Projective Space Stern Decoding and Application to SDitH

Kevin Carrier[1], Valerian Hatey[1], and Jean-Pierre Tillich[2]

[1] ETIS UMR 8051 - Cergy-Paris Université, ENSEA, CNRS,
kevin.carrier@cyu.fr,valerian.hatey@ensea.fr
[2] Project COSMIQ, Inria de Paris, jean-pierre.tillich@inria.fr *

Abstract. We show that here standard decoding algorithms for generic linear codes over a finite field can speeded up by a factor which is essentially the size of the finite field by reducing it to a low weight codeword problem and working in the relevant projective space. We apply this technique to SDitH and show that the parameters of both the original submission and the updated version fall short of meeting the security requirements asked by the NIST.

1 Introduction

Code-based cryptography is based on the hardness of the decoding problem. In its syndrome (and fixed weight) version it is given for the Hamming metric (where we denote the Hamming weight of a vector $\mathbf{x}$ by $|\mathbf{x}|$) by

Problem 1.1 (Syndrome Decoding $\mathsf{SD}(\mathbf{H}, \mathbf{s}, t)$). *Given a matrix $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$, a syndrome $\mathbf{s} \in \mathbb{F}_q^{n-k}$ and a weight $t \in [\![0, n]\!]$, find a vector $\mathbf{e} \in \mathbb{F}_q^n$ such that $\mathbf{He} = \mathbf{s}$ and $|\mathbf{e}| = t$.*

In other words, it consists in solving a linear system with a constraint on the weight of the solution. This non-linear constraint is commonly believed to make the problem difficult on average over $\mathbf{H}$ for suitable values of t. Despite that many efforts have been spent over the last 60 years [Pra62, Ste88, Dum91, MMT11, BJMM12, MO15, BM17, BM18, CDMT22], the problem remains hard in the range of parameters given above, even with the help of a quantum computer [Ber10, KT17]. Thus, the decoding problem has raised interest among cryptosystem designers. It is today the heart of the security of PKE and signature schemes submitted to the NIST competitions[3] such as Classic McEliece [AAB+22], BIKE [ABC+22], Wave [BCC+23] and SDitH [AMFG+23]. It is quite common to study the binary version of the decoding problem but the non-binary case also aroused interest [BLP10, BLP11] or more recently with the signature schemes Wave [DST19] or SDitH [FJR22] for instance. The security of Wave is

* The work of KC, VH and JPT was funded by the French Agence Nationale de la Recherche through ANR JCJC DECODE (ANR-22-CE39-0004-01) for KC and VH and ANR-22-PETQ-0008 PQ-TLS for JPT.

[3] https://csrc.nist.gov/projects/post-quantum-cryptography

based on ternary codes and SDitH addresses the syndrome decoding over the fields $\mathbb{F}_{256}$ and $\mathbb{F}_{251}$. In this article, we focus on the case where q is large, as is the case inSDitH.

The best known decoding algorithm are Information Set Decoding (ISD) initiated by Prange in [Pra62]. The idea of Prange basically consists in guessing that $\mathbf{e}$ is zero on an *information set*, that is a set of k positions that determines the whole vector $\mathbf{e}$ considering the linear relation $\mathbf{He} = \mathbf{s}$. If the guess does not allow to find a vector of weight t, then we repeat the process changing the information set until we make the right guess on it.

There have been numerous improvements to the Prange algorithm. One of the first breakthrough in this domain uses the *birthday paradox* [Ste88, Dum91]. Basically, the ISD template is used here to reduce the decoding problem to a collision search. Later, other techniques were introduced to improve ISD. For instance, [MMT11] and [BJMM12] exploit the fact that a low weight vector can be represented in multiple ways as the sum of two lower weight vectors, it is the so-called *representation technique* introduced by Howgrave-Graham and Joux in [HJ10]. In the SDitH specifications, it is noticed that the representation technique was originally designed for the binary case and that it loses its interest when q is large. This claim is supported by Meurer in his PhD thesis [Meu12] and also by Canto-Torres in [Can17] where he shows that the MMT [MMT11] and BJMM [BJMM12] complexity exponent tend to the Prange complexity exponent when q tends to infinity. This is the reason why the algorithm on which SDitH focuses on is Stern [Ste88] which is considered optimal by the authors of SDitH in their particular context.

1.1 Our contribution

Our main observation here is that decoding over a big field can basically be speeded up by a factor which is the size of the field by a simple homogenizing trick (or what is the same by a reduction to the low weight codeword search problem). The idea is that instead of looking for a vector $\mathbf{e}$ of weight t satisfying $\mathbf{He} = \mathbf{s}$ we look for a vector $\mathbf{x}$ of weight t such that $\mathbf{Hx}$ is *proportional* to $\mathbf{s}$, *i.e.* is such that $\mathbf{Hx} = \lambda\mathbf{s}$ for some $\lambda \in \mathbb{F}_q$. If we find such a vector (and if $\lambda \neq 0$ which will happen with large probability as we will see in what follows) then we get from such an $\mathbf{x}$ our $\mathbf{e}$ by taking $\mathbf{e} = \lambda^{-1}\mathbf{x}$. The point is that basically all the collision search techniques used for solving the decoding problem get speeded by essentially a factor $q-1$ by identifying all vectors which are proportional. This is particularly helpful in the case where we work with big field sizes as is the case for the NIST submission SDitH [AMFG+23]. We will adapt this idea to one of the simplest collision decoding technique, namely Stern's decoding algorithm over $\mathbb{F}_q$ [Pet10]. We call this variant, *projective Stern's algorithm* since we work here essentially in the projective space. We provide here a clean counting of the complexity of this variant of Stern's algorithm in the spirit of [Pet11, Ch. 6]. This precise complexity counting includes the use of the Canteaut-Chabaud technique [CC98] to gain in the complexity of Gaussian elimination. This part can not be neglected at all for giving tight security estimates in the case of SDitH

because the list sizes in an optimal Stern algorithm are in this case really small. In SDitH there is also a variant of the decoding which is considered, which is the d-split variant: the support of the error is split into d equal parts and it asked to find an error $\mathbf{e}$ of weight t/d on each of the part. We give an adaptation of our projective Stern's algorithm to this case too.

We will study in detail the impact of this technique to SDitH both for the initial submission [AMFG+23] and for the recent update that can be found on `https://sdith.org/`. The initial submission was unfortunately affected by a mistake in the choice of parameters that corresponded to a region where there were several hundred of solutions to the decoding problem whereas the analysis implicitly assumed that there was just one. The security claims made in [AMFG+23] were incorrect because of this. The new algorithm presented here also reduces the security of this proposal. All in all, this shows that the security of the initial submission [AMFG+23] is below the NIST requirements by 9 to 14 bits depending on the SDitH variant. Three days after preliminary results of this work were made public [CTH23], new parameters of SDitH were released and announced on the NIST forum (see `https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/OOnB655mCN8/m/rL4bPD20AAAJ`). This new parameter set corrected the initial error in the parameter choice (now the parameters are chosen such that there is typically just one solution to the decoding problem). The authors took a 4 bit security margin between the NIST security requirements and the estimate for the best attack provided in `https://sdith.org/`. We show here that this is still a little bit short of meeting the NIST requirements by roughly one bit. It should be noted that contrarily to [AMFG+23] which uses (i) a non tight reduction from standard decoding to d-split decoding which gives an overestimate on the attacks, (ii) neglects the cost of Gaussian elimination in the attack, our security estimate is based on a precise count of the complexity of the attack which does not neglect the cost of Gaussian elimination. It turns out that the optimal parameters for the projective Stern algorithm are in the regime where the cost of Gaussian elimination is non negligible.

2 Preliminaries

Vectors and matrices. Vectors and matrices are respectively denoted in bold letters and bold capital letters such as $\mathbf{v}$ and $\mathbf{M}$. The entry at index i of the vector $\mathbf{v}$ is denoted by v_i. $\mathbf{M}^\top$ stands for the transpose of the matrix $\mathbf{M}$. To limit the use of transposition notation as much as possible, we consider in this paper that the vectors are column vectors; so $\mathbf{v}^\top$ represents a row vector. Let I be a list of indexes. We denote by $\mathbf{v}_I$ the vector $(v_i)_{i\in I}$. In the same way, we denote by $\mathbf{M}_I$ the submatrix made up of the columns of $\mathbf{M}$ which are indexed by I. The notation $\mathrm{supp}(\mathbf{v})$ stands for the support of $\mathbf{v}$, that is the set of the non-zero positions of $\mathbf{v}$.

The double square brackets stand for a set of consecutive integers. For instance, $[\![a, b]\!]$ are the integers between a and b.

Coding background. A linear code $\mathcal{C}$ of length n and dimension k over the field $\mathbb{F}_q$ is a subspace of $\mathbb{F}_q^n$ of dimension k. We say that it is an $[n,k]_q$-code. It can be defined by a *generator matrix* $\mathbf{G} \in \mathbb{F}_q^{k\times n}$ whose rows form a basis of the code:

$$\mathcal{C} \stackrel{\text{def}}{=} \left\{\mathbf{G}^\intercal \mathbf{u} \ : \ \mathbf{u} \in \mathbb{F}_q^k\right\}. \tag{2.1}$$

A *parity-check matrix* for $\mathcal{C}$ is a matrix $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ whose right kernel is $\mathcal{C}$:

$$\mathcal{C} \stackrel{\text{def}}{=} \left\{\mathbf{c} \in \mathbb{F}_q^n \ : \ \mathbf{H}\mathbf{c} = \mathbf{0}\right\}. \tag{2.2}$$

A set of k positions that fully defines a code $\mathcal{C}$ is called an *information set*. In other words, for $I \subseteq [\![1,n]\!]$ such that $|I| = k$ and $J \stackrel{\text{def}}{=} [\![1,n]\!] \setminus I$, the subset I is an information set if and only if $\mathbf{G}_I$ is invertible or equivalently $\mathbf{H}_J$ is invertible. In that case, J is called *redundancy set*.

In this paper, we address the decoding problem 1.1. We focus on the case where the decoding distance t is lower than $n - \frac{n}{q}$. We distinguish two particular regimes: when the decoding problem has typically less than one solution and when it has more. For $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ and $\mathbf{s} \in \mathbb{F}_q^{n-k}$ that are drawn uniformly at random, the greatest distance t for which the decoding problem has less than one solution on expectation is called the *Gilbert-Varshamov distance* and it is denoted by:

$$d_{\mathsf{GV}}(n,k) \stackrel{\text{def}}{=} \sup\left(\left\{t \in \left[\!\left[0, n - \tfrac{n}{q}\right]\!\right] \ : \ \binom{n}{t}(q-1)^t \leqslant q^{n-k}\right\}\right) \tag{2.3}$$

How we measure complexities. Because one of our goals is to compare our results to those of the SDitH specifications, we measure the complexities in the same way as they do. In particular, we assume that the additions and multiplications in $\mathbb{F}_q$ are implemented using lookup tables and that these two operations therefore have the same cost. In the SDitH specifications, this cost is considered as $\log_2(q)$ which is the estimated cost of a memory access. In the following, we count the complexities in number of additions/multiplications and therefore we ignore the factor $\log_2(q)$. However, the results of Section 7 are given with this factor.

3 The Stern decoder and Peters' improvements

In this section, we recall the main results of [Ste88] and [Pet10] that are used in SDitH specifications to solve the decoding problem $\mathsf{SD}(\mathbf{H},\mathbf{s},t)$. Stern's decoding algorithm is an iterative algorithm that is parametrized by two integers p and ℓ to optimize. Each iteration starts by selecting an information set $I \subseteq [\![1,n]\!]$ of size k. We denote $J \stackrel{\text{def}}{=} [\![1,n]\!] \setminus I$. Then, we search for $\mathbf{x} \in \mathbb{F}_q^k$ of weight $2p$ such that:

$$|\mathbf{P}\mathbf{x} - \mathbf{y}| = t - 2p \tag{3.1}$$

where

$$\mathbf{P} \stackrel{\text{def}}{=} \mathbf{H}_J^{-1}\mathbf{H}_I \qquad \text{and} \qquad \mathbf{y} \stackrel{\text{def}}{=} \mathbf{H}_J^{-1}\mathbf{s} \tag{3.2}$$

We can easily verify that if we find such an $\mathbf{x}$, then the vector $\mathbf{e} \in \mathbb{F}_q^n$ defined by $\mathbf{e}_I \stackrel{\text{def}}{=} \mathbf{x}$ and $\mathbf{e}_J \stackrel{\text{def}}{=} \mathbf{y} - \mathbf{P}\mathbf{x}$ is a solution to the decoding problem. By making the additional bet that the sought error $\mathbf{e}$ is such that $\mathbf{e}_I$ is of weight p on each half and $\mathbf{e}_J$ is $\mathbf{0}$ on its ℓ first positions, we can use collision search to find $\mathbf{e}$ more efficiently. Indeed, using lookup tables, one can find all pairs $(\mathbf{x}_1, \mathbf{x}_2) \in \mathbb{F}_q^k \times \mathbb{F}_q^k$ such that $\mathbf{x}_1$ is zero on its second half (resp. $\mathbf{x}_2$ is zero on its first half), $|\mathbf{x}_1| = p$ (resp. $|\mathbf{x}_2| = p$) and $\mathbf{P}\mathbf{x}_1 - \mathbf{y}$ and $\mathbf{P}\mathbf{x}_2$ collide on their ℓ first positions. Thus, for each of these collisions, the vector $\mathbf{e}$ such that $\mathbf{e}_I \stackrel{\text{def}}{=} \mathbf{x}_1 + \mathbf{x}_2$ and $\mathbf{e}_J \stackrel{\text{def}}{=} \mathbf{y} - \mathbf{P}(\mathbf{x}_1 + \mathbf{x}_2)$ is a potential solution to the SD problem because it has syndrome $\mathbf{s}$ and it is of particular low weight on at least $k+\ell$ positions. Finally, Algorithm 1 summarizes the Stern decoder.

Algorithm 1: Stern's algorithm to solve $\mathsf{SD}(\mathbf{H}, \mathbf{s}, t)$

Input: $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$, $\mathbf{s} \in \mathbb{F}_q^{n-k}$ and $t \in [\![0, n]\!]$.
Parameters: $p \in \left[\!\left[0, \frac{\min(t,k)}{2}\right]\!\right]$ and $\ell \in [\![0, n-k-t+2p]\!]$.
Output: $\mathbf{e} \in \mathbb{F}_q^n$ such that $\mathbf{H}\mathbf{e} = \mathbf{s}$ and $|\mathbf{e}| = t$.

repeat as many times as necessary
  draw $I \subseteq [\![1, n]\!]$ of size k uniformly at random
  $J \leftarrow [\![1, n]\!] \setminus I$
  $\mathbf{P} \leftarrow \mathbf{H}_J^{-1}\mathbf{H}_I$ `/* if` $\mathbf{H}_J$ `is not invertible, go back to step 2 */`
  $\mathbf{y} \leftarrow \mathbf{H}_J^{-1}\mathbf{s}$
  $\mathbf{R} \leftarrow$ the ℓ first rows of $\mathbf{P}$
  $\mathbf{z} \leftarrow$ the ℓ first positions of $\mathbf{s}$
  $\mathscr{L}_1 \leftarrow \left\{\mathbf{R}\mathbf{x}_1 - \mathbf{z} \,:\, \mathbf{x}_1 \in \mathbb{F}_q^{\lfloor k/2 \rfloor} \times 0^{k-\lfloor k/2 \rfloor} \text{ and } |\mathbf{x}_1| = p\right\}$
  $\mathscr{L}_2 \leftarrow \left\{\mathbf{R}\mathbf{x}_2 \,:\, \mathbf{x}_2 \in 0^{\lfloor k/2 \rfloor} \times \mathbb{F}_q^{k-\lfloor k/2 \rfloor} \text{ and } |\mathbf{x}_2| = p\right\}$
  forall $(\mathbf{R}\mathbf{x}_1 - \mathbf{z}, \mathbf{R}\mathbf{x}_2) \in \mathscr{L}_1 \times \mathscr{L}_2$ *such that* $\mathbf{R}\mathbf{x}_1 - \mathbf{z} = \mathbf{R}\mathbf{x}_2$ **do**
    if $|\mathbf{P}(\mathbf{x}_1 - \mathbf{x}_2) - \mathbf{y}| = t - 2p$ **then**
      return $\mathbf{e}$ such that $\mathbf{e}_I \stackrel{\text{def}}{=} \mathbf{x}_1 - \mathbf{x}_2$ and $\mathbf{e}_J \stackrel{\text{def}}{=} \mathbf{y} - \mathbf{P}(\mathbf{x}_1 - \mathbf{x}_2)$

Note that if a particular error vector $\mathbf{e}$ has the good weight distribution – that is $\mathbf{e}_I$ is of weight p on each half, $\mathbf{e}_J$ is $\mathbf{0}$ on its ℓ first positions and of weight $t-2p$ on its $n-k-\ell$ other positions – then Stern's algorithm will find $\mathbf{e}$. So the probability to find a particular solution is

$$p_{\text{part}} = \frac{\binom{\lfloor k/2 \rfloor}{p}\binom{k-\lfloor k/2 \rfloor}{p}\binom{n-k-\ell}{t-2p}}{\binom{n}{t}} \tag{3.3}$$

Moreover, in the case where $\mathbf{s}$ has been produced as the syndrome of an error $\widetilde{\mathbf{e}}$ of weight t – that means $\mathbf{s}$ has been drawn uniformly at random in $\{\mathbf{H}\widetilde{\mathbf{e}} \,:\, |\widetilde{\mathbf{e}}| = t\}$ – then, the expected number of solutions to the decoding problem we address is

$$N_{\text{sol}} \stackrel{\text{def}}{=} \mathbb{E}_{\mathbf{H}}\left(\left|\left\{\mathbf{e} \in \mathbb{F}_q^n \,:\, |\mathbf{e}| = t \text{ and } \mathbf{He} = \mathbf{H}\widetilde{\mathbf{e}}\right\}\right|\right) \tag{3.4}$$

$$= 1 + \sum_{\substack{\mathbf{e} \in \mathbb{F}_q^n \setminus \{\widetilde{\mathbf{e}}\} \\ |\mathbf{e}| = t}} \mathbb{P}_{\mathbf{H}}\left(\mathbf{He} = \mathbf{H}\widetilde{\mathbf{e}}\right) \tag{3.5}$$

$$= 1 + \frac{\binom{n}{t}(q-1)^t - 1}{q^{n-k}} \tag{3.6}$$

Thus, the success probability of one iteration of Stern's algorithm is

$$p_{\text{succ}} = 1 - (1 - p_{\text{part}})^{N_{\text{sol}}} \tag{3.7}$$

So on average over the choice of $\mathbf{H}$, we need to repeat Stern's procedure $\frac{1}{p_{\text{succ}}}$ times before finding a solution to the decoding problem. To determine the complexity of Stern's algorithm, we still have to measure the time complexity of one iteration. Using some of the tricks proposed in [Pet10], the designers of SDitH claim to be able to perform each iteration of Stern with a running time

$$\begin{aligned} T_{\text{iter}} = {} & \tfrac{1}{2}(n-k)^2(n+k) \\ & + \ell\left(\tfrac{k}{2} - p + 1 + \left(\binom{\lfloor k/2 \rfloor}{p} + \binom{k - \lfloor k/2 \rfloor}{p}\right)(q-1)^p\right) \\ & + \tfrac{q}{q-1}(t - 2p + 1)2p\left(1 + \tfrac{q-2}{q-1}\right)\frac{\binom{\lfloor k/2 \rfloor}{p}\binom{k-\lfloor k/2 \rfloor}{p}(q-1)^{2p}}{q^\ell} \end{aligned} \tag{3.8}$$

4 Reducing the decoding problem to the low weight codeword search

Working in projective spaces is only interesting if the syndrome is zero. In that case, we are actually looking for a low weight codeword instead of an error vector. This can be readily achieved by a well known reduction from decoding in an $[n,k]_q$ linear code to a low weight codeword search in an $[n,k+1]_q$ linear code that we now recall. Let $\mathbf{H}$ be a parity check matrix of a code $\mathcal{C}$. Without loss of generality, we can consider that $\mathbf{H}$ is in systematic form[4]:

$$\mathbf{H} \stackrel{\text{def}}{=} [\mathbf{A}|\mathbf{I}_{n-k}] \qquad \text{where } \mathbf{A} \in \mathbb{F}_q^{(n-k)\times k} \tag{4.1}$$

To solve the decoding problem $\mathsf{SD}(\mathbf{H}, \mathbf{s}, t)$, one can find a low weight codeword in the new code

$$\mathcal{C}' \stackrel{\text{def}}{=} \langle \mathcal{C}, \mathbf{z} \rangle \stackrel{\text{def}}{=} \{\mathbf{c} + \alpha\mathbf{z} \,:\, \mathbf{c} \in \mathcal{C} \text{ and } \alpha \in \mathbb{F}_q\} \tag{4.2}$$

[4] The first operation of Stern's algorithm precisely consists in putting the parity-check matrix in systematic form (up to a permutation). And therefore making this assumption does not induce any additional cost.

where $\mathbf{z} \in \mathbb{F}_q^n$ is any solution of the equation $\mathbf{Hz} = \mathbf{s}$ (without any weight constraint on $\mathbf{z}$). Because $\mathbf{H}$ is in systematic form, we can take $\mathbf{z}^\top \stackrel{\text{def}}{=} (\mathbf{0}^\top, \mathbf{s}^\top) \in \mathbb{F}_q^n$. Then a generator matrix of $\mathcal{C}'$ is

$$\mathbf{G}' \stackrel{\text{def}}{=} \begin{bmatrix} \mathbf{I}_k & -\mathbf{A}^\top \\ \mathbf{0}^\top & \mathbf{s}^\top \end{bmatrix} \tag{4.3}$$

By only one step of a Gaussian elimination (one column to eliminate), we can find a parity-check matrix $\mathbf{H}'$ of the augmented code $\mathcal{C}'$.

By looking for a low weight codeword in $\mathcal{C}'$ – *i.e.* a vector $\mathbf{e}$ such that $\mathbf{H}'\mathbf{e} = \mathbf{0}$ –, we actually find a low weight error $\mathbf{e}$ of $\mathcal{C}$ that has syndrome $\mathbf{He} = \alpha\mathbf{s}$ where α can be any scalar in $\mathbb{F}_q$. There are two possible situations: either $\alpha = 0$ or $\alpha \neq 0$. If $\alpha = 0$ then we have actually found a codeword in $\mathcal{C}$ instead of an error vector (we want to avoid this situation). On the contrary, if $\alpha \neq 0$ then a solution to our original decoding problem is simply $\alpha^{-1}\mathbf{e}$. We now claim that the probability to get $\alpha = 0$ is lower than $\frac{1}{q}$:

Theorem 4.1. *Let a code $\mathcal{C}$ be the right kernel of a parity-check matrix $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ and let $\mathbf{s} \in \{\mathbf{H}\widetilde{\mathbf{e}} : |\widetilde{\mathbf{e}}| = t\}$ for $t \in [\![0, n]\!]$. Let $\mathcal{C}' \stackrel{def}{=} \langle \mathcal{C}, \mathbf{z} \rangle$ be the code generated by the codewords in $\mathcal{C}$ and any word $\mathbf{z} \in \mathbb{F}_q^n$ such that $\mathbf{Hz} = \mathbf{s}$. We denote by $\mathbf{H}'$ a parity-check matrix of this augmented code. Then we can solve the decoding problem $\mathsf{SD}(\mathbf{H}, \mathbf{s}, t)$ by solving, on average over $\mathbf{H}$, at most $\frac{q}{q-1}$ low weight codeword searches $\mathsf{SD}(\mathbf{H}', \mathbf{0}, t)$.*

Proof. First, by construction of $\mathbf{s}$ and $\mathbf{z}$, we know there exists a codeword $\mathbf{c} \in \mathcal{C}$ and an error vector $\widetilde{\mathbf{e}} \in \mathbb{F}_q^n$ of weight t such that $\mathbf{z} = \mathbf{c} + \widetilde{\mathbf{e}}$. So we have $\langle \mathcal{C}, \mathbf{z} \rangle = \langle \mathcal{C}, \widetilde{\mathbf{e}} \rangle$ and $\mathbf{s} = \mathbf{H}\widetilde{\mathbf{e}}$.

Let $\mathbf{e}$ be any solution of the low weight codeword search problem $\mathsf{SD}(\mathbf{H}', \mathbf{0}, t)$. Then, as said before, $\mathbf{e}$ is an error vector of weight t such that $\mathbf{He} = \alpha\mathbf{s}$ for a scalar $\alpha \in \mathbb{F}_q$, and $\mathbf{e}$ induces a solution of the original decoding problem if and only if $\alpha \neq 0$. On average over $\mathbf{H}$ and for $\mathbf{e}$ drawn uniformly at random in the solutions of $\mathsf{SD}(\mathbf{H}', \mathbf{0}, t)$, the probability that $\alpha = 0$ is

$$\mathbb{P}_{\mathbf{H},\mathbf{e}}(\alpha = 0) = \frac{\mathbb{E}_{\mathbf{H}}(|\mathcal{C}(t)|)}{\mathbb{E}_{\mathbf{H}}(|\mathcal{C}'(t)|)}$$

where

$$\mathcal{C}(t) \stackrel{\text{def}}{=} \{\mathbf{e} \in \mathcal{C} \ : \ |\mathbf{e}| = t\}$$
$$\mathcal{C}'(t) \stackrel{\text{def}}{=} \{\mathbf{e} \in \mathcal{C}' \ : \ |\mathbf{e}| = t\}$$

We already know that

$$\mathbb{E}_{\mathbf{H}}(|\mathcal{C}(t)|) = \frac{\binom{n}{t}(q-1)^t}{q^{n-k}}.$$

Let us count $\mathcal{C}'(t)$. We remark that $\mathcal{C}'$ is the disjoint union of the cosets $\mathcal{C} + \alpha\widetilde{\mathbf{e}}$ for all the $\alpha \in \mathbb{F}_q$. So

$$\mathbb{E}_{\mathbf{H}}(|\mathcal{C}'(t)|) = \sum_{\alpha \in \mathbb{F}_q} \mathbb{E}_{\mathbf{H}}(|\mathcal{C}'_\alpha(t)|)$$

where

$$\mathcal{C}'_\alpha(t) \stackrel{\text{def}}{=} \{\mathbf{e} + \alpha\widetilde{\mathbf{e}} \,:\, \mathbf{e} \in \mathcal{C} \text{ and } |\mathbf{e} + \alpha\widetilde{\mathbf{e}}| = t\}\,.$$

On one hand, we have that $\mathcal{C}'_0(t) = \mathcal{C}(t)$. On another hand, for all non-zero $\alpha, \alpha' \in \mathbb{F}_q^*$, the map $\mathbf{x} \longmapsto \alpha^{-1}\alpha'\mathbf{x}$ is a bijection from $\mathcal{C}'_\alpha(t)$ to $\mathcal{C}'_{\alpha'}(t)$; so for all $\alpha \in \mathbb{F}_q^*$, $|\mathcal{C}'_\alpha(t)| = |\mathcal{C}'_1(t)|$. Thus, we have

$$\mathbb{E}_{\mathbf{H}}\left(|\mathcal{C}'(t)|\right) = \mathbb{E}_{\mathbf{H}}\left(|\mathcal{C}(t)|\right) + (q-1)\mathbb{E}_{\mathbf{H}}\left(|\mathcal{C}'_1(t)|\right)$$

By doing a calculation similar to that of the Equation (3.6), we can show that

$$\begin{aligned}\mathbb{E}_{\mathbf{H}}\left(|\mathcal{C}'_1(t)|\right) &= 1 + \tfrac{\binom{n}{t}(q-1)^t - 1}{q^{n-k}} \\ &= 1 - \tfrac{1}{q^{n-k}} + \mathbb{E}_{\mathbf{H}}\left(|\mathcal{C}(t)|\right).\end{aligned}$$

and so

$$\mathbb{E}_{\mathbf{H}}\left(|\mathcal{C}'(t)|\right) = q\mathbb{E}_{\mathbf{H}}\left(|\mathcal{C}(t)|\right) + (q-1)\left(1 - \tfrac{1}{q^{n-k}}\right).$$

Finally, the probability that the reduction succeeds is

$$1 - \mathbb{P}_{\mathbf{H},\mathbf{e}}\left(\alpha = 0\right) = 1 - \frac{1}{q + \frac{(q-1)(1-1/q^{n-k})}{\mathbb{E}_{\mathbf{H}}(|\mathcal{C}(t)|)}} \geqslant \frac{q-1}{q}.$$

□

5 Stern's algorithm in projective space

In Section 4, we gave a reduction of decoding to low weight codeword searching. In this section, we address the second problem, that is given a parity-check matrix $\mathbf{H}' \in \mathbb{F}_q^{(n-k-1)\times n}$ of $\mathcal{C}'$, we want to find $\mathbf{e} \in \mathbb{F}_q^n$ such that $|\mathbf{e}| = t$ and $\mathbf{H}'\mathbf{e} = \mathbf{0}$.

We have to be careful about the distribution of $\mathbf{H}'$ which has not been drawn uniformly at random in $\mathbb{F}_q^{(n-k-1)\times n}$. Indeed, for an error $\widetilde{\mathbf{e}} \in \mathbb{F}_q^n$ of weight t, $\mathbf{H}'$ has been drawn such that $\widetilde{\mathbf{e}}$ is a codeword in $\mathcal{C}'$, so $\mathbf{H}'$ verifies $\mathbf{H}'\widetilde{\mathbf{e}} = \mathbf{0}$.
Essentially, our method consists in running a Stern procedure in the projective space $\mathbb{F}_q^n/\!\sim$. In particular, we show that Peters' improvements of Stern [Pet11, Ch. 6] are still applicable in the projective space.

5.1 The algorithm

When the syndrome is zero, Stern's algorithm essentially consists in finding pairs $(\mathbf{x}_1, \mathbf{x}_2) \in \mathbb{F}_q^{k+1} \times \mathbb{F}_q^{k+1}$ such that $\mathbf{x}_1$ (resp. $\mathbf{x}_2$) is of weight p on its first $\lfloor \frac{k+1}{2} \rfloor$ (resp. last $k+1-\lfloor \frac{k+1}{2} \rfloor$) positions, zero elsewhere and

$$\mathbf{R}'\mathbf{x}_1 = \mathbf{R}'\mathbf{x}_2 \tag{5.1}$$

where $\mathbf{R}'$ are the ℓ first rows of $\mathbf{P}' \stackrel{\text{def}}{=} \mathbf{H}'^{-1}_J\mathbf{H}'_I$. Each pair $(\mathbf{x}_1, \mathbf{x}_2)$ that collides gives a candidate codeword $\mathbf{e}$ defined by

$$\mathbf{e}_I = \mathbf{x}_1 - \mathbf{x}_2 \qquad \text{and} \qquad \mathbf{e}_J = \mathbf{P}'(\mathbf{x}_1 - \mathbf{x}_2). \tag{5.2}$$

One can remark that if the pair $(\mathbf{x}_1, \mathbf{x}_2)$ is a solution to the collision search, then for all $\alpha \in \mathbb{F}_q^*$, $\alpha\mathbf{x}_1$ and $\alpha\mathbf{x}_2$ also collide.

Remark 5.1. Note that this trick is specific to the fact that the syndrome is zero. If the syndrome is non-zero, then given a pair $(\mathbf{x}_1, \mathbf{x}_2)$ that is such that $\mathbf{R}'\mathbf{x}_1 - \mathbf{y}' = \mathbf{R}'\mathbf{x}_2$, we can no longer guarantee that for any non-zero α, we still have $\alpha\mathbf{R}'\mathbf{x}_1 - \mathbf{y}' = \alpha\mathbf{R}'\mathbf{x}_2$. That is the reason why the reduction in Section 4 is essential.

Moreover, $\alpha\mathbf{x}_1$ and $\alpha\mathbf{x}_2$ respectively share the same support as $\mathbf{x}_1$ and $\mathbf{x}_2$ so the Stern procedure enumerates all the collinear equivalents of $\mathbf{x}_1$ and $\mathbf{x}_2$ and consequently, it explores all the candidate codewords that are collinear to $\mathbf{e}$. However, we only need one of them. Indeed, if $\mathbf{e}$ is in $\mathcal{C}'$ but not in $\mathcal{C}$ – that is $\mathbf{H}'\mathbf{e} = \mathbf{0}$ but $\mathbf{H}\mathbf{e} \neq \mathbf{0}$ – then there is a unique $\alpha \in \mathbb{F}_q^*$ such that the syndrome $\alpha\mathbf{H}\mathbf{e}$ is exactly $\mathbf{s}$ and not a multiple of it. The solution $\alpha\mathbf{e}$ can be found from any vector that is collinear to $\mathbf{e}$ and so, we only need to find one of them.

From the discussion above, we remark that when the syndrome is zero, Stern's algorithm can be run in the projective space. For a space $\mathcal{E}$ over $\mathbb{F}_q$, the projective space $\mathcal{E}/_\sim$ is the quotient set of $\mathcal{E}$ by the *equivalence relation* $\sim$:

$$\forall \mathbf{x}, \mathbf{y} \in \mathcal{E}, \quad \mathbf{x} \sim \mathbf{y} \iff \exists \alpha \in \mathbb{F}_q^*, \ \mathbf{x} = \alpha\mathbf{y}. \tag{5.3}$$

The *equivalence class* of a vector $\mathbf{x} \in \mathcal{E}$ is denoted by:

$$[\mathbf{x}] \stackrel{\text{def}}{=} \{\mathbf{y} \in \mathcal{E} \ : \ \mathbf{y} \sim \mathbf{x}\}. \tag{5.4}$$

And so

$$\mathcal{E}/_\sim \stackrel{\text{def}}{=} \{[\mathbf{x}] \ : \ \mathbf{x} \in \mathcal{E}\}. \tag{5.5}$$

Now, if $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{F}_q^{k+1}$ are such that $\mathbf{R}'\mathbf{x}_1 = \mathbf{R}'\mathbf{x}_2$ then we also have $\mathbf{R}'[\mathbf{x}_1] = \mathbf{R}'[\mathbf{x}_2]$ where $[\mathbf{x}_1]$ and $[\mathbf{x}_2]$ live in $\mathbb{F}_q^{k+1}/_\sim$. But we have to note that if $\mathbf{R}'[\mathbf{x}_1] = \mathbf{R}'[\mathbf{x}_2]$ then we do not necessarily have $\mathbf{R}'\mathbf{x}_1 = \mathbf{R}'\mathbf{x}_2$. So we need to choose representatives $\overline{\mathbf{x}_1} \in [\mathbf{x}_1]$ and $\overline{\mathbf{x}_2} \in [\mathbf{x}_2]$ which guarantee

$$\mathbf{R}'\overline{\mathbf{x}_1} = \mathbf{R}'\overline{\mathbf{x}_2} \tag{5.6}$$

To do that, we distinguish a particular class representative:

Definition 5.2 (Particular class representative). *Let* $\mathbf{R}' \in \mathbb{F}_q^{\ell \times (k+1)}$. *For all* $[\mathbf{x}] \in \mathbb{F}_q^{k+1}/_\sim$, *if* $\mathbf{R}'\mathbf{x} = \mathbf{0}$ *then the vector* $\overline{\mathbf{x}}$ *is any representative of* $[\mathbf{x}]$, *otherwise it is the unique representative of* $[\mathbf{x}]$ *such that the first non-zero symbol of* $\mathbf{R}'\overline{\mathbf{x}}$ *is* 1.

Lemma 5.3. *For any* $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{F}_q^{k+1}$,

$$\mathbf{R}'[\mathbf{x}_1] = \mathbf{R}'[\mathbf{x}_2] \iff \mathbf{R}'\overline{\mathbf{x}_1} = \mathbf{R}'\overline{\mathbf{x}_2}. \tag{5.7}$$

Proof. If $\mathbf{R}'\mathbf{x}_1 = \mathbf{R}'\mathbf{x}_2 = \mathbf{0}$ then the proof is trivial. Otherwise, for either $i = 1$ or 2, $\mathbf{R}'[\mathbf{x}_i]$ is made of all the vectors that are collinear to $\mathbf{R}'\mathbf{x}_i$. Thus, all the elements in $\mathbf{R}'[\mathbf{x}_i]$ share the same support, in particular they have the same first non-zero position, and there is a unique vector in $\mathbf{R}'[\mathbf{x}_i]$ for which this first non-zero position contains a 1. So, we first notice that $\overline{\mathbf{x}_1}$ and $\overline{\mathbf{x}_2}$ exist and they are unique.

Assume $\mathbf{R}'[\mathbf{x}_1] = \mathbf{R}'[\mathbf{x}_2]$. That means $\mathbf{R}'\overline{\mathbf{x}_1} \in \mathbf{R}'[\mathbf{x}_2]$. On another hand, the first non-zero symbol in $\mathbf{R}'\overline{\mathbf{x}_1}$ is a one and the only element of this kind in $\mathbf{R}'[\mathbf{x}_2]$ is $\mathbf{R}'\overline{\mathbf{x}_2}$, so we necessarily have $\mathbf{R}'\overline{\mathbf{x}_1} = \mathbf{R}'\overline{\mathbf{x}_2}$.

Conversely, $\mathbf{R}'\overline{\mathbf{x}_1} = \mathbf{R}'\overline{\mathbf{x}_2} \Rightarrow [\mathbf{R}'\overline{\mathbf{x}_1}] = [\mathbf{R}'\overline{\mathbf{x}_2}] \Rightarrow \mathbf{R}'[\mathbf{x}_1] = \mathbf{R}'[\mathbf{x}_2]$.

□

Now we are ready to describe our adaptation of Stern's algorithm to the projective space. Algorithm 2 gives the pseudo code of the method. Note that unlike Algorithm 1, here the syndrome is zero and $\overline{\mathbf{x}_1}, \overline{\mathbf{x}_2} \in \mathbb{F}_q^{k+1}$ are some particular representatives of $[\mathbf{x}_1], [\mathbf{x}_2] \in \mathbb{F}_q^{k+1}/\sim$. Moreover, we must treat differently the case where $\mathbf{R}'\overline{\mathbf{x}_1} = \mathbf{R}'\overline{\mathbf{x}_2} = \mathbf{0}$ because this case generates $q-1$ collisions that are not collinear with each other. Lemma 5.3 guarantees that a collision in projective space is still a collision when using the good representative so that guarantees the correctness of the algorithm.

Algorithm 2: Projective Stern's algorithm to solve $\mathsf{SD}(\mathbf{H}', \mathbf{0}, t)$

Input: $\mathbf{H}' \in \mathbb{F}_q^{(n-k-1)\times n}$ and $t \in [\![0, n]\!]$.

Parameters: $p \in \left[\!\left[0, \frac{\min(t,k+1)}{2}\right]\!\right]$ and $\ell \in [\![0, n-k-1-t+2p]\!]$.

Output: $\mathbf{e} \in \mathbb{F}_q^n$ such that $\mathbf{H}'\mathbf{e} = \mathbf{0}$ and $|\mathbf{e}| = t$.

repeat as many times as necessary
  draw $I \subseteq [\![1, n]\!]$ of size $k+1$ uniformly at random
  $J \leftarrow [\![1, n]\!] \setminus I$
  $\mathbf{P}' \leftarrow \mathbf{H}'^{-1}_J \mathbf{H}'_I$ `/* if` $\mathbf{H}'_J$ `is not invertible, go back to step 2 */`
  $\mathbf{R}' \leftarrow$ the ℓ first rows of $\mathbf{P}'$
  $\mathscr{L}'_1 \leftarrow \left\{ \mathbf{R}'\overline{\mathbf{x}_1} \,:\, [\mathbf{x}_1] \in \left(\mathbb{F}_q^{\lfloor \frac{k+1}{2} \rfloor} \times 0^{k+1-\lfloor \frac{k+1}{2} \rfloor} \right)/\sim \text{ and } |\mathbf{x}_1| = p \right\}$
  $\mathscr{L}'_2 \leftarrow \left\{ \mathbf{R}'\overline{\mathbf{x}_2} \,:\, [\mathbf{x}_2] \in \left(0^{\lfloor \frac{k+1}{2} \rfloor} \times \mathbb{F}_q^{k+1-\lfloor \frac{k+1}{2} \rfloor} \right)/\sim \text{ and } |\mathbf{x}_2| = p \right\}$
  forall $(\mathbf{R}'\overline{\mathbf{x}_1}, \mathbf{R}'\overline{\mathbf{x}_2}) \in \mathscr{L}'_1 \times \mathscr{L}'_2$ *such that* $\mathbf{R}'\overline{\mathbf{x}_1} = \mathbf{R}'\overline{\mathbf{x}_2}$ **do**
    if $\mathbf{R}'\overline{\mathbf{x}_1} = \mathbf{0}$ **and** $\exists \alpha \in \mathbb{F}_q^*, |\mathbf{P}'(\alpha\overline{\mathbf{x}_1} - \overline{\mathbf{x}_2})| = t - 2p$ **then**
      return $\mathbf{e}$ such that $\mathbf{e}_I = \alpha\overline{\mathbf{x}_1} - \overline{\mathbf{x}_2}$ and $\mathbf{e}_J^\intercal = \mathbf{P}'(\alpha\overline{\mathbf{x}_1} - \overline{\mathbf{x}_2})$
    else if $|\mathbf{P}'(\overline{\mathbf{x}_1} - \overline{\mathbf{x}_2})| = t - 2p$ **then**
      return $\mathbf{e}$ such that $\mathbf{e}_I = \overline{\mathbf{x}_1} - \overline{\mathbf{x}_2}$ and $\mathbf{e}_J^\intercal = \mathbf{P}'(\overline{\mathbf{x}_1} - \overline{\mathbf{x}_2})$

5.2 Reducing the cost of Gaussian elimination

For large q and p, the Gaussian elimination step is negligible and so, we can afford to perform it on $n-k-1$ columns drawn independently at each iteration. Thus, Gaussian elimination needs $(n-k-1)^2(n+k+2)$ operations. However, in the context of SDitH, we are far away from this regime and the Gaussian elimination is actually one of the most expensive operation we have to perform. In this sub-section, we present two modifications of our original projective Stern algorithm that allow to reduce the impact of the Gaussian elimination step. Those tricks are inspired by [Pet10] and [BLP08] and have been adapted to our situation.

Factorizing the Gaussian elimination step. An iteration of Algorithm 2 begins with selecting an information set I and a window of size ℓ. Let denote by I_1 (resp. I_2) the first half of I (resp. the second half of I) and J_ℓ the ℓ first positions of $J \stackrel{\text{def}}{=} [\![1,n]\!] \setminus I$. The iteration succeeds in finding the particular error vector $\mathbf{e}$ of weight t if it verifies

$$|\mathbf{e}_{I_1}| = |\mathbf{e}_{I_2}| = p \qquad \text{and} \qquad |\mathbf{e}_{J_\ell}| = 0. \tag{5.8}$$

To save some Gaussian elimination steps, we can test several partitions (I_1, I_2, J_ℓ) for one given information set. In other words, the main loop in Algorithm 2 can be divided into an *outer loop* and an *inner loop*. The *outer loop* consists in selecting an information set I and performing a Gaussian elimination on it (steps 2-4). The *inner loop* starts by partitioning I into (I_1, I_2) and selecting a window $J_\ell \subset J$ of size ℓ, then it performs the steps 5-12 with

$$\mathbf{R}' \stackrel{\text{def}}{=} \text{ The rows of } \mathbf{P}' \text{ indexed by } J_\ell \tag{5.9}$$

$$\mathscr{L}_1' \stackrel{\text{def}}{=} \left\{ \mathbf{R}'\overline{\mathbf{x}_1} \,:\, [\mathbf{x}_1] \in \mathbb{F}_q^{k+1}/\sim \text{ and } \operatorname{supp}(\mathbf{x}_1) \subseteq I_1 \text{ and } |\mathbf{x}_1| = p \right\} \tag{5.10}$$

$$\mathscr{L}_2' \stackrel{\text{def}}{=} \left\{ \mathbf{R}'\overline{\mathbf{x}_2} \,:\, [\mathbf{x}_2] \in \mathbb{F}_q^{k+1}/\sim \text{ and } \operatorname{supp}(\mathbf{x}_2) \subseteq I_2 \text{ and } |\mathbf{x}_2| = p \right\} \tag{5.11}$$

For a given information set I, we choose the partition (I_1, I_2, J_ℓ) uniformly at random and independently from one iteration to another. Assuming we are looking for a t-weight codeword $\mathbf{e} \in \mathcal{C}'$ that verifies $|\mathbf{e}_I| = 2p$, then the success probability of finding this particular codeword during an iteration of the inner loop is

$$q_{\text{in}} = \frac{\binom{\lfloor \frac{k+1}{2} \rfloor}{p}\binom{k+1-\lfloor \frac{k+1}{2} \rfloor}{p}\binom{n-k-1-\ell}{t-2p}}{\binom{k+1}{2p}\binom{n-k-1}{t-2p}}. \tag{5.12}$$

So the number of trials needed to get $\mathbf{e}$ follows a geometric distribution of parameter q_{in} and so, by iterating $N_{\text{in}}^{\text{tmp}}$ times the inner loop, we will find $\mathbf{e}$ with probability

$$p_{\text{in}} \stackrel{\text{def}}{=} 1 - (1 - q_{\text{in}})^{N_{\text{in}}^{\text{tmp}}} \tag{5.13}$$

Note that taking

$$N_{\text{in}}^{\text{tmp}} \stackrel{\text{def}}{=} \frac{1}{q_{\text{in}}} \tag{5.14}$$

allows to achieve a success probability p_{in} for the inner loop that is exponentially close to 1.

Reusing pivots in the Gaussian elimination. In [CC98], Canteaut and Chabaud propose to simplify the Gaussian elimination step by changing only one index in the information set I. Thus, only one pivot is necessary from one iteration of the outer loop to another. This idea is generalized in [BLP08] where this time, the number of columns to eliminate from one iteration to another can be greater than 1. By doing this, we reduce the cost of Gaussian elimination but we also induce some dependencies between the selected information sets that impact the number of iterations of the outer loop that is needed.

To estimate the impact of the technique described above, we lean on the analysis in [BLP08, Pet10]. We first introduce the parameter c which represents the number of columns to eliminate in each iteration[5]. Thus, the cost of Gaussian elimination per iteration of the outer loop is

$$T_{\text{Gauss}} = 2\sum_{i=1}^{c}(n-k-1)(k+1+i) \tag{5.15}$$

$$= c(n-k-1)(2k+c+3) \tag{5.16}$$

Note that if a t-weight codeword $\mathbf{e} \in \mathcal{C}'$ is such that $|\mathbf{e}_I| = 2p$, then the corresponding iteration of the outer loop will find a representative of $[\mathbf{e}]$ with probability p_{in}. So we need to count the average number of iterations of the outer loop that we need for having this particular weight distribution. However, there are some dependencies between the iterations that must be taken into account. Indeed, we do not draw the $k+1$ positions of the information set independently from one iteration to another ($k+1-c$ positions are kept).

The situation can be modeled by a $(t+2)$-state absorbing Markov chain. Given a t-weight codeword $\mathbf{e} \in \mathcal{C}'$, let X_i be the random variable that represents the weight of $\mathbf{e}_I$ at iteration $i \in \mathbb{N}$ of the outer loop or "Done" if the previous iteration succeeds. For the first iteration, the information set I is chosen uniformly at random as a subset of $[\![1,n]\!]$ of size k. So the distribution of X_0 is given by

$$\forall v \in [\![0,t]\!], \quad \mathbb{P}(X_0 = v) = \frac{\binom{k+1}{v}\binom{n-k-1}{t-v}}{\binom{n}{t}} \qquad \text{and} \qquad \mathbb{P}(X_0 = \mathsf{Done}) = 0. \tag{5.17}$$

Let $\mathbf{\Pi}$ be the transition matrix of the Markov chain. It is defined as the following stochastic matrix:

$$\forall (u,v) \in \{\mathsf{Done}, 0, \cdots, t\}^2, \ \ \mathbf{\Pi}[u,v] \stackrel{\text{def}}{=} \mathbb{P}(X_{i+1} = v \mid X_i = u) \tag{5.18}$$

The state Done is the absorbing state, that means when we are in this state, we cannot get out anymore. So we have

$$\forall v \in [\![0,t]\!], \ \ \mathbf{\Pi}[\mathsf{Done}, v] = 0 \qquad \text{and} \qquad \mathbf{\Pi}[\mathsf{Done}, \mathsf{Done}] = 1. \tag{5.19}$$

[5] In [BLP08], another parameter r is introduced but its interest is only for small field.

From an iteration to another, the information set I is updated by swapping c indexes drawn uniformly at random in I with c indexes drawn uniformly at random in J. So an iteration moves from state u to state v with probability

$$\mathbf{\Pi}[u,v] = \sum_j \frac{\binom{u}{j}\binom{k+1-u}{c-j}\binom{t-u}{v-u+j}\binom{n-k-1-t+u}{c-v+u-j}}{\binom{k+1}{c}\binom{n-k-1}{c}} \tag{5.20}$$

except for $u = 2p$ because then the algorithm succeeds with probability:

$$\mathbf{\Pi}[2p, \mathsf{Done}] = p_{\text{in}}. \tag{5.21}$$

So for all $v \in [\![0,t]\!]$:

$$\mathbf{\Pi}[2p,v] = (1-p_{\text{in}}) \cdot \sum_j \frac{\binom{2p}{j}\binom{k+1-2p}{c-j}\binom{t-2p}{v-2p+j}\binom{n-k-1-t+2p}{c-v+2p-j}}{\binom{k+1}{c}\binom{n-k-1}{c}}. \tag{5.22}$$

Finally, to determine the number of iterations needed to get the first success, one only has to compute the fundamental matrix associated to $\mathbf{\Pi}$:

$$\mathbf{F} \stackrel{\text{def}}{=} (\mathbf{I}_{t+1} - \mathbf{\Pi}')^{-1} \tag{5.23}$$

where $\mathbf{I}_{t+1}$ is the identity matrix of size $t+1$ and $\mathbf{\Pi}'$ is the $(t+1) \times (t+1)$ sub-matrix of $\mathbf{\Pi}$ such that

$$\forall (u,v) \in \{0,\cdots,t\}^2, \;\; \mathbf{\Pi}'[u,v] \stackrel{\text{def}}{=} \mathbf{\Pi}[u,v]. \tag{5.24}$$

Then, the average number of iterations of the outer loop needed to find a representative of $[\mathbf{e}]$ is

$$N_{\text{out}}^{\text{tmp}} = \sum_{u=0}^{t} \sum_{v=0}^{t} \mathbb{P}(X_0 = v)\, \mathbf{F}[u,v]. \tag{5.25}$$

Finding one solution from many. With $N_{\text{out}}^{\text{tmp}} \cdot N_{\text{in}}^{\text{tmp}}$ repetitions of the inner loop, we are able to find one particular t-weight projective codeword $[\mathbf{e}] \in \mathcal{C}'/_{\sim}$. But there is potentially more than one such projective codeword since this number is

$$N_{\text{sol}} = \mathbb{E}_{\mathbf{H}'}\left(\left|\left\{[\mathbf{e}] \in \mathbb{F}_q^n/_{\sim} \;:\; |\mathbf{e}| = t \text{ and } \mathbf{H}'[\mathbf{e}] = [\mathbf{0}]\right\}\right|\right) \tag{5.26}$$

$$= 1 + \frac{\binom{n}{t}(q-1)^{t-1} - 1}{q^{n-k-1}} \tag{5.27}$$

Because we want to find only one solution from the N_{sol} ones, we actually need to approximately iterate the outer loop N_{out} times and for each iteration of the outer loop, we iterate the inner loop N_{in} times where

$$N_{\text{out}} \stackrel{\text{def}}{=} \max\left(1, \frac{N_{\text{out}}^{\text{tmp}}}{N_{\text{sol}}}\right) \tag{5.28}$$

$$N_{\text{in}} \stackrel{\text{def}}{=} \max\left(1, N_{\text{in}}^{\text{tmp}} \cdot \min\left(1, \frac{N_{\text{out}}^{\text{tmp}}}{N_{\text{sol}}}\right)\right). \tag{5.29}$$

5.3 Complexity of our projective Stern decoding

Finally, considering the modifications of the previous sub-section and using the implementation tricks of Peters [Pet10], we are able to state the following Theorem 5.4 that gives the complexity of Stern's algorithm in projective space.

Theorem 5.4. *Let $\widetilde{\mathbf{e}} \in \mathbb{F}_q^n$ be such that $|\mathbf{e}| = t$. On average over the choice of $\mathbf{H}' \in \mathbb{F}_q^{(n-k-1)\times n}$ that is such that $\mathbf{H}'\widetilde{\mathbf{e}} = \mathbf{0}$, we can solve the low weight codeword search problem $\mathsf{SD}(\mathbf{H}', \mathbf{0}, t)$ with a running time of order*

$$T_{\text{Stern-proj}} = N_{\text{out}}\Big(T_{\text{Gauss}} + N_{\text{in}}\big(T_{\text{lists}} + T_{\text{check}}\big)\Big) \tag{5.30}$$

where N_{out}, N_{in} and T_{Gauss} are given by Equations (5.28), (5.29) *and* (5.16), *and*

$$L_1 = \binom{\lfloor \frac{k+1}{2} \rfloor}{p}(q-1)^{p-1} \tag{5.31}$$

$$L_2 = \binom{k+1-\lfloor \frac{k+1}{2} \rfloor}{p}(q-1)^{p-1} \tag{5.32}$$

$$T_{\text{lists}} = \ell\left(k + 2p - 1 + 2\left(L_1 + L_2\right)\right) \tag{5.33}$$

$$N_{\text{collisions}} = \frac{(q-1)L_1L_2}{q^\ell} \tag{5.34}$$

$$T_{\text{check}} = \left(2p + \tfrac{q}{q-1}(t-2p+1)2p\left(1+\tfrac{q-2}{q-1}\right)\right)\cdot N_{\text{collision}} \tag{5.35}$$

Proof. According to the Sub-section 5.2, we have to iterate N_{out} times the outer loop which consists in a Gaussian elimination over c columns and N_{in} iterations of the inner loop. All that remains is to determine the cost of one iteration of the inner loop. To perform this iteration optimally, we will use Peters' implementation tricks [Pet10].

To build the lists $\mathscr{L}_1$ and $\mathscr{L}_2$, we need to define another representative of an equivalence class in $\mathbb{F}_q^{k+1}/\sim$. Let $\mathbf{x} \in \mathbb{F}_q^{k+1}$, we denote by $\widehat{\mathbf{x}}$, the representative of $[\mathbf{x}]$ whose first non-zero symbol is 1. Thus, for $\mathscr{L}_1$, we produce successively the representatives $\widehat{\mathbf{x}_1}$ that have a weight p on the first half and zero elsewhere using exactly the same trick as Peters (except we fix the first non-zero symbol to 1 and that there is no syndrome to add). So we can compute successively $\mathbf{P}'\widehat{\mathbf{x}_1}$ by only adding one column or two consecutive columns (except for the first element that needs $2p-2$ column additions). The single column additions allow to browse all the vectors of a same support and the two columns additions allow to move from one support to another. The two columns additions can actually be replaced by single column additions if we pre-compute all the sums of two consecutive columns in $\mathbf{P}'$. This costs $k+1$ column additions. By fixing the first non zero symbol to one, we browse only one representative per equivalence class. However, it is not the good representative that we defined in Definition 5.2. But it is quite easy to find the factor $\alpha \in \mathbb{F}_q^*$ such that $\overline{\mathbf{x}_1} = \alpha\widehat{\mathbf{x}_1}$ by multiplying one column by a scalar. Then, we do not compute $\overline{\mathbf{x}_1}$ but we save the pair $(\alpha, \widehat{\mathbf{x}_1})$

instead. We proceed similarly for the list $\mathscr{L}_2$. So finally, the cost of producing $\mathscr{L}_1$ and $\mathscr{L}_2$ is

$$T_{\text{lists}} = \ell(k+1) + \ell(2p-2) + 2\ell\left(\binom{\lfloor \frac{k+1}{2} \rfloor}{p} + \binom{k+1-\lfloor \frac{k+1}{2} \rfloor}{p}\right)(q-1)^{p-1} \tag{5.36}$$

$$= \ell\left(k + 2p - 1 + 2\left(\binom{\lfloor \frac{k+1}{2} \rfloor}{p} + \binom{k+1-\lfloor \frac{k+1}{2} \rfloor}{p}\right)(q-1)^{p-1}\right) \tag{5.37}$$

To deal with the collisions we first need to count them. On average, there are

$$N_{\text{collision}} = \frac{\binom{\lfloor \frac{k+1}{2} \rfloor}{p}\binom{k+1-\lfloor \frac{k+1}{2} \rfloor}{p}(q-1)^{2p-2}}{1 + (q^\ell - 1)/(q-1)} \tag{5.38}$$

$$\cdot \left((q-1)\mathbb{P}\left(\mathbf{P}'\overline{\mathbf{x}_1} = \mathbf{0}\right) + \mathbb{P}\left(\mathbf{P}'\overline{\mathbf{x}_1} \neq \mathbf{0}\right)\right) \tag{5.39}$$

$$= \frac{\binom{\lfloor \frac{k+1}{2} \rfloor}{p}\binom{k+1-\lfloor \frac{k+1}{2} \rfloor}{p}(q-1)^{2p-2}}{1 + (q^\ell - 1)/(q-1)} \cdot \left(\frac{q-1}{q^\ell} + 1 - \frac{1}{q^\ell}\right) \tag{5.40}$$

$$= \frac{\binom{\lfloor \frac{k+1}{2} \rfloor}{p}\binom{k+1-\lfloor \frac{k+1}{2} \rfloor}{p}(q-1)^{2p-1}}{q^\ell} \tag{5.41}$$

For each collision, we must first apply the scalar multiplication to change the representative of the equivalence class. This costs $2p$ multiplications. Then we can apply the same trick as Peters and check for only $\frac{q}{q-1}(t-2p+1)$ rows on average. The cost to treat a row is $2p$ additions and $2p\frac{q-2}{q-1}$ multiplications. So the cost to check all the candidates (each coming from a collision) is

$$T_{\text{check}} = \left(2p + \tfrac{q}{q-1}(t-2p+1)2p\left(1 + \tfrac{q-2}{q-1}\right)\right) \cdot N_{\text{collision}}. \tag{5.42}$$

□

6 The *d*-split decoding problem

The security of SDitH is actually based on the d-split decoding problem. Before stating this problem, let us bring in the following notation:

Notation 6.1. *Let $\mathbf{v} \in \mathbb{F}_q^n$. For d that divides n and for all $i \in [\![1, d]\!]$, we denote by $\mathbf{v}_{[i]}$ the i^{th} piece of $\mathbf{v}$ of length $\frac{n}{d}$. More formally,*

$$\mathbf{v}_{[i]} \overset{def}{=} \mathbf{v}_{[\![(i-1)\frac{n}{d}+1, i\frac{n}{d}]\!]} \overset{def}{=} (v_j)_{j \in [\![(i-1)\frac{n}{d}+1, i\frac{n}{d}]\!]} \tag{6.1}$$

Then the d-split syndrome decoding problem can be stated as follows:

Problem 6.2 (d-split Syndrome Decoding $\mathsf{SD}(d, \mathbf{H}, \mathbf{s}, t)$). *Given a matrix $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$, a syndrome $\mathbf{s} \in \mathbb{F}_q^{n-k}$ and a distance $t \in [\![0, n]\!]$, find a vector $\mathbf{e} \in \mathbb{F}_q^n$ such that $\mathbf{He} = \mathbf{s}$ and $\forall i \in [\![1, d]\!]$, $\left|\mathbf{e}_{[i]}\right| = t/d$.*

The d-split decoding problem is quite similar to the standard decoding problem but with the additional constraint that the error weight must be regularly distributed over d blocks. Note that the syndrome $\mathbf{s}$ was actually produced using an injected solution $\widetilde{\mathbf{e}}$ whose weight is precisely regularly distributed. In other word, there exists at least one $\widetilde{\mathbf{e}} \in \mathbb{F}_q^n$ such that $\mathbf{s} = \mathbf{H}\widetilde{\mathbf{e}}$ and $\forall i \in [\![1, d]\!]$, $\left|\widetilde{\mathbf{e}}_{[i]}\right| = t/d$.

In the SDitH specifications, the hardness of the d-split syndrome decoding problem is lower bounded by a quantity that depends on the complexity to solve the standard decoding problem. This lower bound is based on the following theorem:

Theorem 6.3 ([FJR22]). *Let $\mathbf{H}$ be drawn uniformly at random in $\mathbb{F}_q^{(n-k)\times n}$ and let $\mathbf{s} \in \mathbb{F}_q^n$. If an algorithm can find any particular solution of the d-split syndrome decoding problem $\mathsf{SD}(d, \mathbf{H}, \mathbf{s}, t)$ in time T with probability ε_d, then there is an algorithm that can find any particular solution of the syndrome decoding problem $\mathsf{SD}(1, \mathbf{H}, \mathbf{s}, t)$ in time T with probability ε_1 with*

$$\varepsilon_1 \geqslant \frac{\binom{n/d}{t/d}^d}{\binom{n}{t}} \varepsilon_d \tag{6.2}$$

Using Theorem 6.3, it can be argued that the best average complexity to solve $\mathsf{SD}(d, \mathbf{H}, \mathbf{s}, t)$ cannot be lower than $\binom{n/d}{t/d}^d / \binom{n}{t}$ times the best average complexity to solve $\mathsf{SD}(1, \mathbf{H}, \mathbf{s}, t)$. SDitH measures the difficulty of the d-split decoding problem with this lower bound on the complexity together with the complexity of the best known attack on $\mathsf{SD}(1, \mathbf{H}, \mathbf{s}, t)$. However, this bound is not tight and gives optimistic security levels. Indeed, it is considered here that we are only looking for a particular solution; the number of solutions to the problem is not taken into account. But recall that if there are many solutions, we just want to find one of them. We therefore state the following theorem that gives a more precise lower bound on the complexity we can expect to achieve:

Theorem 6.4. *Let $\mathbf{H}$ be drawn uniformly at random in $\mathbb{F}_q^{(n-k)\times n}$ and let $\mathbf{s} \in \mathbb{F}_q^n$. If an algorithm can solve $\mathsf{SD}(d, \mathbf{H}, \mathbf{s}, t)$ in time T with probability p_d, then there is an algorithm that can solve $\mathsf{SD}(1, \mathbf{H}, \mathbf{s}, t)$ in time T with probability p_1 with*

$$p_d \leqslant 1 - \left(1 - \frac{\binom{n}{t}}{\binom{n/d}{t/d}^d} \left(1 - (1 - p_1)^{1/N_1}\right)\right)^{N_d} \approx \frac{\binom{n}{t}}{\binom{n/d}{t/d}^d} \frac{N_d}{N_1} p_1 \tag{6.3}$$

where N_1 is the expected number of solutions to the problem $\mathsf{SD}(1, \mathbf{H}, \mathbf{s}, t)$ and N_d is the expected number of solutions to the problem $\mathsf{SD}(d, \mathbf{H}, \mathbf{s}, t)$.

Proof. Let $\mathscr{A}_d$ be an algorithm which finds a particular solution in $\mathsf{SD}(d, \mathbf{H}, \mathbf{s}, t)$ in times T with probability ε_d. From Theorem 6.3, there exists an algorithm $\mathscr{A}_1$ which finds a particular solution in $\mathsf{SD}(1, \mathbf{H}, \mathbf{s}, t)$ in times T with probability ε_1 with

$$1 - (1 - \varepsilon_d)^{N_d} \leqslant 1 - \left(1 - \frac{\binom{n}{t}}{\binom{n/d}{t/d}^d} \varepsilon_1\right)^{N_d}$$

We end the proof by noticing that

$$p_d = 1 - (1 - \varepsilon_d)^{N_d} \qquad \text{and} \qquad p_1 = 1 - (1 - \varepsilon_1)^{N_1}$$

□

In Theorem 6.4, since we address the d-split syndrome decoding problem where $\mathbf{s}$ is the syndrome of a d-split error vector of weight t, we have:

$$N_d = 1 + \frac{\binom{n/d}{t/d}^d (q-1)^t - 1}{q^{n-k}}. \tag{6.4}$$

Moreover, the algorithm $\mathscr{A}_1$ consists essentially in repeating the algorithm $\mathscr{A}_d$ by permuting the code randomly so this algorithm solves the standard decoding problem where $\mathbf{s}$ is the syndrome of any t-weight error vector. So we have:

$$N_1 = 1 + \frac{\binom{n}{t}^d (q-1)^t - 1}{q^{n-k}}. \tag{6.5}$$

Remark 6.5. Note that when t is smaller than the Gilbert-Varshamov distance, then the only solution to $\mathsf{SD}(1, \mathbf{H}, \mathbf{s}, t)$ is the injected solution and has a probability $\binom{n/d}{t/d}^d / \binom{n}{t}$ to be a solution for $\mathsf{SD}(d, \mathbf{H}, \mathbf{s}, t)$. So we get the same lower bound as in SDitH since $p_1 \approx \varepsilon_1$ and $p_d \approx \varepsilon_d$. On the contrary, if t is such that we have many solutions, then the injected solution has little impact and so we only have $p_d \leqslant p_1(1 + o(1))$. Note that when the number of solutions is large, p_d is close to p_1 because it is simpler to find a particular solution to the d-split decoding problem but there are also less solutions in proportion.

Adapting the projective Stern algorithm for d-split. Theorem 6.4 induces a lower bound on the complexity of d-split decoding, but we cannot guarantee that it is actually possible to reach this bound. It is possible to give an actual algorithm to solve the d-split decoding problem. Indeed, we can apply the reduction of Section 4 and adapt our projective Stern algorithm to take into account the regularity of the weight of the solution we are looking for. More precisely, at each iteration of Algorithm 2, one can choose the information set as

$$I \stackrel{\text{def}}{=} I_1 \cup \ldots \cup I_d \tag{6.6}$$

where each I_i is a subset of $[\![(i-1)\frac{n}{d} + 1, i\frac{n}{d}]\!]$ of size $\frac{k}{d}$. At each iteration, we bet that at least one sought solution $\mathbf{e}$ is such that for all $i \in [\![1, d]\!]$, $|\mathbf{e}_{I_i}| = \frac{p}{d}$.

Note that, in the context of SDitH, when adapting projective Stern to d-split, the optimal parameter p increases: it goes from 1 to 2. So the cost to produce the lists $\mathscr{L}_1$ and $\mathscr{L}_2$ increases quadratically. Consequently, the Gaussian elimination step becomes negligible, especially if we factorize it. It follows that the Canteaut-Chabaud technique is not relevant because it increases the number of iterations but it does not substantially reduce their cost. This is why finally, for $d > 1$, we do not use the Canteaut-Chabaud's trick. Appendix A, gives the formulas for the 2-split projective Stern algorithm.

7 Application to SDitH

In this section, we analyze the security of SDitH and we compare our results with those given in the SDitH specifications document [AMFG+23]. In the context of the NIST competition[6], the authors tried to reach different security levels:

- **category I:** at least 143 bits of security (≈ AES-128);
- **category III:** at least 207 bits of security (≈ AES-192);
- **category V:** at least 272 bits of security (≈ AES-256).

Table 7.1 summarizes the parameters proposed in the NIST submission [AMFG+23] of SDitH to achieve the above security levels.

Table 7.1: Parameters of SDitH v1.0 for various security levels.

Parameter sets	**NIST recommendations**		**d-split SD parameters**				
	category	target security	q	n	k	t	d
SDitH_L1_gf256_v1.0	I	143 bits	256	230	126	79	1
SDitH_L1_gf251_v1.0	I	143 bits	251	230	126	79	1
SDitH_L3_gf256_v1.0	III	207 bits	256	352	193	120	2
SDitH_L3_gf251_v1.0	III	207 bits	251	352	193	120	2
SDitH_L5_gf256_v1.0	V	272 bits	256	480	278	150	2
SDitH_L5_gf251_v1.0	V	272 bits	251	480	278	150	2

In the SDitH specifications, it is considered that the best algorithm to solve the decoding problem is Peters' version of Stern's algorithm of Section 3. However, there is a mistake in SDitH v1.0: it is considered that there is only one solution to the syndrome decoding problem when in fact there are several hundred for the parameters that have been chosen. Moreover, Theorem 6.3 which lower bounds the complexity of solving the d-split is not tight when there are many solutions. In Table 7.2 we give (i) the results claimed in the specifications of SDitH v1.0, (ii) the lower bound on the security when considering the multiple solutions and a tighter lower bound obtained from Theorem 6.4, (iii) the lower bound on the security we achieve with our projective Stern decoding used in conjunction with Theorem 6.4 and (iv) the security we achieve with the actual d-split Stern's algorithm described at the end of Section 6. The security is expressed in number of bits. Note that the last column corresponds to an actual attack on the scheme. Comparing (ii) and (iii), we can see in particular that the projective method is responsible for the loss of around 5 bits of security. In summary:

– even by improving the lower bound of [AMFG+23], this methodology for proving the security fails to meet the NIST requirements by around 11-14 bits, (column (iii))

[6] https://csrc.nist.gov/projects/post-quantum-cryptography

– there is an actual attack on the scheme showing that its complexity is below the NIST requirements by around 9-14 bits, (column (iv)).

Table 7.2: Security level of SDitH v1.0.

Parameter sets	**(i) Claimed in the specification document**			**(ii) Correction from Section 3 and Theorem 6.4**			**(iii) Projective Stern and Theorem 6.4**				**(iv) d-split projective Stern**		
	p	ℓ	security	p	ℓ	security	p	ℓ	c	security	p	ℓ	security
SDitH_L1_gf256_v1.0	1	2	$\geqslant 143.46$	1	2	$\geqslant 135.29$	1	2	1	$\geqslant$ **129.23**	1	2	**129.23**
SDitH_L1_gf251_v1.0	1	2	$\geqslant 143.45$	1	2	$\geqslant 134.58$	1	2	1	$\geqslant$ **128.52**	1	2	**128.52**
SDitH_L3_gf256_v1.0	2	5	$\geqslant 207.67$	2	5	$\geqslant 202.43$	1	2	1	$\geqslant$ **196.76**	2	4	**199.30**
SDitH_L3_gf251_v1.0	2	5	$\geqslant 207.61$	2	5	$\geqslant 201.30$	1	2	1	$\geqslant$ **195.68**	2	4	**198.19**
SDitH_L5_gf256_v1.0	2	5	$\geqslant 272.35$	2	5	$\geqslant 267.40$	1	2	1	$\geqslant$ **262.63**	2	4	**264.30**
SDitH_L5_gf251_v1.0	2	5	$\geqslant 272.29$	2	5	$\geqslant 265.91$	1	2	1	$\geqslant$ **261.19**	2	4	**262.84**

Recently, after we communicated preliminary results in [CTH23], the SDitH designers proposed a new version 1.1 in `https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/OOnB655mCN8/m/rL4bPD2OAAAJ` with updated parameters. Table 7.3 presents the new parameters. Then, Table 7.4 compares the security claimed in SDitH v1.1 and our projective Stern decoder on the updated parameters. Our projective Stern's algorithm is around 2^5 times faster than the complexity claimed in SDitH v1.1. However, the SDitH designers took a margin of error of 4 bits compared to the NIST recommendations and therefore our attack is only one bit under the NIST recommendations for the category I parameters. For the other parameters, the lower bound methodology of [AMFG+23] (even after improvement) fails to meet the NIST criterion by about one bit in all cases (column (iii)). It remains to see whether the attack on the d-split version can be improved (this is used in category III and V parameters) because our corresponding attack (column (iv)) is just one bit above the required security

Table 7.3: Parameters of the d-split decoding problem in SDitH v1.1 for various security levels.

Parameter sets	**NIST recommendations**		**d-split SD parameters**				
	category	target security	q	n	k	t	d
SDitH_L1_gf256_v1.1	I	143 bits	256	242	126	87	1
SDitH_L1_gf251_v1.1	I	143 bits	251	242	126	87	1
SDitH_L3_gf256_v1.1	III	207 bits	256	376	220	114	2
SDitH_L3_gf251_v1.1	III	207 bits	251	376	220	114	2
SDitH_L5_gf256_v1.1	V	272 bits	256	494	282	156	2
SDitH_L5_gf251_v1.1	V	272 bits	251	494	282	156	2

level. The SageMath program which made it possible to compute the results is available on `https://github.com/kevin-carrier/SDitH_security`.

Table 7.4: Security level of SDitH v1.1.

Parameter sets	**Target security**	**(i-ii) Claimed in the specification document**			**(iii) Projective Stern and Theorem 6.4**				**(iv) d-split projective Stern**		
		p	ℓ	security	p	ℓ	c	security	p	ℓ	security
`SDitH_L1_gf256_v1.1`	143	1	2	$\geqslant 147.73$	1	2	1	$\geqslant$ **141.54**	1	2	**141.54**
`SDitH_L1_gf251_v1.1`	143	1	2	$\geqslant 147.72$	1	2	1	$\geqslant$ **141.54**	1	2	**141.54**
`SDitH_L3_gf256_v1.1`	207	2	5	$\geqslant 211.05$	1	2	1	$\geqslant$ **205.59**	2	4	**207.90**
`SDitH_L3_gf251_v1.1`	207	2	5	$\geqslant 210.99$	1	2	1	$\geqslant$ **205.59**	2	4	**207.86**
`SDitH_L5_gf256_v1.1`	272	2	5	$\geqslant 276.33$	1	2	1	$\geqslant$ **271.69**	2	4	**273.26**
`SDitH_L5_gf251_v1.1`	272	2	5	$\geqslant 276.28$	1	2	1	$\geqslant$ **271.68**	2	4	**273.23**

References

AAB+22. Carlos Aguilar Melchor, Nicolas Aragon, Paulo Barreto, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Santosh Ghosh, Shay Gueron, Tim Güneysu, Rafael Misoczki, Edoardo Persichetti, Jan Richter-Brockmann, Nicolas Sendrier, Jean-Pierre Tillich, Valentin Vasseur, and Gilles Zémor. BIKE. Round 4 Submission to the NIST Post-Quantum Cryptography Call, v. 5.1, October 2022.

ABC+22. Martin Albrecht, Daniel J. Bernstein, Tung Chou, Carlos Cid, Jan Gilcher, Tanja Lange, Varun Maram, Ingo von Maurich, Rafael Mizoczki, Ruben Niederhagen, Edoardo Persichetti, Kenneth Paterson, Christiane Peters, Peter Schwabe, Nicolas Sendrier, Jakub Szefer, Cen Jung Tjhai, Martin Tomlinson, and Wang Wen. Classic McEliece (merger of Classic McEliece and NTS-KEM). `https://classic.mceliece.org`, November 2022. Fourth round finalist of the NIST post-quantum cryptography call.

AMFG+23. Carlos Aguilar Melchor, Thibauld Feneuil, Nicolas Gama, Shay Gueron, James Howe, David Joseph, Antoine Joux, Edoardo Persichetti, Tovohery Randrianarisoa, Matthieu Rivain, and Dongze Yue. SDitH. Round 1 Additional Signatures to the NIST Post-Quantum Cryptography: Digital Signature Schemes Call, May 2023.

BCC+23. Gustavo Banegas, K'evin Carrier, Andr'e Chailloux, Alain Couvreur, Thomas Debris-Alazard, Philippe Gaborit, Pierre Karpman, Johanna Loyer, Ruben Niederhagen, Nicolas Sendrier, Benjamin Smith, and Jean-Pierre Tillich. Wave. Round 1 Additional Signatures to the NIST Post-Quantum Cryptography: Digital Signature Schemes Call, June 2023.

Ber10. Daniel J. Bernstein. Grover vs. McEliece. In Nicolas Sendrier, editor, *Post-Quantum Cryptography 2010*, volume 6061 of *LNCS*, pages 73–80. Springer, 2010.

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.

BLP08. Daniel J. Bernstein, Tanja Lange, and Christiane Peters. Attacking and defending the McEliece cryptosystem. In *Post-Quantum Cryptography 2008*, volume 5299 of *LNCS*, pages 31–46, 2008.

BLP10. Daniel J. Bernstein, Tanja Lange, and Christiane Peters. Wild McEliece. In Alex Biryukov, Guang Gong, and Douglas R. Stinson, editors, *Selected Areas in Cryptography*, volume 6544 of *LNCS*, pages 143–158, 2010.

BLP11. Daniel J. Bernstein, Tanja Lange, and Christiane Peters. Wild McEliece Incognito. In Bo-Yin Yang, editor, *Post-Quantum Cryptography 2011*, volume 7071 of *LNCS*, pages 244–254. Springer Berlin Heidelberg, 2011.

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.

BM18. Leif Both and Alexander May. Decoding linear codes with high error rate and its impact for LPN security. In Tanja Lange and Rainer Steinwandt, editors, *Post-Quantum Cryptography 2018*, volume 10786 of *LNCS*, pages 25–46, Fort Lauderdale, FL, USA, April 2018. Springer.

Can17. Rodolfo Canto Torres. Asymptotic analysis of ISD algorithms for the $q-$ary case. In *Proceedings of the Tenth International Workshop on Coding and Cryptography WCC 2017*, September 2017.

CC98. Anne Canteaut and Florent Chabaud. A new algorithm for finding minimum-weight words in a linear code: Application to McEliece's cryptosystem and to narrow-sense BCH codes of length 511. *IEEE Trans. Inform. Theory*, 44(1):367–378, 1998.

CDMT22. Kevin Carrier, Thomas Debris-Alazard, Charles Meyer-Hilfiger, and Jean-Pierre Tillich. Statistical decoding 2.0: Reducing decoding to LPN. In *Advances in Cryptology - ASIACRYPT 2022*, LNCS. Springer, 2022.

CTH23. Kévin Carrier, Jean-Pierre Tillich, and Valerian Hatey. Security analysis of SDiTH. Slides of the one-day Workshop on Code-Based Cryptography, Nov. 20, 2023. See `https://inria.hal.science/hal-0431126`.

DST19. Thomas Debris-Alazard, Nicolas Sendrier, and Jean-Pierre Tillich. Wave: A new family of trapdoor one-way preimage sampleable functions based on codes. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology - ASIACRYPT 2019, Part I*, volume 11921 of *LNCS*, pages 21–51, Kobe, Japan, December 2019. Springer.

Dum91. Ilya Dumer. On minimum distance decoding of linear codes. In *Proc. 5th Joint Soviet-Swedish Int. Workshop Inform. Theory*, pages 50–52, Moscow, 1991.

FJR22. Thibauld Feneuil, Antoine Joux, and Matthieu Rivain. Syndrome decoding in the head: Shorter signatures from zero-knowledge proofs. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology - CRYPTO 2022*, volume 13508 of *LNCS*, pages 541–572. Springer, 2022.

HJ10. Nicholas Howgrave-Graham and Antoine Joux. New generic algorithms for hard knapsacks. In Henri Gilbert, editor, *Advances in Cryptology - EUROCRYPT 2010*, volume 6110 of *LNCS*. Sringer, 2010.

KT17. Ghazal Kachigar and Jean-Pierre Tillich. Quantum information set decoding algorithms. preprint, arXiv:1703.00263 [cs.CR], February 2017.

Meu12. Alexander Meurer. *A Coding-Theoretic Approach to Cryptanalysis*. PhD thesis, Ruhr University Bochum, November 2012.

MMT11. Alexander May, Alexander Meurer, and Enrico Thomae. Decoding random linear codes in $O(2^{0.054n})$. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology - ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 107–124. Springer, 2011.

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.

Pet10. Christiane Peters. Information-set decoding for linear codes over $\mathbf{F}_q$. In *Post-Quantum Cryptography 2010*, volume 6061 of *LNCS*, pages 81–94. Springer, 2010.

Pet11. Christiane Peters. *Curves, Codes, and Cryptography*. PhD thesis, Technische Universiteit Eindhoven, 2011.

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

Ste88. Jacques Stern. A method for finding codewords of small weight. In G. D. Cohen and J. Wolfmann, editors, *Coding Theory and Applications*, volume 388 of *LNCS*, pages 106–113. Springer, 1988.

A Formulas for the 2-split projective Stern algorithm

Here we address the 2-split decoding problem which is one of the instances of SDitH. In Section 6, we explained how to adapt our projective Stern's algorithm to d-split by splitting the support of the error into d equal parts and looking for an error $\mathbf{e}$ of weight t/d on each of the part.

We just give the formulas which differ from those we presented for the standard version 1-split in Section 5. In particular, for the 2-split version of our projective Stern's algorithm, Equations (5.12), (5.25), (5.16) (5.27), (5.31) and (5.32) become respectively:

$$\begin{aligned} q_{\text{in}} = & \frac{\binom{\lfloor (k+1)/4 \rfloor}{\lfloor p/2 \rfloor}\binom{\lfloor (k+1)/2 \rfloor - \lfloor (k+1)/4 \rfloor}{p - \lfloor p/2 \rfloor}}{\binom{\lfloor (k+1)/2 \rfloor}{p}} \\ & \cdot \frac{\binom{\lfloor (k+1)/2 \rfloor - \lfloor (k+1)/4 \rfloor}{\lfloor p/2 \rfloor}\binom{k+1-2\lfloor (k+1)/2 \rfloor + \lfloor (k+1)/4 \rfloor}{p - \lfloor p/2 \rfloor}}{\binom{k+1-\lfloor (k+1)/2 \rfloor}{p}} \\ & \cdot \frac{\binom{\lfloor (n-k-1)/2 \rfloor - \lfloor \ell/2 \rfloor}{t/2-p}\binom{n-k-1-\lfloor (n-k-1)/2 \rfloor - \ell + \lfloor \ell/2 \rfloor}{t/2-p}}{\binom{\lfloor (n-k-1)/2 \rfloor}{t/2-p}\binom{n-k-1-\lfloor (n-k-1)/2 \rfloor}{t/2-p}} \end{aligned} \tag{A.1}$$

$$N_{\text{out}}^{\text{tmp}} = \frac{\binom{\lfloor n/2 \rfloor}{\lfloor t/2 \rfloor}^2}{\binom{\lfloor (k+1)/2 \rfloor}{p}\binom{k+1-\lfloor (k+1)/2 \rfloor}{p}\binom{\lfloor (n-k-1)/2 \rfloor}{t/2-p}\binom{n-k-1-\lfloor (n-k-1)/2 \rfloor}{t/2-p}} \tag{A.2}$$

$$T_{\text{Gauss}} = 2(n-k-1)\sum_{i=1}^{n-k-1}(n-i+1) = (n-k-1)^2(n+k+2) \tag{A.3}$$

$$N_{\text{sol}} = 1 + \frac{\binom{n/2}{t/2}^2 (q-1)^{t-1} - 1}{q^{n-k-1}} \tag{A.4}$$

$$L_1 = \binom{\lfloor (k+1)/4 \rfloor}{\lfloor p/2 \rfloor}\binom{\lfloor (k+1)/2 \rfloor - \lfloor (k+1)/4 \rfloor}{p - \lfloor p/2 \rfloor}(q-1)^{p-1} \tag{A.5}$$

$$L_2 = \binom{\lfloor (k+1)/2 \rfloor - \lfloor (k+1)/4 \rfloor}{\lfloor p/2 \rfloor}\binom{k+1-2\lfloor (k+1)/2 \rfloor + \lfloor (k+1)/4 \rfloor}{p - \lfloor p/2 \rfloor}(q-1)^{p-1} \tag{A.6}$$