Ternary Syndrome Decoding with Large Weight

Rémi Bricout[1,2], André Chailloux[2], Thomas Debris-Alazard[1,2], and Matthieu Lequesne[1,2]

[1] Sorbonne Universités, UPMC Univ Paris 06
[2] Inria, Paris
{remi.bricout,andre.chailloux,thomas.debris,matthieu.lequesne}@inria.fr

Abstract. The Syndrome Decoding problem is at the core of many code-based cryptosystems. In this paper, we study ternary Syndrome Decoding in large weight. This problem has been introduced in the Wave signature scheme but has never been thoroughly studied. We perform an algorithmic study of this problem which results in an update of the Wave parameters. On a more fundamental level, we show that ternary Syndrome Decoding with large weight is a really harder problem than the binary Syndrome Decoding problem, which could have several applications for the design of code-based cryptosystems.

1 Introduction

Syndrome decoding is one of the oldest problems used in coding theory and cryptography [McE78]. It is known to be NP-complete [BMvT78] and its average case variant is still believed to be hard forty years after it was proposed, even against quantum computers. This makes code-based cryptography a credible candidate for post-quantum cryptography. There has been numerous proposals of post-quantum cryptosystems based on the hardness of the Syndrome Decoding (SD) problem, some of which where proposed for the NIST standardization process for quantum-resistant cryptographic schemes. Most of them are qualified for the second round of the competition [ABB+17, ACP+17, AMAB+17, BBC+19, BCL+17]. It is therefore a significant task to understand the computational hardness of the Syndrome Decoding problem.

Informally, the Syndrome Decoding problem is stated as follows. Given a matrix $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$, a vector $\mathbf{s} \in \mathbb{F}_q^{n-k}$ and a weight $w \in [\![0, n]\!]$, the goal is to find a vector $\mathbf{e} \in \mathbb{F}_q^n$ such that $\mathbf{H}\mathbf{e}^\intercal = \mathbf{s}^\intercal$ and $|\mathbf{e}| = w$, where $|\mathbf{e}|$ denotes the Hamming weight, namely $|\mathbf{e}| = |\{i : \mathbf{e}_i \neq 0\}|$. The binary case, *i.e.* when $q = 2$ has been extensively studied. Even before this problem was used in cryptography, Prange [Pra62] constructed a clever algorithm for solving the binary problem using a method now referred to as Information Set Decoding (ISD).

1.1 Binary vs. Ternary Case

The binary case of the SD problem has been thoroughly studied. Its complexity is always studied for relative weight $W \in [0, 0.5]$ because the case $W > 0.5$ is symmetric (see Remark 2). However, this argument is no longer valid in the general case ($q \geq 3$). Indeed, the large weight case does not behave similarly to the small weight case, as we can see on Figure 1.

The general case $q \geq 3$ has received much less attention than the binary case. One possible explanation for this is that there were no cryptographic applications for the general case. This has recently changed. Indeed, the new signature scheme Wave [DST18] based on the difficulty of SD on a ternary alphabet and with large weight was designed. This scheme makes uses of the new regime of large weight induced by the asymmetry of the ternary case for cryptographic applications. Therefore, on top of studying the general syndrome decoding for its algorithmic interest, we also have a real cryptosystem where this study can be applied.

Another reason why the general case $q \geq 3$ has been less studied is that the Hamming weight measure becomes less meaningful as q grows larger. Indeed, the Hamming weight only counts the number of non-zero elements but not their repartition. Hence, the weight loses a significant amount of information for large values of q. Therefore, $q = 3$ seems to be the best candidate to understand the structure of the non-binary case without losing too much information.

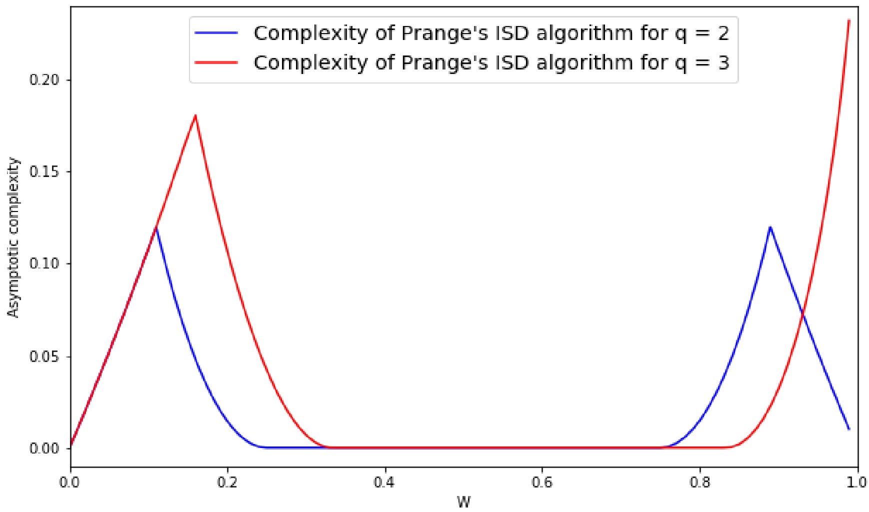

Fig. 1. Asymptotic complexity of Prange's ISD algorithm for $R = 0.5$.

1.2 State of the Art for $q \geq 3$

Still, there exist some interesting results concerning the SD problem in the general case. Coffey and Goodman [CG90] were the firsts to propose a generalization of Prange's ISD algorithm to $\mathbb{F}_q$. Following this seminal work, most existing ISD algorithms were extended to cover the q-ary case. In 2010, Peters [Pet10] generalized Stern's algorithm. In his dissertation thesis, Meurer [Meu17] generalized the BJMM algorithm. Hirose [Hir16] proposed a generalization of Stern's algorithm with May-Ozerov's approach (using nearest neighbors) and showed that for $q \geq 3$ this does not improve the complexity compared to Stern's classical approach. Later, Gueye, Klamti and Hirose [GKH17] extended the BJMM algorithm with May-Ozerov's approach and improved the complexity of the general SD problem. A result from Canto-Torres [CT17] proves that all ISD-based algorithms converge to the same asymptotical complexity when $q \to \infty$. Finally, a recent work [IKR$^+$18] proposed a generalization of the ball-collision decoding over $\mathbb{F}_q$.

All these papers focus solely on the SD problem for relative weight $W < 0.5$. None of them mentions the case of large weight. The claimed worst case complexities in these papers should be understood as the worst case complexity for the SD problem with relative weight $W < 0.5$, but as we can see on Figure 1 the highest complexity is actually reached for large relative weight.

1.3 Our Contributions

Our contribution is to perform a general study of ternary syndrome decoding with large weight.

The PGE+SS framework. A first minor contribution consists in a modular description of most ISD-based algorithms. All these algorithms contain two steps. First, performing a partial Gaussian elimination (PGE), and then, solving a variant of the Subset Sum problem (SS). This was already implicitly used in previous papers but we want to make it explicit to simplify the analysis and hopefully make those algorithms easier to understand for non-specialists.

Ternary SD with large weight. We then study specifically the SD problem in the ternary case and for large weights. From our modular description, we can focus only on finding many solutions of a specific instance of the Subset Sum problem. At a high level, we combine Wagner's algorithm [Wag02] and representation techniques [BCJ11, BJMM12] to obtain our algorithm. Our first take-away is that, while representations are very useful to obtain a unique solution (as in [BCJ11]), there are some drawbacks in using them to obtain many solutions. These drawbacks are strongly mitigated in the binary case as in [BJMM12] but it becomes much harder for larger values of q. We manage to partially compensate this by changing the moduli size, the place and the number of representations. For instance, for the Wave [DST18] parameters, we derive an algorithm that is a Wagner tree with seven floors where the last two floors have partial representations and the others have none.

New parameters for Wave. We then use our algorithms to study the complexity of the Wave signature scheme, for which we significantly improve the original analysis. We show that the key sizes of the original scheme presented for 128 bits of security have to be more than doubled, going roughly from 1Mb to 2.2Mb, to achieve the claimed security. This requires to study the Decode One Out of Many (DOOM) problem, on which Wave actually relies. This problem corresponds to a multiple target SD problem. More precisely, given N syndromes $(\mathbf{s}_1, \ldots, \mathbf{s}_N)$ (N can be large, for example $N = 2^{64}$) the goal is to find an error $\mathbf{e}$ of Hamming weight w and an integer i such that $\mathbf{e}\mathbf{H}^\intercal = \mathbf{s}_i$.

Applications of large weight SD in cryptography. Not surprisingly, ternary syndrome decoding is harder than its binary counterpart (with equivalent code length and dimension). This is because the input matrix has elements in $\mathbb{F}_3$ hence the input size is $\log_2(3)$ times larger. A much more surprising implication of our work is that ternary syndrome decoding is significantly harder than the binary case *for the same input size.* This new result is in sharp contrast with all the previous work on q-ary syndrome decoding that showed that the problem (when normalizing the exponent by $\log_2(q)$) becomes simpler as q increases. This is due to the fact that all the previous literature only considered the small weight case while we now take large weights into account.

We believe that this result has strong implications for code-based cryptography. Of course, in code-based cryptosystems the public key is usually not a random parity-check matrix but rather a more compact description of a pseudo-random matrix or a matrix with some structure that admits a trapdoor. Nevertheless, starting from ternary syndrome decoding with large weights that has (i) a very similar structure to binary syndrome decoding and (ii) is significantly harder for the same input size, we can only expect to make code-based cryptosystems more efficient. Since key sizes are often the weak point of code-based cryptosystems, it is an interesting direction to follow.

Table 5 represents the minimum input size for which the underlying syndrome decoding problem offers 128 bits of security (*i.e.* the associated algorithm can solve the problem in time at least 2^{128}).

Algorithm	$q = 2$	$q = 3$ and $W > 0.5$
Prange	275	44
Dumer/Wagner	295	83
BJMM/Our algorithm	374	99

Table 1. Minimum input sizes (in kbits) for a time complexity of 2^{128}.

We want to stress again that those input sizes in the ternary case take into account the fact that the matrix elements are in $\mathbb{F}_3$. So the increase in efficiency is quite significant.

1.4 Notations

We provide here some notation that will be used throughout the paper. The notation $x \stackrel{\triangle}{=} y$ means that x is defined to be equal to y. We denote by $\mathbb{F}_q = \{0, 1, \cdots, q-1\}$ the finite field of size q. Vectors will be written with bold letters (such as $\mathbf{e}$) and uppercase bold letters will be used to denote matrices (such as $\mathbf{H}$). Vectors are in row notation. Let $\mathbf{x}$ and $\mathbf{y}$ be two vectors, we will write $(\mathbf{x}, \mathbf{y})$ to denote their concatenation.

2 A General Framework for Solving the Syndrome Decoding Problem

2.1 The Syndrome Decoding Problem

The goal of this paper is to study the Syndrome Decoding problem, which is at the core of most code-based cryptosystems.

Problem 1. [Syndrome Decoding - $\mathrm{SD}(q, R, W)$]

Instance: $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ of full rank,
$\mathbf{s} \in \mathbb{F}_q^{n-k}$ (usually called the *syndrome*).

Output: $\mathbf{e} \in \mathbb{F}_q^n$ such that $|\mathbf{e}| = w$ and $\mathbf{e}\mathbf{H}^\intercal = \mathbf{s}$,

where $k \stackrel{\triangle}{=} \lceil Rn \rceil$, $w \stackrel{\triangle}{=} \lceil Wn \rceil$ and $|\mathbf{e}| \stackrel{\triangle}{=} |\{i : \mathbf{e}_i \neq 0\}|$.

The problem $\mathrm{SD}(q, R, W)$ is parametrized by the field size q, the rate $R \in [0, 1]$ and the relative weight $W \in [0, 1]$. We are always interested in the average case complexity (as a function of n) of this problem, where $\mathbf{H}$ is chosen uniformly at random and $\mathbf{s}$ is chosen uniformly from the set $\{\mathbf{e}\mathbf{H}^\intercal : |\mathbf{e}| = w\}$. This ensures the existence of a solution for each input and corresponds to the typical situation in cryptanalysis.

Remark 1. The matrix length n is not considered as a parameter of the problem since we are only interested in the asymptopic complexity, that is the coefficient $F(q, R, W)$ (which does not depend on n) such that the complexity of the Syndrome Decoding problem for a matrix of size n can be expressed as $2^{n(F(q,R,W)+o(1))}$.

State of the Art on $\mathbb{F}_2$. This problem was mostly studied in the case $q = 2$. Depending on the parameters R and W, the complexity of the problem can greatly vary. Let us fix a value R, and let W_{GV} denote the Gilbert-Varshamov bound, that is $W_{\mathrm{GV}} \stackrel{\triangle}{=} h_2^{-1}(1 - R)$ where h_2 is the binary entropy function restricted to the input space $\left[0, \frac{1}{2}\right]$. For $W \in \left[0, \frac{1}{2}\right]$, there exist three different regimes.

1. $W \approx W_{\mathrm{GV}}$. When W is close to W_{GV}, there is on average a small number of solutions. This is the regime where the problem is the hardest and where it is the most studied. To the best of our knowledge, we only know one code-based cryptosystem in this regime, namely the CFS signature scheme [CFS01].
2. $W \gg W_{\mathrm{GV}}$. In this case, there are on average exponentially many solutions and this makes the problem simpler. When W reaches $\frac{1-R}{2}$, the problem can be solved in average polynomial time using Prange's algorithm [Pra62]. There is a cryptographic motivation to consider W much larger than W_{GV}, for instance to build signatures schemes following the [GPV08] paradigm as it was done in [DST17] but one has to be careful to not make SD too simple.
3. $W \ll W_{\mathrm{GV}}$. In this regime, we have with high probability a unique solution. However, the search space, *i.e.* the set of vectors $\mathbf{e}$ st. $|\mathbf{e}| = \lceil Wn \rceil$ is much smaller than in the other regimes. The original McEliece system [McE78] or the QC-MDPC systems [MTSB12] are in this regime.

Remark 2. Solving $\mathrm{SD}(2, R, W)$ for $W \in \left[\frac{1}{2}, 1\right]$ can be reduced directly to one of the above-mentioned cases using $\mathrm{SD}(2, R, 1 - W)$.

Remark 3. Contrary to the binary case, when $q \geq 3$ the case of large relative weight can not be reduced to that of small relative weight using tricks as in Remark 2. In fact, the problem has a quite different behaviour in small and large weights as we will see below.

2.2 The PGE+SS Framework in $\mathbb{F}_q$

The SD problem has been extensively studied in the binary case. Most algorithms designed to solve this problem [Dum91, MMT11, BJMM12] follow the same framework:

1. perform a partial Gaussian elimination (PGE) ;
2. solve the Subset Sum problem (SS) on a reduced instance.

We will see how we can extend this framework to the non-binary case. Our goal here is to describe the PGE+SS framework for solving $\mathrm{SD}(q, R, W)$. Fix $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ of full rank and $\mathbf{s} \in \mathbb{F}_q^{n-k}$. Recall that we want to find $\mathbf{e} \in \mathbb{F}_q^n$ such that $|\mathbf{e}| = w \stackrel{\triangle}{=} \lceil Wn \rceil$ and $\mathbf{He}^\intercal = \mathbf{s}^\intercal$. Let us introduce ℓ and p, two parameters of the system, that we will consider fixed for now. In this framework, an algorithm for solving $\mathrm{SD}(q, R, W)$ will consist of 4 steps: a permutation step, a partial Gaussian elimination step, a Subset Sum step and a test step.

1. *Permutation step.* Pick a random permutation π. Let $\mathbf{H}_\pi$ be the matrix $\mathbf{H}$ where the columns have been permuted according to π. We now want to solve the problem $\mathrm{SD}(q, R, W)$ on inputs $\mathbf{H}_\pi$ and $\mathbf{s}$.
2. *Partial Gaussian elimination step.* If the top left square submatrix of $\mathbf{H}_\pi$ of size $n-k-\ell$ is not of full rank, go back to step 1 and choose another random permutation π. This happens with constant probability. Else, if this submatrix is of full rank, perform a Gaussian elimination on the rows of $\mathbf{H}_\pi$ using the first $n-k-\ell$ columns. Let $\mathbf{S} \in \mathbb{F}_q^{(n-k)\times(n-k)}$ be the invertible matrix corresponding to this operation. We now have two matrices $\mathbf{H}' \in \mathbb{F}_q^{(n-k-\ell)\times(k+\ell)}$ and $\mathbf{H}'' \in \mathbb{F}_q^{\ell\times(k+\ell)}$ such that:
$$\mathbf{SH}_\pi = \begin{pmatrix} \mathbf{1}_{n-k-\ell} & \mathbf{H}' \\ \mathbf{0} & \mathbf{H}'' \end{pmatrix}.$$
The error $\mathbf{e}$ can be written as $\mathbf{e} = (\mathbf{e}', \mathbf{e}'')$ where $\mathbf{e}' \in \mathbb{F}_q^{n-k-\ell}$ and $\mathbf{e}'' \in \mathbb{F}_q^{k+\ell}$, and one can write $\mathbf{sS}^\intercal = (\mathbf{s}', \mathbf{s}'')$ with $\mathbf{s}' \in \mathbb{F}_q^{n-k-\ell}$ and $\mathbf{s}'' \in \mathbb{F}_q^{\ell}$.
$$\begin{aligned} \mathbf{H}_\pi\mathbf{e}^\intercal = \mathbf{s}^\intercal &\iff \mathbf{SH}_\pi\mathbf{e}^\intercal = \mathbf{Ss}^\intercal \\ &\iff \begin{pmatrix} \mathbf{1}_{n-k-\ell} & \mathbf{H}' \\ \mathbf{0} & \mathbf{H}'' \end{pmatrix} \begin{pmatrix} \mathbf{e}'^\intercal \\ \mathbf{e}''^\intercal \end{pmatrix} = \begin{pmatrix} \mathbf{s}'^\intercal \\ \mathbf{s}''^\intercal \end{pmatrix} \\ &\iff \begin{cases} \mathbf{e}'^\intercal + \mathbf{H}'\mathbf{e}''^\intercal = \mathbf{s}'^\intercal \\ \mathbf{H}''\mathbf{e}''^\intercal = \mathbf{s}''^\intercal \end{cases} \end{aligned} \tag{1}$$
To solve the problem, we will try to find a solution $(\mathbf{e}', \mathbf{e}'')$ to the above system such that $|\mathbf{e}''| = p$ and $|\mathbf{e}'| = w - p$.
3. *The Subset Sum step.* Compute a set $\mathcal{S} \subseteq \mathbb{F}_q^{k+\ell}$ of solutions $\mathbf{e}''$ of $\mathbf{H}''\mathbf{e}''^\intercal = \mathbf{s}''^\intercal$ such that $|\mathbf{e}''| = p$. We will solve this problem by considering it as a Subset Sum problem as it is described in Subsection 2.4.
4. *The test step.* Take a vector $\mathbf{e}'' \in \mathcal{S}$ and let $\mathbf{e}'^\intercal = \mathbf{s}'^\intercal - \mathbf{H}'\mathbf{e}''^\intercal$. Equation (1) ensures that $\mathbf{H}_\pi(\mathbf{e}', \mathbf{e}'')^\intercal = \mathbf{s}^\intercal$. If $|\mathbf{e}'| = w - p$, $\mathbf{e} = (\mathbf{e}', \mathbf{e}'')$ is a solution of $\mathrm{SD}(q, R, W)$ on inputs $\mathbf{H}_\pi$ and $\mathbf{s}$, which can be turned into a solution of the initial problem by permuting the indices, as detailed in Equation (2). Else, try again for other values of $\mathbf{e}'' \in \mathcal{S}$. If no element of $\mathcal{S}$ gives a valid solution, go back to step 1.

At the end of protocol, we have a vector $\mathbf{e}$ such that $\mathbf{H}_\pi\mathbf{e}^\intercal = \mathbf{s}^\intercal$ and $|\mathbf{e}| = w$. Let $\mathbf{e}_{\pi^{-1}}$ be the vector $\mathbf{e}$ where we permute all the coordinates according to π^{-1}. Hence,
$$\mathbf{He}_{\pi^{-1}}^\intercal = \mathbf{H}_\pi\mathbf{e}^\intercal = \mathbf{s}^\intercal \quad \text{and} \quad |\mathbf{e}_{\pi^{-1}}| = |\mathbf{e}| = w. \tag{2}$$
Therefore, $\mathbf{e}_{\pi^{-1}}$ is a solution to the problem.

2.3 Analysis of the Algorithm

In order to analyse this algorithm, we rely on the following two propositions.

Notation 1 *An important quantity to understand the complexity of this algorithm is the probability of success at step 4. On an input* $(\mathbf{H}, \mathbf{s})$ *uniformly drawn at random, suppose that we have a solution to the Subset Sum problem, i.e. a vector* $\mathbf{e}''$ *such that* $\mathbf{H}''\mathbf{e}''^{\intercal} = \mathbf{s}''^{\intercal}$ *and* $|\mathbf{e}''| = p$*. Let* $\mathbf{e}'^{\intercal} = \mathbf{s}'^{\intercal} - \mathbf{H}'\mathbf{e}''^{\intercal}$*. We will denote:*

$$\mathcal{P}_{p,\ell} \stackrel{\triangle}{=} \mathbb{P}\left(|\mathbf{e}'| = w - p \mid |\mathbf{e}''| = p\right).$$

Proposition 1. *We have, up to a polynomial factor,*

$$\mathcal{P}_{p,\ell} = \frac{\binom{n-k-\ell}{w-p}(q-1)^{w-p}}{\min\left(q^{n-k-\ell}, \binom{n}{w}(q-1)^{w}q^{-\ell}\right)}.$$

Proof. The proof of this statement is simple combinatorics. The numerator corresponds to the number of vectors $\mathbf{e}'$ of weight $w-p$. The denominator corresponds to the inverse of the probability that $\mathbf{e}'^{\intercal} = \mathbf{s}'^{\intercal} - \mathbf{H}'\mathbf{e}''^{\intercal}$. For a typical random behavior, this is equal to $q^{n-k-\ell}$. But here we know that there is at least one solution. Therefore, we know that the number of vectors of weight $w-p$ is bounded from above by the number of vectors $\mathbf{e}$ such that $\mathbf{H}''\mathbf{e}''^{\intercal} = \mathbf{s}''^{\intercal}$. This explains the second term of the minimum. □

Proposition 2. *Assume that we have an algorithm that finds a set* $\mathcal{S}$ *of solutions of the Subset Sum problem in time* T*. The average running time of the algorithm is, up to a polynomial factor,*

$$T \cdot \max\left(1, \frac{1}{|\mathcal{S}| \cdot \mathcal{P}_{p,\ell}}\right).$$

As we can see, all the parameters are entwined. The success probability $\mathcal{P}_{p,\ell}$ depends of p and ℓ, as well as the time T to find the set $\mathcal{S}$ of solutions.

In this work, we will focus on a family of parameters useful in the analysis of the Wave signature scheme [DST18]. More precisely, we will study the following regime:

$$q = 3 \quad ; \quad R \in [0.5, 0.9] \quad ; \quad W \in [0.9, 0.99].$$

One consequence of working with a very high relative weight W is that our best algorithms will work with:

$$\ell = \Omega(n) \quad ; \quad p = k + \ell. \tag{3}$$

Here, ℓ is $\Omega(n)$ for the following reason: if $\ell = o(n)$ then it is readily verified that, asymptotically in n, the average running time of the PGE+SS framework will be bounded from below (up to a polynomial factor) by $1/\mathcal{P}_{p,0}$. This exactly corresponds to the complexity of the simplest generic algorithm to solve SD, namely Prange's ISD algorithm [Pra62].

2.4 Reduction to the Subset Sum Problem

In step 3 of the PGE+SS framework, we have a matrix $\mathbf{H}'' \in \mathbb{F}_q^{\ell \times (k+\ell)}$, a vector $\mathbf{s}'' \in \mathbb{F}_q^{\ell}$ and we want to compute a set $\mathcal{S} \subseteq \mathbb{F}_q^{k+\ell}$ of solutions $\mathbf{e}''$ of $\mathbf{H}''\mathbf{e}''^{\intercal} = \mathbf{s}''^{\intercal}$ such that $|\mathbf{e}''| = p$. At first sight, this looks exactly like a Syndrome Decoding problem with inputs $\mathbf{H}''$ and $\mathbf{s}''$ so we could just recursively apply the best SD algorithm on this subinstance. But the main difference is that, in this case, we want to find many solutions to the problem and not just one. One possibility to solve this problem is to reduce it to the Subset Sum problem on vectors in $\mathbb{F}_q^{\ell}$.

Problem 2. [Subset Sum problem - SS(q, n, m, L, p)]

Instance: n vectors $\mathbf{x}_i \in \mathbb{F}_q^m$ for $1 \le i \le n$, a target vector $\mathbf{s} \in \mathbb{F}_q^m$.

Output: L solutions $\mathbf{b}^{(j)} = (b_1^{(j)}, \dots, b_n^{(j)}) \in \{0,1\}^n$ for $1 \le j \le L$, such that for all j, $\sum_{i=1}^{n} b_i^{(j)}\mathbf{x}_i = \mathbf{s}$ and $|\mathbf{b}^{(j)}| = p$.

We can consider the same problem with elements b in $\mathbb{F}_q$ instead of $\{0,1\}$.

Problem 3. [Subset Sum with non-zero characteristic - SSNZC(q,n,m,L,p)]
Instance: n vectors $\mathbf{x}_i \in \mathbb{F}_q^m$ for $1 \le i \le n$, a target vector $\mathbf{s} \in \mathbb{F}_q^m$.
Output: L solutions $\mathbf{b}^{(j)} = (b_1^{(j)}, \dots, b_n^{(j)}) \in \mathbb{F}_q^n$ for $1 \le j \le L$,
such that for all j, $\sum_{i=1}^n b_i^{(j)} \mathbf{x}_i = \mathbf{s}$ and $|\mathbf{b}^{(j)}| = p$.

Notation 2 *We will denote* SS$(q,n,m,L,\emptyset)$ *(resp.* SSNZC*) the* SS *problem (resp.* SSNZC *problem) without any constraint on the weight.*

Again, we will be interested in the average case, where all the inputs are taken uniformly at random. Notice that the problem that needs to be solved at step 3 of the PGE+SS framework reduces exactly to SSNZC$(q, k+\ell, \ell, |\mathcal{S}|, p)$.

There is an extensive literature [HJ10, BCJ11] about the Subset Sum problem for specific parameter ranges, typically when $L = 1, q = 2, n = m$ and $p = \frac{m}{2}$. This is the hardest case where there is on average a single solution. There are several regimes of parameters, each of which lead to different algorithms. For instance, when $m = O(n^\varepsilon)$ for $\varepsilon < 1$, there are many solutions on average and we are in the high density setting for which we have sub-exponential algorithm [Lyu05]. Table 2 summarizes the complexity of algorithms to solve the Subset Sum problem for some different regimes of parameters when only one solution is required ($L = 1$) and for $q = 2$.

Value of m	Complexity	Reference
$O(\log(n))$	poly(n)	[GM91, CFG89]
$O(\log(n)^2)$	poly(n)	[FP05]
$O(n^\varepsilon)$ for $\varepsilon < 1$	$2^{O\left(\frac{n^\varepsilon}{\log(n)}\right)}$	[Lyu05]
n	$2^{O(n)}$	[HJ10, BCJ11]

Table 2. Complexity of best known algorithms to solve SS$(2,n,m,1,\emptyset)$.

In our case, m will be a small, but constant, fraction of n, which leads to multiple solutions but exponentially complex algorithms to find them. We will be in a moderate density situation. Furthermore, the case $L = 1$ and $L \gg 1$ require quite different algorithms. When $q = 2$, authors of [BJMM12] show how to optimize this whole approach to solve the original Syndrome Decoding problem using better algorithms for the Subset Sum problem.

2.5 Application to the PGE+SS Framework with High Weight

There are quite a lot of interesting regimes that could be studied with this approach and have not been studied yet. Indeed, very few papers tackle the case $q \ge 3$ and they only cover a small fraction of the possible parameters. In this work we focus on the problem SSNZC$(3, k+\ell, \ell, |\mathcal{S}|, k+\ell)$ given by the PGE+SS framework for high weights in $\mathbb{F}_3$. The choice of $p = k+\ell$ for large weights is explained in Equation (3). This is quite convenient because this problem is actually equivalent to solving SS$(3, k+\ell, \ell, |\mathcal{S}|, \emptyset)$ as shown by the following lemma.

Lemma 1. *If we have an algorithm that solves* SS$(3, k+\ell, \ell, |\mathcal{S}|, \emptyset)$ *then we have an algorithm that solves* SSNZC$(3, k+\ell, \ell, |\mathcal{S}|, k+\ell)$ *with the same complexity.*

Proof. Let $\mathcal{A}$ be an algorithm that solves SS$(3, k+\ell, \ell, |\mathcal{S}|, \emptyset)$ and consider an instance $(\mathbf{x}_1, \dots, \mathbf{x}_{k+\ell})$, $\mathbf{s}$ of SSNZC$(3, k+\ell, \ell, |\mathcal{S}|, k+\ell)$. We want to find $b_1, \dots, b_{k+\ell} \in \{1,2\}$ (see $\mathbb{F}_3 = \{0,1,2\}$) such that $\sum_{i=1}^{k+\ell} b_i \mathbf{x}_i = s$. Let $\mathbf{s}' = 2\mathbf{s} + \sum_i \mathbf{x}_i$ and let us run $\mathcal{A}$ on input $(\mathbf{x}_1, \dots, \mathbf{x}_{k+\ell}), \mathbf{s}'$. We obtain $b'_1, \dots, b'_{k+\ell} \in \{0,1\}$ such that $\sum_{i=1}^{k+\ell} b'_i \mathbf{x}_i = \mathbf{s}'$. Take $b_i = \frac{b'_i - 1}{2}$ for $1 \le i \le k+\ell$, where the division is done in $\mathbb{F}_3$ and return $(b_1, \dots, b_{k+\ell})$.

Indeed, this gives a valid solution to the problem: the elements b_i belong to $\{1,2\}$ and we have:

$$\sum_{i=1}^{k+\ell} b_i \mathbf{x}_i = \sum_{i=1}^{k+\ell} \frac{b_i' - 1}{2} \mathbf{x}_i = \frac{\mathbf{s}'}{2} - \frac{\sum_{i=1}^{k+\ell} \mathbf{x}_i}{2} = \mathbf{s}. \qquad \square$$

Hence, in the context of the PGE+SS framework for solving SD with high weights, it is enough to solve $\mathrm{SS}(3, k+\ell, \ell, |\mathcal{S}|, \emptyset)$. However, as explained at the end of Subsection 2.2, we will have to choose $\ell = \Omega(n) = \Omega(k)$ (because $k = \lceil Rn \rceil$). Therefore, we are in a regime where solving the Subset Sum problem is exponential, as explained in the previous subsection. However, as we will see in the next session, we will be able to choose ℓ as a small fraction of k. In this case, generic algorithms as Wagner's [Wag02] perform exponentially better compared to the case $\ell = k$.

3 Ternary Subset Sum with the Generlized Birthday Algorithm

We show in this section how to solve the SS $(3, k+\ell, \ell, L, \emptyset)$ problem, first with Wagner's algorithm [Wag02] . Parameters k and ℓ will be free. We will focus on the values L for which we can find L solutions to SS $(3, k+\ell, \ell, L, \emptyset)$ in time $O(L)$. In such a case, we say that we can find solutions in *amortized time* $O(1)$.

3.1 A Brief Description of Wagner's Algorithm

Recall that we are here in the context of the Subset Sum step of the PGE+SS framework described in Subsection 2.2. Given $k+\ell$ vectors $\mathbf{x}_1, \cdots, \mathbf{x}_{k+\ell} \in \mathbb{F}_3^\ell$ (columns of the matrix $\mathbf{H}''$) and a target vector $\mathbf{s} \in \mathbb{F}_3^\ell$, our goal is to find L solutions of the form $\mathbf{b}^{(j)} = (b_1^{(j)}, \cdots, b_{k+\ell}^{(j)}) \in \{0,1\}^{k+\ell}$ such that for all $1 \le j \le L$,

$$\sum_{i=1}^{k+\ell} b_i^{(j)} \mathbf{x}_i = \mathbf{s}. \tag{4}$$

Here, we are interested in the average case which means that all the vectors $\mathbf{x}_i$'s are independent and follow a uniform law over $\mathbb{F}_3^\ell$. In order to apply Wagner's algorithm [Wag02], let $a \in \mathbb{N}^*$ be some integer parameter, and denote for $i \in [\![1, 2^a]\!]$ by $\mathcal{I}_i$ the sets $\mathcal{I}_i \stackrel{\triangle}{=} [\![1 + \frac{(i-1)(k+l)}{2^a}, \frac{i(k+l)}{2^a}]\!]$. The sets $\mathcal{I}_i$'s form a partition of $[\![1, k+\ell]\!]$.

The first step of Wagner's algorithm is to compute 2^a lists $(\mathcal{L}_i)_{1 \le i \le 2^a}$ of size L such that:

$$\forall i \in [\![1, 2^a]\!],\ \mathcal{L}_i \subseteq \left\{ \sum_{j \in \mathcal{I}_i} b_j \mathbf{x}_j \ :\ \forall j \in \mathcal{I}_i,\ b_j \in \{0,1\} \right\} \text{ and } |\mathcal{L}_i| = L. \tag{5}$$

Each list $\mathcal{L}_i$ consists of L random elements of the form $\sum_{j \in \mathcal{I}_i} b_j \mathbf{x}_j$ where the randomness is on $b_j \in \{0,1\}$. By construction, we make sure that given $\mathbf{y} \in \mathcal{L}_i$ we have access to the coefficients $(b_j)_{j \in \mathcal{I}_i}$ such that $\mathbf{y} = \sum_{j \in \mathcal{I}_i} b_j \mathbf{x}_j$. In other words, we have divided the vectors $\mathbf{x}_1, \dots, \mathbf{x}_{k+\ell}$ in 2^a stacks of $(k+\ell)/2^a$ vectors and for each stack we have computed a list of L random linear combinations of the vectors in the stack. The running time to build theses lists is $O(L)$. Once we have computed these lists we can use the main idea of Wagner to solve (4). In our case we would like to find solutions in amortized time $O(1)$. For this, Wagner's algorithm requires lists $\mathcal{L}_i$'s to be all of the same size:

$$\forall i \in [\![1, 2^a]\!],\ |\mathcal{L}_i| = L = 3^{\ell/a}.$$

This gives a first constraint on the parameters k, ℓ and a, namely:

$$3^{\ell/a} \le 2^{(k+\ell)/2^a} \quad \text{(number of vectors } \mathbf{b}^{(j)} \text{ in each stack)}.$$

which puts a constraint on a since k, ℓ are fixed. With these lists at hand, Wagner's idea is to merge the lists in the following way. For every $p \in \{1, 3, \cdots, 2^a - 3\}$, create a list $\mathcal{L}_{p,p+1}$ from $\mathcal{L}_p$ and $\mathcal{L}_{p+1}$ such that:

$$\mathcal{L}_{p,p+1} \stackrel{\triangle}{=} \{\mathbf{y}_p + \mathbf{y}_{p+1} \ : \ \mathbf{y}_i \in \mathcal{L}_i \text{ and the last } \ell/a \text{ bits of } \mathbf{y}_p + \mathbf{y}_{p+1} \text{ are 0's.-}\}\,.$$

A list $\mathcal{L}_{2^a-1,2^a}$ is created from $\mathcal{L}_{2^a-1}$ and $\mathcal{L}_{2^a}$ in the same way except that the last ℓ/a bits have to be equal to those of $\mathbf{s}$. As the elements of the lists $\mathcal{L}_p$ are drawn uniformly at random in $\mathbb{F}_3^\ell$, it is easily verified that by merging them on ℓ/a bits, the new lists $\mathcal{L}_{p,p+1}$ are typically of size $|\mathcal{L}_i^2|/3^{\ell/a} = (3^{\ell/a})^2/3^{\ell/a} = 3^{\ell/a}$. Therefore, the cost in time and in space of such a merging (by using classical techniques such as hash tables or sorted lists) will be $O(3^{\ell/a})$ on average. This way, we obtain 2^{a-1} lists of size L. It is readily seen that we can repeat this process $a-1$ times, with each time a cost of $O(3^{\ell/a})$ for merging on ℓ/a bits. After a steps, we obtain a list of solutions to the Equation (4) containing $L = 3^{\ell/a}$ elements on average.

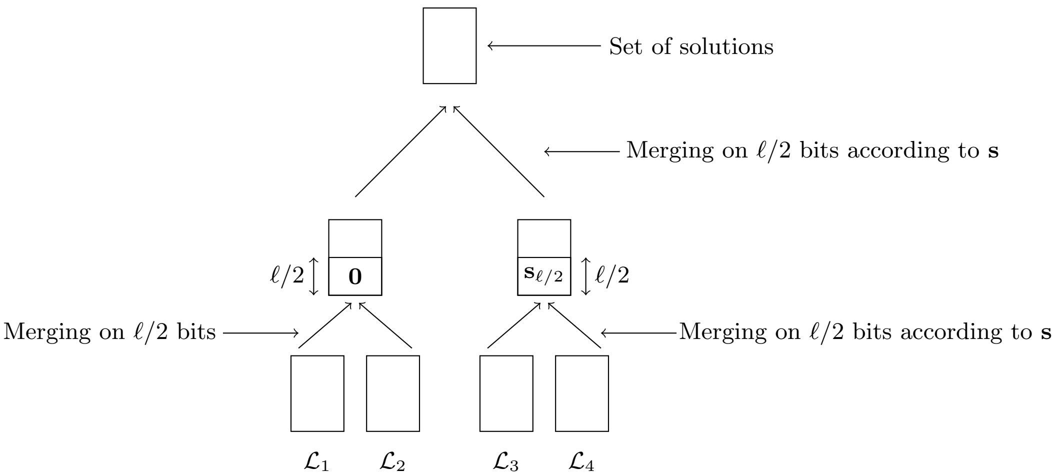

Fig. 2. Wagner's algorithm with $a = 2$

Let us summarize the previous discussion with the following theorem.

Theorem 1. *Fix $k, \ell \in \mathbb{N}^*$ and let a be any non zero integer such that*

$$3^{\ell/a} \leq 2^{(k+\ell)/2^a}.$$

The associated $\mathrm{SS}(3, k+\ell, \ell, 3^{\ell/a}, \emptyset)$ *problem can be solved in average time and space* $O(3^{\ell/a})$.

This theorem indicates for which value L it is possible to find L solutions in time $O(L)$ using Wagner's approach.

3.2 Smoothing of Wagner's Algorithm

Wagner's algorithm as stated above shows how to find L solutions in amortized time $O(1)$ for $L = 3^{\ell/a}$. If we want more than L solutions, we can repeat this algorithm and find all those solutions also in amortized time $O(1)$. So the smaller L is, the better the algorithm performs. So the idea is to take the largest integer a such that $3^{\ell/a} < 2^{(k+\ell)/2^a}$ and take $L = 3^{\ell/a}$, as explained in Theorem 1. But this induces a discontinuity in the optimal value of L and on the complexity: when the optimal value of a changes, the slope of the complexity curve changes, as we can see on Figures 6 and 7. We show here a refinement of Theorem 1 that reduces the discontinuity.

Proposition 3. *Let a be the largest integer such that $3^{\ell/(a-1)} < 2^{(k+\ell)/2^{a-1}}$. If $a \geq 3$, the above algorithm can find 2^λ solutions in time $O(2^\lambda)$ with*

$$\lambda = \frac{\ell \log(3)}{a-2} - \frac{k+\ell}{(a-2)2^{a-1}}.$$

We see that we retrieve the result of Theorem 1 when $3^{\ell/a} = 2^{(k+\ell)/2^a}$. We have not found any statement of this form in the literature, which is surprising because Wagner's algorithm has a variety of applications. We now prove the proposition.

Proof. Parameters k and ℓ are fixed. Let a be the largest integer such that $3^{\ell/(a-1)} < 2^{(k+\ell)/2^{a-1}}$ and we suppose that $a \geq 3$. We will consider Wagner's algorithm on a levels but the merging at the bottom of the tree will be performed with a lighter constraint: we want the sums to agree on less than ℓ/a bits. Indeed, we consider the following list sizes. At the bottom of the trees, we take lists of size $2^{\frac{k+\ell}{2^a}}$ (the maximal possible size); at all other levels, we want lists of size 2^λ. We run Wagner's algorithm by firstly merging on m bits. In order to obtain lists of size 2^λ at the second step, we have to choose m such that

$$\frac{\left(2^{(k+\ell)/2^a}\right)^2}{3^m} = 2^\lambda \qquad i.e. \qquad \frac{2(k+\ell)}{2^a} - m\log_2(3) = \lambda. \tag{6}$$

The other $(a-1)$ merging steps are designed such that merging two lists of size 2^λ gives a new list of size 2^λ, which means that we merge on $\lambda/\log_2(3)$ bits. However, in the final list we want to obtain solutions to the problem, which means that in total we have to have put a constraint on all bits. Therefore, λ and m have to verify:

$$m + (a-1)\frac{\lambda}{\log_2(3)} = \ell. \tag{7}$$

By combining Equations (6) and (7) we get:

$$\lambda = \frac{\ell \log_2(3)}{a-2} - \frac{k+\ell}{(a-2)2^{a-1}}.$$

It is easy to check that under the conditions $3^{\ell/(a-1)} < 2^{(k+\ell)/2^{a-1}}$ and $a \geq 3$, λ and m are positive which concludes the proof. □

4 Ternary Subset Sum Using Representations

4.1 Basic idea

In a Wagner's algorithm tree (see Figure 2), we split each list in two according to the *left-right* procedure. This means that if we start from a set $S = \{\sum_{j\in[\![A,B]\!]} b_j \mathbf{x}_j : |b_j| = p\}$, we decompose each element of $\mathbf{y} \in S$ as $\mathbf{y} = \mathbf{y}_1 + \mathbf{y}_2$ where $\mathbf{y}_1 \in S_1$ and $\mathbf{y}_2 \in S_2$, where

$$S_1 \triangleq \left\{ \sum_{j \in [\![A, \lfloor \frac{B+A}{2} \rfloor]\!]} b_j \mathbf{x}_j : b_j \in \{0,1\},\ |\mathbf{b}| = p/2 \right\}$$

$$S_2 \triangleq \left\{ \sum_{j \in [\![\lfloor \frac{B+A}{2} \rfloor + 1, B]\!]} b_j \mathbf{x}_j : b_j \in \{0,1\},\ |\mathbf{b}| = p/2 \right\}.$$

Such a decomposition does not always exist, but it exists with probability at least $\frac{1}{p}$. Indeed, the probability that a vector of weight p can be splitted this way is

$$\frac{\binom{n/2}{p/2}^2}{\binom{n}{p}} \geq p.$$

Wagner's algorithm uses this principle. When looking for vectors $\mathbf{b}$ containing the same number of 0's and 1's, it looks for $\mathbf{b}$ in the form $\mathbf{b} = \mathbf{b}_1 + \mathbf{b}_2$, where the second half of $\mathbf{b}_1$ and the first half of $\mathbf{b}_2$ are only zeros. The first half of $\mathbf{b}_1$ and the second half of $\mathbf{b}_2$ are expected to have the same number of 0's and 1's.

The idea of representations is to follow the Wagner's approach of list merging but with allowing more possibilities to write $\mathbf{b}$ as the sum of two vectors $\mathbf{b} = \mathbf{b}_1 + \mathbf{b}_2$. We remove the constraint that $\mathbf{b}_1$ has zeroes on its right half and $\mathbf{b}_2$ has zeroes on its left half. We replace it by a less restrictive constraint: we fix the number of 0's, 1's and 2's (see $\mathbb{F}_3 = \{0, 1, 2\}$) in $\mathbf{b}_1$ and $\mathbf{b}_2$.

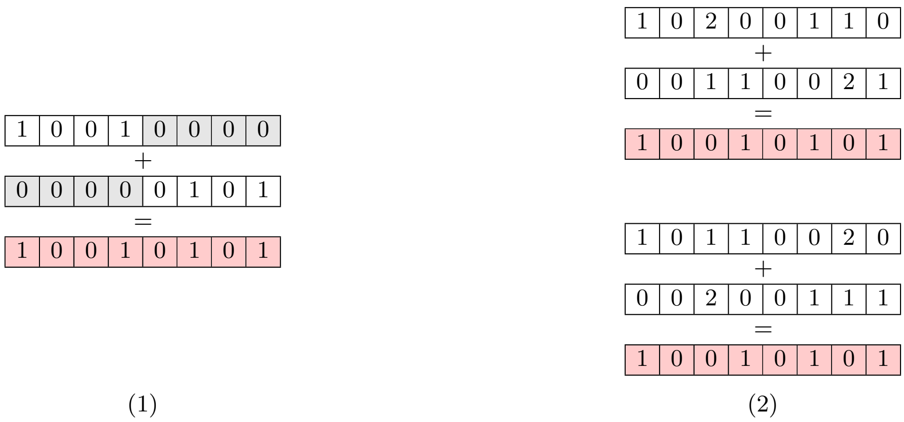

Fig. 3. Same vector (1) using left-right split and (2) using representations.

More precisely, we consider the set

$$S' = \left\{ \sum_{j \in [\![A,B]\!]} b_j \mathbf{x}_j : b_j \in \mathbb{F}_3,\ |\{b_j = 1\}| = p' \text{ and } |\{b_j = 2\}| = p'' \right\}$$

for some weights p' and p'' and we want to decompose each $\mathbf{y}$ into $\mathbf{y}_1 + \mathbf{y}_2$ such that $\mathbf{y}_1, \mathbf{y}_2 \in S'$. On the example of Figure 3, we have $p = 4$, $p' = 3$ and $p'' = 1$. At first sight, this approach may seem unusual. Indeed, except for very specific values of p' and p'', the sum $\mathbf{y}_1 + \mathbf{y}_2$ will rarely match the desired weight p to be in S. However, the positive aspect is that each element $\mathbf{y} \in S$ accepts many decompositions (the so-called "representations") $\mathbf{y}_1 + \mathbf{y}_2$ where $\mathbf{y}_1, \mathbf{y}_2 \in S'$. The results from [HJ10, BCJ11, BJMM12] show that this large number of ways to represent each element can compensate the fact that most decompositions do not belong to S. One can slightly lower number of bits when merging lists to obtain on average the expected number of elements in the merged list.

On thing you might notice in what we presented above is that in the set S', the elements b_j belong to the set $\mathbb{F}_3$ and not $\{0, 1\}$, even though we want to obtain a binary solution. The ternary structure also increases the number of representations as shown in Figure 3. It is actually natural to consider representations of binary strings using 3 elements $\{0, 1, 2\}$, as in [BCJ11].

4.2 Partial Representations

If we relieve too many constraints and allow too many representations of a solution, it may happend that we end up with multiple copies of the same solution. In order to avoid this situation, we use partial representations, which is an intermediate approach between *left-right* splitting and using *representations*, as illustrated in Figure 4.

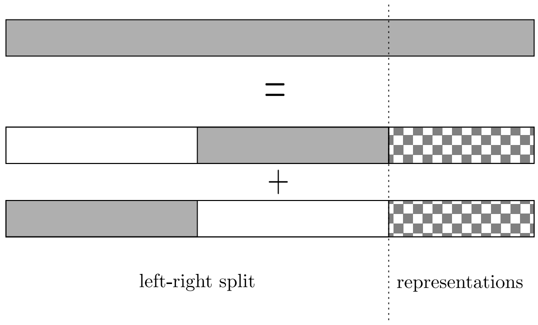

Fig. 4. Decomposing a vector using partial representations

4.3 Presentation of our Algorithm

Plugging representations in Wagner's algorithm can be done in a variety of ways. The way we achieved our best algorithm was mostly done by trial and error. We present here the main features of our algorithm:

- In the regime we will consider, the number of floors a varies from 5 to 7. Notice that this is quite larger than in other similar algorithms and is mostly due to the fact that we have many solutions to our Subset Sum problem.
- With this amount of steps, and because we want to find many solutions, representations become less efficient. Indeed, the fact that we obtain many badly formed elements makes it harder to find solutions in amortized time $O(1)$ (or even just in small time).
- However, we show that representations can still be useful. For most parameter range, the optimal algorithm will a left-right split at the bottom level of the tree, then 2 layers of partial representation and from there to the top level, we use left-right splits again.

Figure 5 represent the tree of our algorithm. Each list merging has an associated number of bits on which we merge. On the bottom floors, we filter as much as the number of representations whereas on top, we filter as much as the list size. This means that the bottom constraints filter the representations and the top constraints filter some amount of solutions. We found that this procedure gave the best results.

Figure 5 illustrates an example for $a = 4$. When we increase the number of floors, we just add some left-right splits. On this figure, the green and red lists stand for a left-right split and the brown lists use representations.

In the next section, we present the different parameters for a particular input to show how our algorithm behaves.

4.4 Application to the Syndrome Decoding problem

We embedded the algorithms we described to solve the subset-sum part, namely the classical Wagner algorithm and the smoothed one in §3 and finally the one using representation technique in §4. in the PGE+SS framework. By using Proposition 1 we derived the exponents given in Figures 6 and 7.

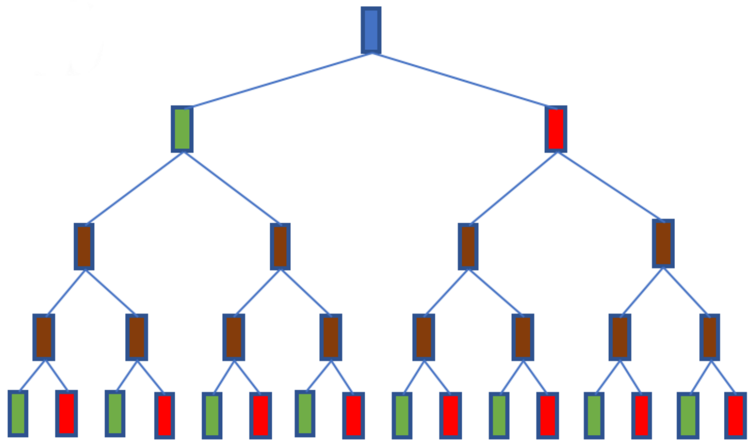

Fig. 5. General structure of our algorithm for $a = 4$

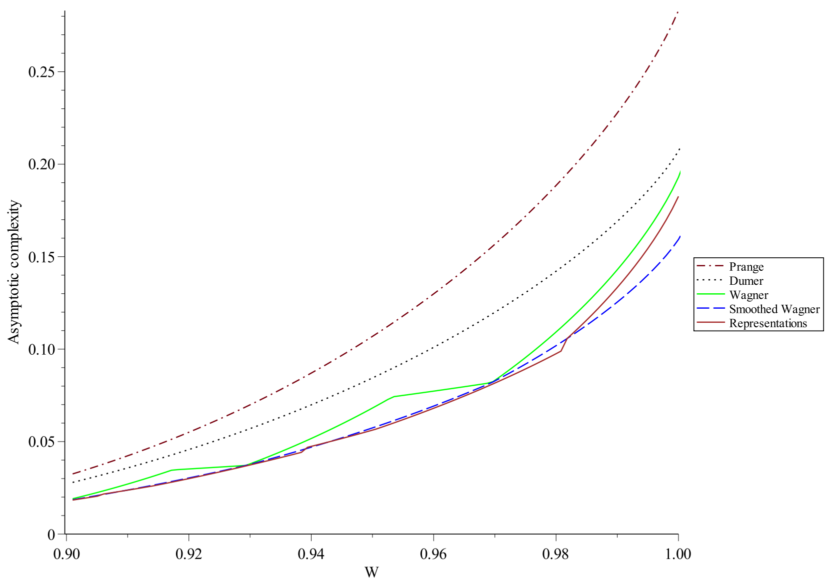

Fig. 6. Comparison of the exponent complexities for $R = 0.5$

We present here the details of our algorithm for the $\mathrm{SD}(3, R, W)$ with $R = 0.676$ and $W = 0.948366$. These are the parameters which are used for the analysis of Wave. In the PGE+SS framework (see Section 2.2), we needed to choose to parameters p and ℓ.

– We choose $p = k + \ell$ and $\ell = 0.060506$.

We use representations to solve this problem. In this parameter range, there are many solutions. The best algorithm we found has the following properties:

– $a = 6$: the associated Wagner tree has 6 floors, i.e. $2^6 = 64$ leaves. The top 3 floors consist of left-right splits. This means at level 3, we have 8 lists

$$\forall i \in [\![1, 8]\!],\ \mathcal{L}_i \subseteq \left\{ \sum_{j \in \mathcal{I}_i} b_j \mathbf{x}_j \ :\ \forall j \in \mathcal{I}_i,\ b_j \in \{0, 1\} \right\} \text{ and } |\mathcal{L}_i| = L.$$

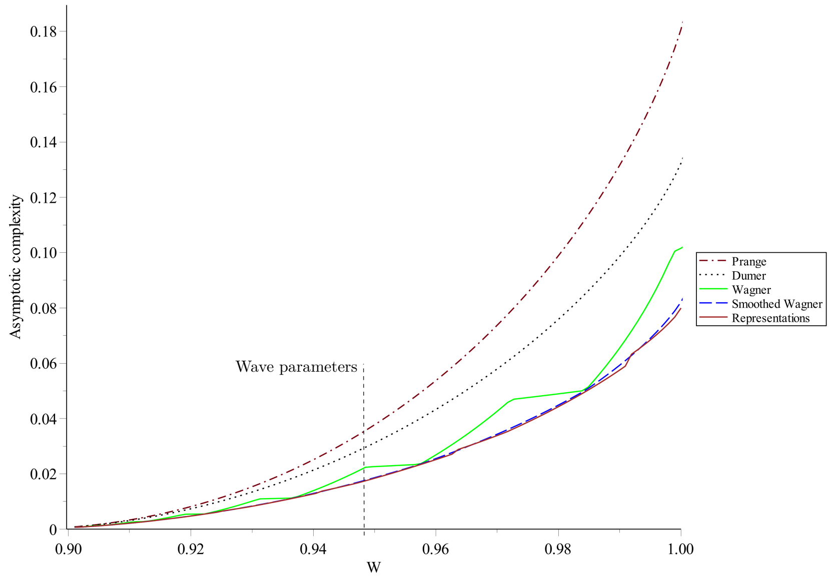

Fig. 7. Comparison of the exponent complexities for $R = 0.676$

with $\mathcal{I}_i \stackrel{\triangle}{=} [\![1 + \frac{(i-1)(k+l)}{8}, \frac{i(k+l)}{8}]\!]$. The size L will be the overall complexity of the underlying subset sum problem. In out algorithm, we will have $L = 2^{0.0177n}$.
- We then perform 2 floors of representations. We maximize over all possible partial representations. The bottom 64 lists will also have size L.

What we actually saw during our simulations, as shown in Figures 6 and 7 is that representations give a very small advantage, unlike the small weight case.

5 New Parameters for the WAVE Signature Scheme

Wave is a new code-based signature scheme proposed in [DST18]. It uses a *hash-and-sign* approach and follows the GPV paradigm [GPV08] with the instantiation of a code-based preimage sampleable family of functions.

Forging a signature in the Wave scheme amounts to solve the SD problem. However, instead of trying to forge a signature for one message of our choice, a natural idea is to try to forge one message among a set of selected set of messages. This context leads directly to a slight variation of the classical SD problem. Instead of having one syndrome, there is a list of possible syndromes and the goal is to decode one of them. This problem is known as the *Decoding One Out of Many* (DOOM) problem.

Problem 4. [Decoding One Out of Many - DOOM(n, z, q, R, W)]

Instance: $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ of full rank,
$\mathbf{s}_1, \cdots, \mathbf{s}_z \in \mathbb{F}_q^{n-k}$.

Output: $\mathbf{e} \in \mathbb{F}_q^n$ and $i \in [\![1, z]\!]$ such that $|\mathbf{e}| = w$ and $\mathbf{e}\mathbf{H}^\intercal = \mathbf{s}_i$,

where $k \stackrel{\triangle}{=} \lceil Rn \rceil$, $w \stackrel{\triangle}{=} \lceil Wn \rceil$ and $|\mathbf{e}| \stackrel{\triangle}{=} |\{i : \mathbf{e}_i \neq 0\}|$.

This problem was first considered in [JJ02] and later analyzed for the binary case ($q = 2$) in [Sen11, DST17]. This papers show that one can solve the DOOM problem with an exponential speed-up compared to the SD problem with equivalent parameters.

The difference induced by DOOM on the PGE+SS framework is that it increases the search space. Namely, instead of searching a solution $\mathbf{e}$ of weight w in the space $\{\mathbf{e} : \mathbf{e}\mathbf{H}^\intercal = \mathbf{s}\}$ we search in $\cup_{i=1}^{z} \{\mathbf{e} : \mathbf{e}\mathbf{H}^\intercal = \mathbf{s}_i\}$.

The idea is to solve this problem with the Wagner's approach is to take $z \geq 3^{\ell/a}$ and replace the bottom-right list of the tree $\mathcal{L}_{2^a}$ by a list containing all the syndromes. Hence, there are only $2^a - 1$ lists to generate from the search space. Therefore, the constraint of Theorem 1 becomes:

$$3^{\ell/a} \leq 2^{(k+\ell)/(2^a-1)}.$$

For the practical parameters, we have $a = 6$ or $a = 7$ so the change from 2^a to $2^a - 1$ has a negligeable impact when we adapt the representation technique to the DOOM problem.

The DOOM parameters stated in [DST18] are derived from the complexity of a key attack detailed in the Wave paper. Our result stated in Section 4.4 provides another attack to consider. We computed the minimal parameters for the Wave scheme so that both attacks would have a time complexity of at least 2^{128}. They are stated in Table 3. These should be considered as the new parameters to use for the Wave scheme.

Public key size (in MB)	Signature length (in kB)
2.27	1.434

Table 3. New parameters of the Wave signature scheme for 128 bits of security.

6 Hardest Instances of Ternary Syndrome Decoding

In the previous sections, we looked to optimize our algorithms for the regime of parameters useful for the Wave signature scheme. The corresponding syndrome decoding problem, uesd $R = 0.676$ and $w \approx 0.948$. It's a regime where there are many solutions to the problem and hence Wagner's algorithm with a large number of floors was efficient. However, this is not the problem where the problem is the hardest.

We will now look at the hardest instances of the ternary syndrome decoding in large weight. As we already teased in the introduction, ternary SD is much harder in large weights than in small weights. In the two examples we considered, namely $R = 0.5$ and $R = 0.676$, the problem was the hardest for $W = 1$. As we will see, there are some lower rates for which the syndrome decoding is the hardest for $w < 1$.

Take an instance $(\mathbf{H}, \mathbf{s})$ of the $\mathrm{SD}(3, R, W)$ syndrome decoding problem with $W \geq \frac{1}{2}$. We have $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ of full rank, and $\mathbf{s} \in \mathbb{F}_q^{n-k}$. As in the binary case, the problem is the hardest where the problem has on average a unique solution, if such a regime exists.

Let $R_{\max} \stackrel{\triangle}{=} \frac{\log_2(3)-1}{\log_2(3)} \approx 0.36907$. For $R \in [0, R_{\max}]$, we define $W_{\mathrm{GV}}^{high}(R)$ as the unique value in $[1/2, 1]$ such that

$$W_{\mathrm{GV}}^{high}(R) + h_2(W_{\mathrm{GV}}^{high}(R)) = (1 - R)\log_2(3).$$

where $h_2(x) \stackrel{\triangle}{=} -x\log_2(x) - (1-x)\log_2(1-x)$. The rate $R_{\max}$ was defined such that $W_{\mathrm{GV}}^{high}(R_{\max}) = 1$ while this quantity is not defined for $R > R_{\max}$. This is why we didn't see a high peak for $R = 0.5$ and $R = 0.676$ but an increasing function up to $W = 1$.

In order to study the above problem for hard high weight instances; we looked at 3 standard algorithms: Prange's algorithm, Dumer's algorithm and the BJMM algorithm.

We then want to look at the hardest instance. We made a case study and showed that for all the algorithms above, the hardest case is actually for $R = R_{\max} \approx 0.36907$ and $W = 1$. We get the following results:

In both cases, we can see that Prange's algorithm does very poorly, but that Dumer already gives much better results and that the BJMM susbet sum techniques, i.e. representations, can also help. The analysis of Prange and Dumer for $q = 3$ is quite straightforward and follows closely the

Algorithm	$q=2$	$q=3$ and $W>0.5$
Prange	0.121 ($R=0.454$)	0.369 ($R=0.369$)
Dumer/Wagner	0.116 ($R=0.447$)	0.269 ($R=0.369$)
BJMM	0.102 ($R=0.427$)	0.247 ($R=0.369$)

Table 4. Best exponents with associated rates.

binary case. For BJMM, i.e. Wagner with representations, the 0.247 comes from a 2 level Wagner tree that includes 1 layer of representations. We searched larger Wagner trees but this didn't give us any improvements.

The differences we see are quite large, the ternary SD seems significantly harder than it's binary variant. It is expected to some extent because in the ternary case, the input matrices have elements in $\mathbb{F}_3$ and not $\mathbb{F}_2$ which makes them larger. But we don't expect such a big difference and it seems that ternary syndrome decoding is indeed much harder than its binary counterpart.

In order to confirm this idea, we define the following metric: what is the smallest input size for which we need 2^{128} time to solve it? We took 128, as 128 security bits is a cryptographic standard. The way we measure the input size is in the size of the matrix $\mathbf{H} \in \mathbb{F}_q^{n(1-R)\times n}$ in systematic form. This means we write

$$\mathbf{H} = \left(\mathbf{1}_{n(1-R)}\ \mathbf{H}'\right).$$

We actually only need to specify $\mathbf{H}'$ which requires $R(1-R)n^2\log_2(q)$ bits. We show that even in this metric, the ternary syndrome decoding problem is much harder, i.e. we require 2^{128} for much smaller input sizes. Our results are summarized in the table below.

Algorithm	$q=2$	$q=3$ and $W>0.5$
Prange	275 ($R=0.384$)	44 ($R=0.369$)
Dumer/Wagner	295 ($R=0.369$)	83 ($R=0.369$)
BJMM/Our algorithm	374 ($R=0.326$)	99 ($R=0.369$)

Table 5. Minimum input sizes (in Kbits) for a time complexity of 2^{128}.

Notice that in this metric, it is actually worth to reduce the rate R, as this reduces the input size. For the ternary case, we don't see this behavior, which shows that the problem quickly becomes simples, as R decreases.

The work we present here is very preliminary but opens many new perspectives. It seems there are many cases in code-based cryptography, from encryption schemes to signatures, where we could use this problem instead of the binary syndrome decoding problem to get better efficiency.

7 Conclusion

In this work, we stressed a strong difference between the cases $q=2$ and $q\geq 3$ of the Syndrome Decoding problem: namely, the symmetry between the small weight and the large weight cases, which occurs in the binary case, is broken for larger values of q. The large weight case of the general Syndrome Decoding problem had never been studied before. We propozed two algorithms to solve the Syndrome Decoding problem in this new regime in the context of the *Partial Gaussian Elimination and Subset Sum* framework. Our first algorithm uses the $q-ary$ Wagner's approach to solve the underlying Subset Sum problem. We proposed a second algorithm making use of representations as in the BJMM approach. We studied both algorithms and proposed a first application for cryptographic purposes, namely for the Wave signature scheme. Considering our complexity analysis, we proposed new parameters for this scheme. Furthermore, we showed that it happens that the worst case complexity of Syndrome Decoding in large weight is higher than in small weight. This implies that it should be possible to develop new code-based cryptographic schemes using this regime of parameters that reach the same security level with smaller key size. The first proposal Wave opens the way for a new family of code-based cryptographic schemes using large weights in the q-ary setting.

References

[ABB+17] Nicolas Aragon, Paulo Barreto, Slim Bettaieb, Loic Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Phillipe Gaborit, Shay Gueron, Tim Güneysu, Carlos Aguilar Melchor, Rafael Misoczki, Edoardo Persichetti, Nicolas Sendrier, Jean-Pierre Tillich, and Gilles Zémor. BIKE, December 2017. NIST Round 1 submission for Post-Quantum Cryptography.

[ACP+17] Martin Albrecht, Carlos Cid, Kenneth G. Paterson, Cen Jung Tjhai, and Martin Tomlinson. NTS-KEM. first round submission to the NIST post-quantum cryptography call, December 2017.

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

[BBC+19] Marco Baldi, Alessandro Barenghi, Franco Chiaraluce, Gerardo Pelosi, and Paolo Santini. LEDAcrypt. second round submission to the NIST post-quantum cryptography call, January 2019.

[BCJ11] Anja Becker, Jean-Sébastien Coron, and Antoine Joux. Improved generic algorithms for hard knapsacks. In *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, pages 364–385, 2011.

[BCL+17] Daniel J. Bernstein, Tung Chou, Tanja Lange, Ingo von Maurich, Ruben Niederhagen, Edoardo Persichetti, Christiane Peters, Peter Schwabe, Nicolas Sendrier, Jakub Szefer, and Wang Wen. Classic McEliece: conservative code-based cryptography. `https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-1/submissions/Classic_McEliece.zip`, November 2017. First round submission to the NIST post-quantum cryptography call.

[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.

[BMvT78] Elwyn Berlekamp, Robert McEliece, and Henk van Tilborg. On the inherent intractability of certain coding problems. *IEEE Trans. Inform. Theory*, 24(3):384–386, May 1978.

[CFG89] Mark Chaimovich, Gregory Freiman, and Zvi Galil. Solving dense subset-sum problems by using analytical number theory. *J. Complexity*, 5(3):271–282, 1989.

[CFS01] Nicolas Courtois, Matthieu Finiasz, and Nicolas Sendrier. How to achieve a McEliece-based digital signature scheme. In *Advances in Cryptology - ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 157–174, Gold Coast, Australia, 2001. Springer.

[CG90] John T Coffey and Rodney M Goodman. The complexity of information set decoding. *IEEE Transactions on Information Theory*, 36(5):1031–1037, 1990.

[CT17] 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.

[DST17] Thomas Debris-Alazard, Nicolas Sendrier, and Jean-Pierre Tillich. Surf: a new code-based signature scheme. preprint, September 2017. arXiv:1706.08065v3.

[DST18] Thomas Debris-Alazard, Nicolas Sendrier, and Jean-Pierre Tillich. Wave: A new code-based signature scheme. Cryptology ePrint Archive, Report 2018/996, October 2018. `https://eprint.iacr.org/2018/996`.

[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.

[FP05] Abraham Flaxman and Bartosz Przydatek. Solving medium-density subset sum problems in expected polynomial time. In *STACS 2005, 22nd Annual Symposium on Theoretical Aspects of Computer Science, Stuttgart, Germany, February 24-26, 2005, Proceedings*, pages 305–314, 2005.

[GKH17] Cheikh Thiécoumba Gueye, Jean Belo Klamti, and Shoichi Hirose. Generalization of BJMM-ISD using may-ozerov nearest neighbor algorithm over an arbitrary finite field \mathbb f_q. In *Codes, Cryptology and Information Security - Second International Conference, C2SI 2017, Rabat, Morocco, April 10-12, 2017, Proceedings - In Honor of Claude Carlet*, pages 96–109, 2017.

[GM91] Zvi Galil and Oded Margalit. An almost linear-time algorithm for the dense subset-sum problem. *SIAM J. Comput.*, 20(6):1157–1189, 1991.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 197–206. ACM, 2008.

[Hir16] Shoichi Hirose. May-ozerov algorithm for nearest-neighbor problem over Ʊq and its application to information set decoding. In *Innovative Security Solutions for Information Technology and Communications - 9th International Conference, SECITC 2016, Bucharest, Romania, June 9-10, 2016, Revised Selected Papers*, pages 115–126, 2016.

[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.

[IKR+18] Carmelo Interlando, Karan Khathuria, Nicole Rohrer, Joachim Rosenthal, and Violetta Weger. Generalization of the ball-collision algorithm. *arXiv preprint arXiv:1812.10955*, 2018.

[JJ02] Thomas Johansson and Fredrik Jönsson. On the complexity of some cryptographic problems based on the general decoding problem. *IEEE Trans. Inform. Theory*, 48(10):2669–2678, October 2002.

[Lyu05] Vadim Lyubashevsky. On random high density subset sums. *Electronic Colloquium on Computational Complexity (ECCC)*, 1(007), 2005.

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

[Meu17] Alexander Meurer. *A Coding-Theoretic Approach to Cryptanalysis.* PhD thesis, Ruhr University Bochum, November 2017.

[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.

[MTSB12] Rafael Misoczki, Jean-Pierre Tillich, Nicolas Sendrier, and Paulo S. L. M. Barreto. MDPC-McEliece: New McEliece variants from moderate density parity-check codes. *IACR Cryptology ePrint Archive, Report2012/409*, 2012, 2012.

[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.

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

[Sen11] Nicolas Sendrier. Decoding one out of many. In *Post-Quantum Cryptography 2011*, volume 7071 of *LNCS*, pages 51–67, 2011.

[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.