$$

The proof of this proposition is similar to the one of Proposition 3 and here we only provide a sketch of it.

Sketch of proof. One can remark that that the only difference between formulas is the factor $q^{-\ell}$ in the denominator. The difference comes from the fact that the probability (over $\mathbf{H}$) that an error $(\mathbf{e}',\mathbf{e}'')$ of weight t verifies $\mathbf{H}''\mathbf{e}''^\intercal=\mathbf{s}''^\intercal$, where $\mathbf{s}''\in\mathbb{F}_q^\ell$, is $q^{-\ell}$. Therefore we have to consider a fraction $q^{-\ell}$ of possible solutions in our probability, which roughly explains the factor $q^{-\ell}$ in the denominator. □

We are now ready to give the running time of ISD algorithms to solve DP. It will use the following assumption

Assumption 2. *Let us use notation of* ISD *algorithm that is described above. Given $\mathbf{x}^{(1)},\ldots,\mathbf{x}^{(S)}$ be solution of $\begin{cases}\mathbf{H}''\mathbf{x}^\intercal=\mathbf{s}''\\ |\mathbf{x}|=p\end{cases}$ Then, the vectors $\mathbf{e}^{(i)}\in\mathbb{F}_q^n$ be defined as*

$$\mathbf{e}^{(i)}_{\overline{\mathscr{J}}}=\mathbf{s}'-\mathbf{x}^{(i)}\mathbf{H}'^\intercal\quad;\quad\mathbf{e}^{(i)}_{\mathscr{J}}=\mathbf{x}^{(i)}$$

are independent random variables (where $\mathscr{J}$ is a random augmented information set).

Proposition 8. *Let $\ell\in[\![0,n-k]\!]$. Let $\mathscr{A}$ be an algorithm that can compute S solutions in time T of the following problem $\begin{cases}\mathbf{H}''\mathbf{x}^\intercal=\mathbf{s}''\\ |\mathbf{x}|=p\end{cases}$ where $\mathbf{H}\in\mathbb{F}_q^{\ell\times(k+\ell)}$ and $\mathbf{s}''\in\mathbb{F}_q^\ell$. Furthermore, we suppose that outputs of*

$\mathscr{A}$ verify Assumption 2. Then, the ISD algorithm using $\mathscr{A}$ in Step 3 *solves* $\mathsf{DP}(n,q,R,\tau)$ *up to a polynomial factor (in n) in time*

$$T\ \max\left(1,\frac{1}{S\ \alpha_{p,\ell}}\right)$$

where $\alpha_{p,\ell}$ is given in Proposition 7.

As in Proposition 7 we only provide a sketch of proof of this proposition.

Sketch of proof. The number of iterations of ISD algorithms is, as for Prange's algorithm, up to a polynomial factor (in n) given by $1/p_{\mathrm{ISD}}$ where p_{ISD} is the probability of success of an iteration. Furthermore, each iteration has a cost given by the time to computing $\mathscr{S}$ in Step 3 (which dominates the cost of a Gaussian elimination). Therefore the cost of the ISD using $\mathscr{A}$ is given by $T\ \frac{1}{p_{\mathrm{ISD}}}$.

Let us compute p_{ISD}. Let $\mathbf{e}''_1,\dots,\mathbf{e}''_S$ be the outputs of $\mathscr{A}$. The probability that any $\mathbf{e}''_i$ does not lead to a solution is given by $1-\alpha_{p,\ell}$. Using the independence given by Assumption 2, the probability that none of the $\mathbf{e}''_i$'s leads to a solution is given by

$$1-(1-\alpha_{p,\ell})^S = 1-\Theta\left(\min(1,S\alpha_{p,\ell})\right).$$

Therefore, $p_{\mathrm{ISD}}=\Theta\left(\min(1,S\alpha_{p,\ell})\right)$ and

$$T\ \frac{1}{p_{\mathrm{ISD}}} = T\ \frac{1}{\Theta\left(\min(1,S\alpha_{p,\ell})\right)} = \Theta\left(T\ \max\left(1,\frac{1}{S\ \alpha_{p,\ell}}\right)\right)$$

which concludes the proof. □

We are now ready to "instantiate" ISD algorithms with Dumer and Wagner algorithms that we have described in Subsections 2.1 and 2.2.

3.1. **ISD with Dumer's algorithm.** A slight variation of Proposition 4 shows that, given an instance $(\mathbf{H}'',\mathbf{s}'')\in\mathbb{F}_q^{\ell\times(k+\ell)}\times\mathbb{F}_q^{\ell}$ of a decoding problem at distance p, Dumer's algorithm find

$$\frac{\binom{k+\ell}{p}(q-1)^p}{q^\ell}$$

solutions in average time

$$\sqrt{\binom{k+\ell}{p}(q-1)^p}+\frac{\binom{k+\ell}{p}(q-1)^p}{q^\ell}.$$

Here there is no maximum in the fomula as we are not sure that there is always a solution to our decoding problem.

Therefore we easily deduce the following proposition which gives the complexity of the ISD using Dumer's algorithm.

Proposition 9. *The complexity $C_{\mathrm{Dumer}}(n,q,R,\tau)$ of the* ISD *using Dumer's algorithm (described in Subsection 2.1) to solve $\mathsf{DP}(n,q,R,\tau)$ is up to a polynomial factor (in n) given by*

$$\left(\sqrt{\binom{k+\ell}{p}(q-1)^p}+\frac{\binom{k+\ell}{p}(q-1)^p}{q^\ell}\right)\cdot\max\left(1,\frac{\min\left(q^{n-k},\binom{n}{t}(q-1)^t\right)}{\binom{n-k-\ell}{t-p}(q-1)^{t-p}\ \binom{k+\ell}{p}(q-1)^p}\right) \tag{3.24}$$

This complexity is parametrized by p and ℓ. According to our wish, finding a short or large solution, the optimization will not be the same. Let us describe our strategy for both of them but before let us fix the relative quantities that we will consider

$$R\stackrel{\text{def}}{=}\frac{k}{n},\quad \tau\stackrel{\text{def}}{=}\frac{w}{n},\quad \lambda\stackrel{\text{def}}{=}\frac{\ell}{n}\quad\text{and}\quad \pi\stackrel{\text{def}}{=}\frac{p}{n}.$$

These quantities will be useful as we are interested in the *asymptotic complexity* of the ISD's.

Strategy to reach short solutions. Our first choice is to force Dumer's algorithm to produce decoding solutions in amortized time one. Let us stress that here we give a method to optimize the complexity of ISD's, but we do not claim that it will lead to optimal parameters. Anyway, Dumer's algorithm computes solutions in amortized time one if

$$\sqrt{\binom{k+\ell}{p}(q-1)^p} = \frac{\binom{k+\ell}{p}}{q^\ell} \iff q^\ell = \sqrt{\binom{k+\ell}{p}(q-1)^p}$$

Using Equation (3.12), it implies asymptotically the following equality

$$\lambda = \frac{R+\lambda}{2} h_q\left(\frac{\pi}{R+\lambda}\right) \iff \pi = (R+\lambda) h_q^{-1}\left(\frac{2\lambda}{R+\lambda}\right) \tag{3.25}$$

Let $\pi(\lambda)$ be the parameter π that reaches the above equality. We can now verify that according to Equations (3.12), (3.24) and (3.25) that

$$\frac{1}{n}\log_q(C_{\text{Dumer}}) = f(\lambda)(1+o(1))$$

where

$$f(\lambda) \stackrel{\text{def}}{=} \lambda + \max\left(0, \min\left(1-R, h_q(\tau)\right) - (1-R-\lambda) h_q\left(\frac{\tau-\pi(\lambda)}{1-R-\lambda}\right) - 2\lambda\right).$$

To optimize $\lambda \mapsto f(\lambda)$, a good approximation (which can be verified for many parameters) is to suppose that it is an unimodal function. Then its minimization is easy to obtain with for instance the golden section search (see `https://en.wikipedia.org/wiki/Golden-section_search`). We used this method to draw the exponent (for relative weights $\tau \leq (q-1)/q(1-R)$) of the ISD with Dumer's algorithm given in Figures 3.4, 3.7 and 3.8 . Furthermore we multiplied the above formula by a term $\log_2(q)$ to get exponents in base 2.

Strategy to reach large solutions. Let us suppose that $q > 2$. Otherwise we can symmetrize the complexity of the algorithm from the short case as shown in Exercise 4. Contrary to the strategy to get short solutions, if one wants to use an ISD to compute solutions with a large weight, one has to choose p as $k+\ell$ (see the discussion in the beginning of this section entitled "Far or close codeword"). Therefore, with Dumer's algorithm we will choose parameters such that

$$\lambda = \frac{R+\lambda}{2} h_q\left(\frac{\pi}{R+\lambda}\right) \text{ and } \pi = R+\lambda$$

which leads to (as $h_q(1) = \log_q(q-1)$),

$$\lambda = \frac{R+\lambda}{2}\log_q(q-1) \iff \lambda = \frac{R}{2}\,\frac{\log_q(q-1)}{1-\frac{1}{2}\log_q(q-1)}$$

However if one uses this strategy directly with Dumer's algorithm it would lead to very high exponent as build lists of size $q^{\lambda n}$ would be too large. The idea (before using Wagner's algorithm as we are going to do) is to change Dumer's algorithm and to use the variation given in Exercise 5. Suppose that one build lists of size S in Dumer's algorithm. Then, according to Proposition 8, the complexity of the ISD with this algorithm is given (up to polynomial factor by) (we fixed p to $k+\ell$)

$$\left(S + \frac{S^2}{q^\ell}\right) \cdot \max\left(1, \frac{\min\left(q^{n-k}, \binom{n}{t}(q-1)^t\right)}{\binom{n-k-\ell}{t-k-\ell}(q-1)^{t-k-\ell}\, S^2}\right)$$

Let $\sigma \stackrel{\text{def}}{=} \frac{1}{n}\log_q S$. Using this algorithm leads to the following asymptotic complexity

$$g(\lambda,\sigma) \stackrel{\text{def}}{=} \max(\sigma, 2\sigma-\lambda) + \max\left(0, \min(1-R, h_q(\tau)) - (1-R-\lambda)h_q\left(\frac{\tau-R-\lambda}{1-R-\lambda}\right) - 2\sigma\right) \tag{3.26}$$

However we do not have to forget that we have a constraint on the size of built lists, namely $S \leq \binom{(k+\ell)/2}{p/2}(q-1)^{p/2}$, therefore σ has necessarily to verify

$$\sigma \leq \frac{R+\lambda}{2} h_q\left(\frac{\pi}{R+\lambda}\right). \tag{3.27}$$

To optimize (3.26) we used to golden section search to first finding $\sigma(\lambda)$ "minimizing" (according to the method) $\sigma \mapsto g(\lambda,\sigma)$ for a fixed λ and σ verifying Constraint (3.27). Then we also used the golden section search to "minimize" $\lambda \mapsto g(\lambda,\sigma(\lambda))$. We draw in Figures 3.7 and 3.8 the exponent of Prange and the ISD with Dumer's algorithm for a fixed rate and as function of τ. As we see Dumer's algorithm provides an improvement over Prange's algorithm. Even if the improvement seems slight, don't forget that it means an *exponential* improvement as we draw exponents.

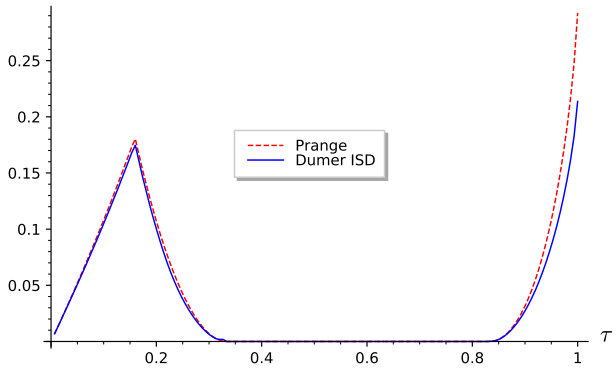


FIGURE 3.7. Exponent in base 2 of Prange's algorithm and ISD with Dumer's algorithm (in base 2) to solve $\mathsf{DP}(n,q,R,\tau)$ for $q=3$ and $R=1/2$ as function of $\tau \in [0,1]$.

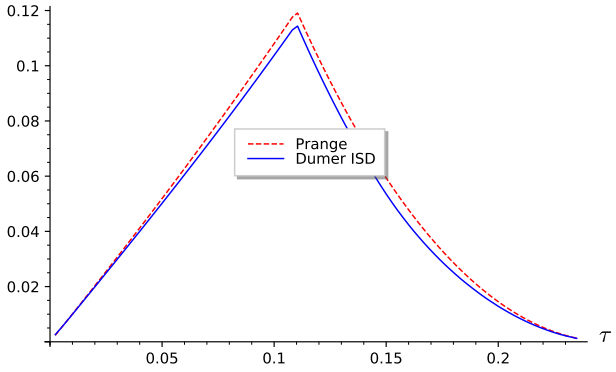


FIGURE 3.8. Exponent in base 2 of Prange's algorithm and ISD with Dumer's algorithm (in base 2) to solve $\mathsf{DP}(n,q,R,\tau)$ for $q=2$ and $R=1/2$ as function of $\tau \in \left[0, \frac{q-1}{q}(1-R)\right]$.

3.2. **ISD with Wagner's algorithm.** We are now ready to instantiate an ISD with Wagner's algorithm as a subroutine. In this case we choose to parametrize the algorithm to output solutions in amortized time one. Combining Proposition 8 and Proposition 5 (assertion (2)) lead to the following proposition

Proposition 10. *The complexity $C_{\text{Dumer}}(n,q,R,\tau)$ of the* ISD *using Wagner's algorithm (described in Subsection 2.2) to solve $\mathsf{DP}(n,q,R,\tau)$ is up to a polynomial factor (in n) given by*

$$q^{\frac{\ell}{a}} \cdot \max\left(1, \frac{\min\left(q^{n-k-\ell}, \binom{n}{t}(q-1)^t q^{-\ell}\right)}{\binom{n-k-\ell}{t-p}(q-1)^{t-p}\, q^{\frac{\ell}{a}}}\right) \tag{3.28}$$

where a is the largest integer such that $q^{\frac{\ell}{a}} \leq \binom{(k+\ell)/2^a}{p/2^a}(q-1)^{p/2^a}$

We used this proposition (with the same kind of strategy that above) to draw the exponent of the ISD with Wagner's algorithm. As we can see in Figure 3.4 the ISD with Wagner's algorithm has far better exponent compared to the ISD with Dumer's algorithm for large weight; otherwise exponents are the same.

References

[AAB+17] Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Edoardo Persichetti, and Gilles Zémor. HQC, November 2017. NIST Round 1 submission for Post-Quantum Cryptography.

[Ale03] Alekhnovich, Michael. More on Average Case vs Approximation Complexity. In *44th Symposium on Foundations of Computer Science (FOCS 2003), 11-14 October 2003, Cambridge, MA, USA, Proceedings*, pages 298–307. IEEE Computer Society, 2003.

[BJMM12] Anja Becker, Antoine Joux, Alexander May, and Alexander Meurer. Decoding random binary linear codes in $2^{n/20}$: How $1+1 = 0$ improves information set decoding. In *Advances in Cryptology - EUROCRYPT 2012*, LNCS. Springer, 2012.

[BM17] Leif Both and Alexander May. Optimizing BJMM with Nearest Neighbors: Full Decoding in $2^{2/21n}$ and McEliece Security. In *WCC Workshop on Coding and Cryptography*, September 2017.

[CS16] Rodolfo Canto-Torres and Nicolas Sendrier. Analysis of information set decoding for a sub-linear error weight. In *Post-Quantum Cryptography 2016*, LNCS, pages 144–161, Fukuoka, Japan, February 2016.

[Dum86] Ilya Dumer. 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*, pages 157–159, Leningrad, 1986.

[FS09] Matthieu Finiasz and Nicolas Sendrier. Security bounds for the design of code-based cryptosystems. In M. Matsui, editor, *Advances in Cryptology - ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 88–105. Springer, 2009.

[McE78] Robert J. McEliece. *A Public-Key System Based on Algebraic Coding Theory*, pages 114–116. Jet Propulsion Lab, 1978. DSN Progress Report 44.

[MO15] Alexander May and Ilya Ozerov. On computing nearest neighbors with applications to decoding of binary linear codes. In E. Oswald and M. Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015*, volume 9056 of *LNCS*, pages 203–228. Springer, 2015.

[MS09] L. Minder and A. Sinclair. The extended k-tree algorithm. In C. Mathieu, editor, *Proceedings of SODA 2009*, pages 586–595. SIAM, 2009.

[MTSB13] Rafael Misoczki, Jean-Pierre Tillich, Nicolas Sendrier, and Paulo S. L. M. Barreto. MDPC-McEliece: New McEliece variants from moderate density parity-check codes. In *Proc. IEEE Int. Symposium Inf. Theory - ISIT*, pages 2069–2073, 2013.

[Pra62] Eugene Prange. The use of information sets in decoding cyclic codes. *IRE Transactions on Information Theory*, 8(5):5–9, 1962.

[Wag02] David Wagner. A generalized birthday problem. In Moti Yung, editor, *Advances in Cryptology - CRYPTO 2002*, volume 2442 of *LNCS*, pages 288–303. Springer, 2002.