Classical and Quantum algorithms for generic Syndrome Decoding problems and applications to the Lee metric

André Chailloux[1], Thomas Debris-Alazard[2], and Simona Etinski[1]

[1] Inria de Paris, EPI COSMIQ
[2] Inria Saclay
{andre.chailloux, thomas.debris, simona.etinski}@inria.fr

Abstract. The security of code-based cryptography usually relies on the hardness of the syndrome decoding (SD) problem for the Hamming weight. The best generic algorithms are all improvements of an old algorithm by Prange, and they are known under the name of Information Set Decoding (ISD) algorithms. This work aims to extend ISD algorithms' scope by changing the underlying weight function and alphabet size of SD. More precisely, we show how to use Wagner's algorithm in the ISD framework to solve SD for a wide range of weight functions. We also calculate the asymptotic complexities of ISD algorithms, both for the classical and quantum case. We then apply our results to the Lee metric, which is currently receiving a significant amount of attention. By providing the parameters of SD for the Lee weight for which decoding seems to be the hardest, our study could have several applications for designing code-based cryptosystems and their security analysis, especially against quantum adversaries.

1 Introduction

Code-based cryptography is one of the leading proposals for post-quantum cryptography, and it traditionally relies on the hardness of the syndrome decoding problem. For fixed q, n, k, w, the problem is defined as follows: starting from a parity check matrix $\mathbf{H} \in \mathbb{F}_q^{n\times(n-k)}$, and a syndrome $\mathbf{s} \in \mathbb{F}_q^{n-k}$, the goal is to find a vector $\mathbf{e} \in \mathbb{F}_q^n$ such that $\mathbf{He} = \mathbf{s}$, and $\mathbf{e}$ has the Hamming weight[(3)] w. This problem has been studied for a long time, and mostly for the alphabet size $q = 2$. Despite many efforts, the best algorithms for solving this problem [Pra62, Ste88, Dum91, Bar97, MMT11, BJMM12, MO15] require an exponential running time, and they are all refinements of the original Prange's algorithm [Pra62]. As such, they are all commonly known under one name: Information Set Decoding (ISD) algorithms.

[(3)] The Hamming weight of a vector $\mathbf{e} = (e_1, \ldots, e_n)$ is $|\mathbf{e}|_{\mathrm{H}} \stackrel{\text{def}}{=} |\{i : e_i \neq 0\}|$.

It is, however, notoriously difficult to put the syndrome decoding problem into practice. For example, constructing an efficient signature scheme in code-based cryptography often requires utilizing pseudo-random functions, and some other cryptographic assumptions. A generalized version of the problem promises to be harder and to offer a more exploitable structure that leads to creating more efficient constructions. Like DURANDAL [ABG+19], some proposals replace the Hamming weight with the rank metric based weight, which allows designers to use a Schnorr-Lyubashevski type signature. Another proposal, WAVE signature scheme [DST19], utilizes syndrome decoding problem for which $q = 3$, and the Hamming weight is large. This further enables one to construct and exploit a trapdoor one-way preimage sampleable function, which would not be possible for $q = 2$ or $q = 3$ in small weight.

These examples already show the usefulness of going beyond $q = 2$ and the Hamming weight setting. We are, however, still at an early stage of using these variants for cryptographic schemes. Therefore, it is important to study their hardness, especially against quantum computers, since a big appeal of code-based cryptography is post-quantum security.

Our work. In this paper, we perform a generic analysis of different ISD algorithms. The analysis is applicable to any weight function $wt : \mathbb{F}_q^n \to \mathbb{R}_+$ satisfying $wt(\mathbf{e}) \stackrel{\text{def}}{=} \sum_{i=1}^{n} wt'(e_i)$, for some function $wt' : \mathbb{F}_q \to \mathbb{R}_+$, and $wt'(0) = 0$. However, we primarily focus on the Lee weight analysis, and the comparison between the Lee and Hamming weight. The reason we chose these two weight functions is that the two are commonly encountered in coding theory, and both led to proposals for cryptographic schemes.

Which ISD algorithms do we study here? We analyze algorithms by Prange and Stern/Dumer and the ISD algorithms based on Wagner's approach to solving a Generalized Birthday problem [Wag02]. Starting from [BCDL19], where classical algorithms for a ternary alphabet and the Hamming weight were analyzed, we broaden the analysis to the higher alphabet sizes, usage of a different weight function, and the study of both classical and quantum algorithms. This is the first time such a generic analysis of quantum ISD algorithms was done since the work of [KT17] that studied only the standard case of $q = 2$ and the Hamming weight.

To perform such a generic analysis, we need a way of computing sphere surface areas in a vector space endowed with an arbitrary metric. More precisely, we aim to calculate the sizes of sets of the form $\{\mathbf{e} \in \mathbb{F}_q^n : wt(\mathbf{e}) = p\}$. To do this, we start with the approach presented in [Ast84], applied to the Lee metric case, and we derive a convex optimization method for calculating the asymptotic sphere surface area independently of the metric. We thus provided a simple approach

to analyzing syndrome decoding problems in a vector space endowed with an arbitrary metric and a weight function derived from it.

Our framework can also be used for studying the security of the Restricted Syndrome Decoding problem [BBC+20a]. Nevertheless, it does not work for the rank metric norm where we do not know how to construct ISD algorithms better than Prange's algorithm[(4)].

Notations

Throughout the paper, we use $[n] \stackrel{\text{def}}{=} \{1, \ldots, n\}$ and, given a finite set $\mathscr{E}$, we denote by $|\mathscr{E}|$ its size. We consider a weight function $wt : \mathbb{F}_q^n \to \mathbb{R}_+$ which satisfies the following:

$$\exists wt' : \mathbb{F}_q \to \mathbb{R}_+ : wt'(0) = 0 \text{ and } \forall \mathbf{e} = (e_1, \ldots, e_n) \in \mathbb{F}_q^n, wt(\mathbf{e}) = \sum_i wt'(e_i). \tag{1}$$

This weight function is usually - but not always - obtained as $wt(\mathbf{x}) = d(\mathbf{x}, 0)$ where d is a distance. We will sometimes use the terminology of distance instead of weight when this is the case. When q and wt are fixed and explicit, we define the surface area of a sphere of weight w in a vector space of dimension n as:

$$S_w^n \stackrel{\text{def}}{=} \left|\{\mathbf{e} \in \mathbb{F}_q^n : wt(\mathbf{e}) = w\}\right|.$$

2 Quantum preliminaries

We refer to [NC00] for a basic introduction to quantum computing. In this paper, we use the canonical gate model where the running time of a quantum algorithm is the number of gates in its corresponding circuit description. We utilize the QRAM model, for which we assume the operation $U_{QRAM} : |i\rangle |y\rangle |b_1, \ldots, b_n\rangle \to |i\rangle |y + x_i\rangle |b_1, \ldots, b_n\rangle$ can be done in time $polylog(n)$ when each b_i is a single bit.

Grover's algorithm. [Gro96] For a function $f : \{0,1\}^n \to \{0,1\}$ that has an efficient classical description, Grover's algorithm can find x such $f(x) = 1$ in time $O(poly(n)2^{n/2})$ if such an x exists and output 'no solution' otherwise.

(4) There are other algorithms [BBB+20, BBC+20b] based on Gröbner basis that perform better than ISD algorithms for the rank metric.

Amplitude amplification. [BH97] Fix a function $f : \{0,1\}^n \to \{0,1\}$ that has an efficient classical description. Consider then a quantum algorithm $\mathscr{A}$ that outputs x such that $f(x) = 1$ with probability p and does not perform intermediate quantum measurements. Using amplitude amplification, one can find x such that $f(x) = 1$ by making $O(\frac{1}{\sqrt{p}})$ calls to $\mathscr{A}$. Notice that if we start from a classical algorithm $\mathscr{A}$, there are generic ways to run $\mathscr{A}$ coherently as a quantum algorithm $\mathscr{A}'$ that does not have intermediate quantum measurements and behaves exactly like $\mathscr{A}$.

3 Syndrome Decoding Problems

When we fix an alphabet size q and a weight function wt, the syndrome decoding problem is defined as follows:

Problem 1. Syndrome Decoding $\mathrm{SD}(n, k \leq n, w)$

- Input: A matrix $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$, a column vector (the syndrome) $\mathbf{s} \in \mathbb{F}_q^{n-k}$.
- Goal: Find a column vector $\mathbf{e} \in \mathbb{F}_q^n$ s.t. $\mathbf{He} = \mathbf{s}$ and $wt(\mathbf{e}) = w$.

The decision version of this problem, which asks whether there exists a vector $\mathbf{e}$ of weight w such that $\mathbf{He} = \mathbf{s}$, is NP-complete for $q = 2$ with the Hamming weight function [BMvT78].

Consider now the input distribution $\mathscr{D}$ sampled as follows: pick a random matrix $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ of rank $n-k$, pick a random $\mathbf{e} \in \mathbb{F}_q^n$ with $wt(\mathbf{e}) = w$, and output $(\mathbf{H}, \mathbf{s} = \mathbf{He})$. Notice that the problem always has at least one solution for this distribution and that SD is believed to be hard, even against quantum computers. That is why, in this paper, we study algorithms for SD with this input distribution. We only consider a prime q to avoid attacks that would use sub-fields of the alphabet field $\mathbb{F}_q$.

Another problem of interest, which we call Checkable Multiple Syndrome Decoding, is the following:

Problem 2. Checkable Multiple Syndrome Decoding $\mathrm{CMSD}(n, m, w, Y, Z)$

- Input: A matrix $\mathbf{H} \in \mathbb{F}_q^{m\times n}$, a syndrome $\mathbf{s} \in \mathbb{F}_q^m$.
- Goal: output the description of a function $f : [Y] \to \mathbb{F}_q^n$ such that f is efficiently computable, and $|\{\mathbf{e} : \mathbf{e} \in \mathrm{Im}(f), \mathbf{He} = \mathbf{s} \text{ and } wt(\mathbf{e}) = w\}| = Z$.

This problem is a bit funny looking at first sight, but we are interested in it because, in our framework, it is used as a building block for solving the generic SD problem. It is very similar to asking for Z solutions to the syndrome decoding problem. Indeed, from a description f, one can output Z solutions to

SD in time Y by enumerating all the $f(1), \ldots, f(Y)$. Reciprocally, if one can find Z solutions $\mathbf{e}_1, \ldots, \mathbf{e}_Z$ to $\mathrm{SD}(n, m, w)$ in time $T \geq Z$, then one can solve $\mathrm{CMSD}(n, m, w, Y, Z)$ by defining $f(i) = \mathbf{e}_i$.

In the quantum setting, we want to have access to the function f but without paying for a time cost of Z for writing down these solutions. That will allow us to search over solutions more efficiently, using Grover's algorithm, and also justifies the slightly odd definition. Another remark is that while f should be efficiently computable, it need not have an efficient description. Typically, f can store some large precomputed databases, but computing $f(x)$ will only query the database a small number of times.

4 Information Set Decoding Algorithms for any Metric

We present Information Set Decoding algorithms for SD, which consist of a partial Gaussian elimination followed by solving an instance of CMSD. The description here is essentially the one from [BCDL19] with the difference that here we use the CMSD problem.

4.1 Information Set Decoding Framework

Fix $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ of rank $(n-k)$ and $\mathbf{s} \in \mathbb{F}_q^{n-k}$. Recall that we want to find $\mathbf{e} \in \mathbb{F}_q^n$ such that $wt(\mathbf{e}) = w$ and $\mathbf{He} = \mathbf{s}$. Let us introduce ℓ, p, Y, and Z, four parameters of the system that we consider fixed for now. In this framework, an algorithm for solving $\mathrm{SD}(n, k, w)$ consists of 4 steps: a permutation step, a partial Gaussian Elimination step, a CMSD step, and a test step.

1. *Permutation step.* Pick a random permutation π. Let $\mathbf{H}_\pi$ be the matrix $\mathbf{H}$ with the columns permuted according to π. We now want to solve $\mathrm{SD}(n, k, w)$ 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. That happens with constant probability.[5] If the submatrix is of full rank, perform Gaussian elimination on the rows of $\mathbf{H}_\pi$ using the first $n-k-\ell$ columns. Let now $\mathbf{S} \in \mathbb{F}_q^{(n-k)\times(n-k)}$ be the invertible matrix corresponding to this operation. There are two matrices then, $\mathbf{H'} \in \mathbb{F}_q^{(n-k-\ell)\times(k+\ell)}$ and $\mathbf{H''} \in \mathbb{F}_q^{\ell\times(k+\ell)}$, such that:

$$\mathbf{SH}_\pi = \begin{pmatrix} \mathbf{1}_{n-k-\ell} & \mathbf{H'} \\ \mathbf{0} & \mathbf{H''} \end{pmatrix}.$$

[5] For $q = 2$, this happens with probability at least 0.288 and this probability increases as q increases (see [Coo00], for example).

A vector $\mathbf{e} \in \mathbb{F}_q^n$ can be written as $\mathbf{e} = \begin{pmatrix} \mathbf{e}' \\ \mathbf{e}'' \end{pmatrix}$, where $\mathbf{e}' \in \mathbb{F}_q^{n-k-\ell}$ and $\mathbf{e}'' \in \mathbb{F}_q^{k+\ell}$, and one can write $\mathbf{Ss} = \begin{pmatrix} \mathbf{s}' \\ \mathbf{s}'' \end{pmatrix}$, with $\mathbf{s}' \in \mathbb{F}_q^{n-k-\ell}$ and $\mathbf{s}'' \in \mathbb{F}_q^{\ell}$.

$$\begin{aligned} \mathbf{H}_\pi \mathbf{e} = \mathbf{s} &\iff \mathbf{SH}_\pi \mathbf{e} = \mathbf{Ss} \\ &\iff \begin{pmatrix} \mathbf{1}_{n-k-\ell} & \mathbf{H}' \\ \mathbf{0} & \mathbf{H}'' \end{pmatrix} \begin{pmatrix} \mathbf{e}' \\ \mathbf{e}'' \end{pmatrix} = \begin{pmatrix} \mathbf{s}' \\ \mathbf{s}'' \end{pmatrix} \\ &\iff \begin{cases} \mathbf{e}' + \mathbf{H}'\mathbf{e}'' = \mathbf{s}' \\ \mathbf{H}''\mathbf{e}'' = \mathbf{s}'' \end{cases} \end{aligned} \tag{2}$$

To solve the problem, we try to find a solution $\begin{pmatrix} \mathbf{e}' \\ \mathbf{e}'' \end{pmatrix}$ to the above system such that $wt(\mathbf{e}'') = p$ and $wt(\mathbf{e}') = w - p$.

3. *The* CMSD *step.* Solve CMSD$(k+\ell, \ell, p, Y, Z)$ on input $(\mathbf{H}'', \mathbf{s}'')$, and let f be the output function.
4. *The test step.* For each $i \in [Y]$, let $\mathbf{e}_i'' = f(i)$ and let $\mathbf{e}_i' = \mathbf{s}' - \mathbf{H}'\mathbf{e}_i''$. For each i such that $\mathbf{H}''\mathbf{e}'' = \mathbf{s}''$, Equation (2) ensures that $\mathbf{H}_\pi \begin{pmatrix} \mathbf{e}_i' \\ \mathbf{e}_i'' \end{pmatrix} = \mathbf{s}$. If $wt(\mathbf{e}_i'') = p$ and $wt(\mathbf{e}_i') = w - p$, $\mathbf{e}_i = \begin{pmatrix} \mathbf{e}_i' \\ \mathbf{e}_i'' \end{pmatrix}$ is therefore a solution to SD(n, k, w) on inputs $\mathbf{H}_\pi$ and $\mathbf{s}$. The solution to SD(n, k, w) can then be turned into a solution of the initial problem by permuting the indices, as detailed in Equation (3) below. If we do not find any solution after checking all $i \in [Y]$, we go back to step 1.

At the end of the protocol, we have a vector $\mathbf{e}$ such that $\mathbf{H}_\pi \mathbf{e} = \mathbf{s}$ and $wt(\mathbf{e}) = w$. Let $\mathbf{e}_{\pi^{-1}}$ be the vector $\mathbf{e}$ with the permuted coordinates according to π^{-1}. Hence,

$$\mathbf{H}\mathbf{e}_{\pi^{-1}} = \mathbf{H}_\pi \mathbf{e} = \mathbf{s} \quad \text{and} \quad wt(\mathbf{e}_{\pi^{-1}}) = wt(\mathbf{e}) = w. \tag{3}$$

Therefore, $\mathbf{e}_{\pi^{-1}}$ is a solution to the problem.

4.2 Information Set Decoding: Complexity Analysis (Classical and Quantum)

We fix q and a weight function wt. Recall that for any n and w, the surface area of a sphere (according to wt) of radius w in $\mathbb{F}_q^n$ is defined as:

$$S_w^n = |\{\mathbf{e} \in \mathbb{F}_q^n : wt(\mathbf{e}) = w\}|.$$

With this definition at hand, we now present the complexity analysis of the algorithm for solving SD(n, k, w) for fixed parameters ℓ, p, Y, Z (see section 4.1 for more details).

Lemma 1. *Let P_1 be the probability that at step 4, for a fixed i, $wt(\mathbf{e}'_i) = w-p$. We have:*

$$P_1 = \min\{1, O(\frac{S_{w-p}^{n-k-\ell}}{\max\{1, \min\{S_w^n q^{-\ell}, q^{n-k-\ell}\}\}})\}.$$

This lemma can be seen as a generalization of Proposition 2 of [BCDL19] (where a max was omitted) for any weight function.

Proof. Let $S = \{\mathbf{e} : wt(\mathbf{e}) = w \wedge \mathbf{H}_\pi \mathbf{e} = \mathbf{s}\}$ be the set of solutions to our syndrome decoding problem on input $\mathbf{H}_\pi, \mathbf{s}$. Let also $S_2 = \{\mathbf{e} = \begin{pmatrix} \mathbf{e}' \\ \mathbf{e}'' \end{pmatrix} : wt(\mathbf{e}) = w \wedge \mathbf{H}''\mathbf{e}'' = \mathbf{s}''\}$, where $\mathbf{H}''$ is the matrix from step 2. By definition, $S \subseteq S_2$, so we have that S has average size $\max\{1, S_w^n q^{-(n-k)}\}$ and S_2 has average size $\max\{S_w^n q^{-\ell}, 1\}$.
Fix i and $\mathbf{e}''_i = f(i)$ satisfying $\mathbf{H}''\mathbf{e}''_i = \mathbf{s}''$ and $wt(\mathbf{e}''_i) = p$. $T_i = \left\{\mathbf{e}_i = \begin{pmatrix} \mathbf{e}'_i \\ \mathbf{e}''_i \end{pmatrix} : wt(\mathbf{e}_i) = w\right\}$. T_i is of average size $S_{w-p}^{n-k-\ell}$. Step 4 will find a solution if $T_i \cap S \neq \emptyset$. Since $T_i \subseteq S_2$ and is uniformly distributed in this set, this happens with the following probability:

$$\begin{aligned} P_1 &= \min\{1, O(\frac{|T_i||S|}{|S_2|})\} = \min\{1, O(\frac{S_{w-p}^{n-k-\ell} \cdot \max\{1, S_w^n q^{-(n-k)}\}}{\max\{S_w^n q^{-\ell}, 1\}})\} \\ &= \min\{1, O(\frac{S_{w-p}^{n-k-\ell}}{\max\{1, \min\{S_w^n q^{-\ell}, q^{n-k-\ell}\}\}})\}. \end{aligned}$$

□

We now present our generic formula for the running time of the Information Set Decoding algorithm from Section 4.1.

Proposition 1. *Fix parameters ℓ, p, Y, and Z of the information set decoding algorithm. The classical running time of the algorithm, T_{ISD}, is given as:*

$$T_{\mathrm{ISD}} = O\left(\max\left\{1, \frac{1}{P_1 Z}\right\} \cdot (\mathrm{poly}(n) + T_{\mathrm{CMSD}} + \mathrm{poly}(n)Y)\right),$$

where P_1 is the probability from the above lemma, and T_{CMSD} is the running time of step 3, i.e., the time required for solving $\mathrm{CMSD}(k+\ell, \ell, p, Y, Z)$.

Proof. Steps 1 and 2 take time $\mathrm{poly}(n)$, step 3 takes time T_{CMSD}, and step 4 takes time $\mathrm{poly}(n)$ for each $i \in [Y]$, hence the right part of the expression. How many times does the algorithm loop over this process? Step 2 succeeds with constant probability, and step 4 finds a solution with probability $1-(1-P_1)^Z$,

so it loops over the steps $O\left(\frac{1}{1-(1-P_1)^Z}\right) = O\left(\max\left\{1, \frac{1}{P_1 Z}\right\}\right)$ times, hence the result. □

The quantum setting. Our formulation allows for a simple extension to the quantum setting. We consider the algorithm described earlier with the following two changes: (1) in step 4, the algorithm uses Grover's search to check whether there is i such that $f(i)$ gives us a solution; (2) for each loop, *i.e.*, each time the algorithm starts from step 1, it finds a solution with probability $p = \Omega(\min\{1, P_1 Z\})$. This loop can be made coherently with a quantum algorithm $\mathscr{A}$ that does not do intermediate measurements and outputs a solution with probability p. The algorithm then use amplitude amplification to find a solution by repeating the loop $O(\frac{1}{\sqrt{p}})$ times.

Proposition 2. *Fix parameters ℓ, p, Y, and Z of the information set decoding algorithm. The quantum running time of the algorithm, T^Q_{ISD}, is given as:*

$$T^Q_{\mathrm{ISD}} = O\left(\sqrt{\max\left\{\frac{1}{ZP_1}, 1\right\}} \cdot \left(\mathrm{poly}(n) + T_{\mathrm{CMSD}} + \mathrm{poly}(n)\sqrt{Y}\right)\right),$$

where P_1 is the probability from Lemma 1, and T_{CMSD} is the running time of step 3*, i.e., of solving* $\mathrm{CMSD}(k+\ell, \ell, p, Y, Z)$.

Proof. Again, Steps 1 and 2 take time $\mathrm{poly}(n)$, and step 3 takes time T_{CMSD}. In step 4, the algorithm runs Grover's search, so this whole step takes time $\mathrm{poly}(n)O(\sqrt{Y})$. That can be done because the function on input i determines whether $wt\,(\mathbf{e}'_i) = w - p$ runs in polynomial time (since f runs in polynomial time). As we described above, we repeat the loop $O\left(\sqrt{\max\left\{\frac{1}{ZP_1}, 1\right\}}\right)$ times, which gives the result. □

The full ISD algorithm. To find the best ISD algorithm for solving $\mathrm{SD}(n, k, w)$, we minimize the running time of the algorithm presented earlier over parameters p, ℓ, Y, and Z. In many cases, we do not have full control over Y and Z, which are predetermined from other values. For instance, in Wagner's algorithm, we present next, there is an extra parameter a (the number of levels) that predetermines Y and Z, so we optimize over p, ℓ, and a.

5 Solving CMSD

This section presents our analysis of the application of Wagner's algorithm [Wag02] to solving $\mathrm{CMSD}(N, m_0 N, \omega_0 N, Y, Z)$[6]. We first present the list merg-

[6] As Wagner's algorithm is used for solving Generalized Birthday Problem, it can be easily seen that is well suited for solving CMSD problems, too.

ing procedure, which we utilize throughout the section, and then the two versions of our algorithm: the first one that aims to solve the CMSD problem using classical algorithms only, and the second one that utilizes both classical and quantum algorithms.

Notice here the change of the variables' names when referring to the CMSD problem. It is introduced so that our statements can be made independently of the previous section. Notice also that the asymptotic values of the algorithms' running times are calculated when N goes to $+\infty$ and that when presenting a proof, we ignore all the polynomial and constant terms.

5.1 List Merging

Let us take 3 lists of vectors in $\mathbb{F}_q^n$: L_1, L_2, and L. Take also a set $J \subseteq [n]$ and a random vector $\mathbf{t} \in \mathbb{F}_q^{|J|}$. The merging of L_1 and L_2 into L is done using the following algorithm:

List merging algorithm.

- Start from an empty list L, and sort the elements of L_1 according to the lexicographic order on the J coordinates.
- For each vector $\mathbf{y} \in \mathbb{F}_q^n$ from the list L_2, search for elements $\mathbf{x} \in \mathbb{F}_q^n$ of L_1 that satisfy: $\mathbf{x}_{|J} = \mathbf{y}_{|J} + \mathbf{t}_{|J}$, where $\mathbf{x}_{|J} \stackrel{\text{def}}{=} (x_j)_{j \in J}$, $\mathbf{y}_{|J} \stackrel{\text{def}}{=} (y_j)_{j \in J}$, and $\mathbf{t}_{|J} \stackrel{\text{def}}{=} (t_j)_{j \in J}$. For each solution found, add $\mathbf{x} + \mathbf{y}$ in L and register the references to $\mathbf{x}$ and $\mathbf{y}$.

Running time. Sorting L_1 on J coordinates is done in time $O(\log(|L_1|))$ using dichotomic search. If there are $s_{\mathbf{y}}$ solutions for a fixed $\mathbf{y}$, the algorithm takes $O(s_{\mathbf{y}} \log(|L_1|))$ time to find them, and the total size of L is $\sum_{\mathbf{y}} s_{\mathbf{y}}$. Therefore, the algorithm takes time $\widetilde{O}(|L_1|)$ for the first step, *i.e.*, to sort L_1, and it takes $\widetilde{O}(\max\{|L_2|, \sum_i s_{\mathbf{y}}\})$ for the second step. Overall, the algorithm takes time $\widetilde{O}(\max\{|L_1|, |L_2|, |L|\})$.

Expected number of solutions. If the elements in L_1 and L_2 are random vectors in $\mathbb{F}_q^n$, there is, on average, $|L| = \frac{|L_1||L_2|}{q^{|J|}}$ elements in the merged list.

List merging operator. To enable a succinct representation of this procedure in the rest of the text, we define the list merge operator on a set J and random vector $\mathbf{t}$, denoted as $\bowtie_J^{\mathbf{t}}$:

$$L = L_1 \bowtie_J^{\mathbf{t}} L_2 = \{\mathbf{x} + \mathbf{y} : \mathbf{x} \in L_1, \mathbf{y} \in L_2, \mathbf{x}_{|J} + \mathbf{y}_{|J} = \mathbf{t}_{|J}\}.$$

5.2 First Variant

We present here an approach to solving the CMSD problem, based on Wagner's algorithm [Wag02], which utilizes classical algorithms only and is closely related to the original Wagner's algorithm.

We start from relevant definitions. For a number of levels a, where $2^a|n$, and for each $i \in [2^a]$, we define:

$$\mathscr{I}_i \stackrel{\text{def}}{=} \{\mathbf{b} \in \mathbb{F}_q^n : \mathbf{b} = (\mathbf{0}^{(i-1)n/2^a}, \mathbf{b}_i, \mathbf{0}^{(2^a-i)n/2^a}) \text{ with } \mathbf{b}_i \in \mathbb{F}_q^{n/2^a} \wedge wt(\mathbf{b}_i) = N\omega_0/2^a\},$$

$$L_i^f \stackrel{\text{def}}{=} \{\mathbf{H} \cdot \mathbf{b}\}_{\mathbf{b} \in \mathscr{I}_i}, \text{ (7)}$$

where $\mathbf{H}$ is a parity check matrix, defined in Section 4.1.

The sets used for the indexing the lists in the merging procedure (as described in Section 5.1) are chosen so that they form a partition of $[n]$, *i.e.*:

$$\forall j, j' \in [a], \quad J_j \subseteq [n], \quad \bigcup_j J_j = [n], \quad J_j \cap J_{j'} = \emptyset, \text{ when } j \neq j'.$$

The random vectors (again, described in Section 5.1) are chosen such that they satisfy the following constraint:

$$\forall i \in [2^a], \quad \forall j \in [a], \quad \mathbf{t}_j^i \in \mathbb{F}_q^n, \quad \sum_i (\mathbf{t}_j^i)_{|J_j} = \mathbf{s}_{|J_j},$$

where $\mathbf{s}_{|J_j}$ refers to the syndrome, from Section 4.1, indexed by J_j.

List creation and merging. The algorithm starts by constructing 2^a lists of the same sizes: $L_i \subseteq L_i^f$, for all $i \in [2^a]$ (8). At each algorithm's level, the lists are then taken by pairs, $\{L_{2i-1}, L_{2i}\}$, and merged using the list merging procedure described in the previous subsection. More precisely, at the first level, the pairs are merged on a set J_1 and a random vector $\mathbf{t}_1^i$ (*i.e.*, $\bowtie_{J_1}^{\mathbf{t}_1^i}$ is performed). From the 2^{a-1} created lists, at the second level, pairs are taken again and merged similarly using the operator $\bowtie_{J_2}^{\mathbf{t}_2^i}$, for each $i \in [2^{a-2}]$. The same procedure continues up to the top level, where only 2 lists remain and the list merging is performed using $\bowtie_{J_a}^{\mathbf{t}_a^1=\mathbf{s}_{|J_a}}$. A function f, required for the CMSD problem, is then constructed using the method described in Section 3.

(7) From the definitions, it can be easily seen that $|L_i^f| = \frac{s_{\omega_0}}{2^a}$.

(8) There are previous description where $L_i = L_i^f$, but the inclusion improves the algorithm efficiency.

One can check that the final list created by this algorithm contains solutions to the problem. In particular, elements of top level's list are of the form $\mathbf{H} \cdot \mathbf{b}$, with $wt(\mathbf{b}) = N\omega_0$. That comes from the property of the weight function we use (see Equation (1)) and the definitions given earlier in this subsection. An example of the algorithm for $a = 3$, *i.e.*, three levels algorithm, is presented below.

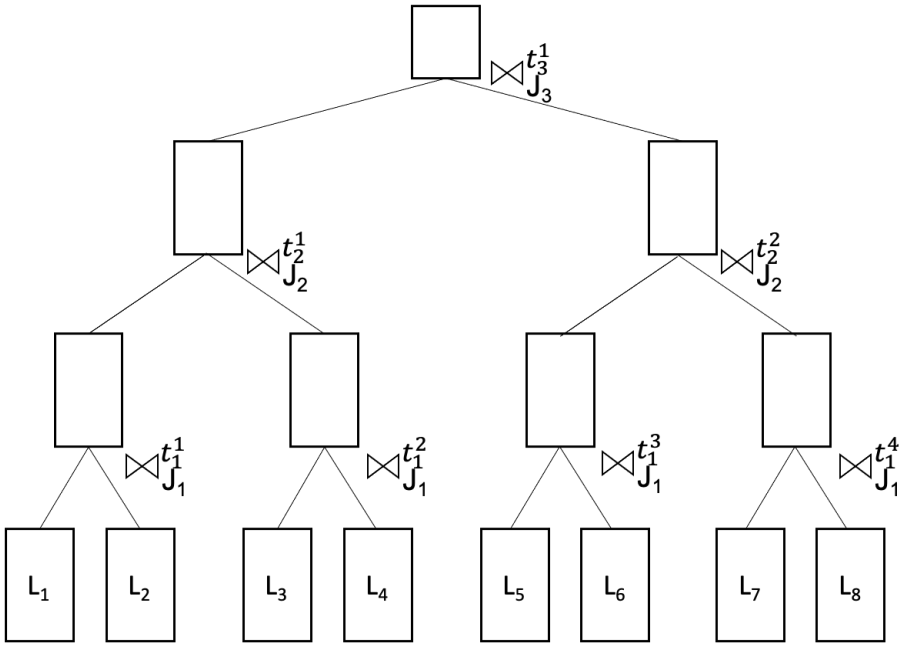

Fig. 1. First variant of Wagner's based algorithm for $a = 3$.

Proposition 3. [9] *Fix parameters m_0, ω_0, as well as a number of levels, a. Let $s_{\omega_0} = \lim_{n\to\infty} \frac{1}{n} \log_q(S^n_{n\omega_0})$, $u = \min\{\frac{s_{\omega_0}}{2^a}, m_0/a\}$, and $x = m_0 - (a-1)u$. The first variant of the algorithm on a levels solves the* $\mathrm{CMSD}(N, m_0N, \omega_0N, Y, Z)$ *problem in time T_{CMSD}, where*

$$Z = q^{N(2u-x+o(1))},\ T_{\mathrm{CMSD}} = q^{N(u+o(1))},\ Y = T_{\mathrm{CMSD}},$$

and the $o(1)$ hides an expression that goes to 0 as N goes to $+\infty$.

[9] Notice that Y and Z in both propositions are determined by m_0, ω_0, and a and cannot be chosen arbitrarily.

Proof. Let us take all bottom list $L_i, \ldots, L_{2^a}$, to be random subsets of size q^{Nu} of $L_i^f, \ldots, L_{2^a}^f$, respecitvely [(10)]. Without loss of generality, we also choose $|J_j| = u$, for $j \in [2^{a-1}]$, and $|J_a| = x$. We thus have that $Y = q^{Nu}$. Furthermore, from the merging algorithm, described earlier in this subsection, we know that all the lists up to the top level are of size q^{Nu}, and the list at the top level is of size $q^{N(2u-x)}$. As all the elements in the final list are solutions to the original problem, we expect $Z = q^{N(2u-x)}$ solutions, on average. All the list mergings take time q^{Nu}, except the last one that takes time $q^{N(2u-x)}$, hence $T_{CMSD} = \max(q^{Nu}, q^{N(2u-x)})$. From the proposition, we know that $u \leq m_0/a$ and $x = m_0 - (a-1)u$, which implies $x \geq m_0/a \geq u$, and thus $T_{CMSD} = q^{Nu}$. Therefore, we have an algorithm that finds $Z = q^{N(2u-x)}$ solutions in time $T_{CMSD} = q^{Nu}$, and for $Y = T_{CMSD} = q^{Nu}$. □

5.3 Second Variant

Starting from the original Wagner's algorithm [Wag02], we derive a quantum version of it and utilize it as part of an algorithm that solves the CMSD problem. Our results are presented in the rest of the section.

We start from relevant definitions. For a number of levels a, where $2^{a+1}|n$ and, for each $i \in [2^a - 1]$, we define:

$$\mathscr{I}_i \stackrel{\text{def}}{=} \{\mathbf{b}_i \in \mathbb{F}_q^n : \mathbf{b}_i = (\mathbf{0}^{(i-1)n/(2^a+1)}, \widetilde{\mathbf{b}_i}, \mathbf{0}^{((2^a+1)-i)n/(2^a+1)}) \\ \text{with } \widetilde{\mathbf{b}_i} \in \mathbb{F}_q^{n/(2^a+1)} \wedge wt\left(\widetilde{\mathbf{b}_i}\right) = N\omega_0/(2^a+1)\},$$

$$L_i^f \stackrel{\text{def}}{=} \{\mathbf{H} \cdot \mathbf{b}_i\}_{\mathbf{b}_i \in \mathscr{I}_i}.$$

For $i = 2^a$, we let:

$$\mathscr{I}_{2^a} \stackrel{\text{def}}{=} \{\mathbf{b}_{2^a} \in \mathbb{F}_q^n : \mathbf{b}_{2^a} = (\mathbf{0}^{(2^a-1)n/(2^a+1)}, \widetilde{\mathbf{b}_{2^a}}) \\ \text{with } \widetilde{\mathbf{b}_{2^a}} \in \mathbb{F}_q^{2n/(2^a+1)} \wedge wt\left(\widetilde{\mathbf{b}_{2^a}}\right) = 2N\omega_0/(2^a+1)\},$$

$$L_{2^a}^f \stackrel{\text{def}}{=} \{\mathbf{H} \cdot \mathbf{b}_{2^a}\}_{\mathbf{b}_{2^a} \in \mathscr{I}_{2^a}}, \text{ [(11)]}$$

[(10)] Notice that $\lim_{n\to\infty} \frac{1}{N} \log_q |L_i^f| = \lim_{N\to\infty} \frac{1}{N} \log_q S_{N\omega_0/2^a}^{N/2^a} = \frac{1}{2^a} s_{\omega_0}$, so we can choose asymptotically any $u \leq \frac{s_{\omega_0}}{2^a}$.

[(11)] From the definitions, it can be easily seen that $|L_i^f| = \frac{s_{\omega_0}}{2^a+1}$, for all $i \in [2^a - 1]$, and $|L_{2^a}^f| = \frac{2s_{\omega_0}}{2^a+1}$, for $i = a$.

In both cases, $\mathbf{H}$ is a parity check matrix, which is defined in Section 4.1.

Like in the first variant of the algorithm, the indexing sets, $J_1, \ldots, J_a$, are chosen so that they form a partition of $[n]$. The random vectors, $\mathbf{t}_j^i \in \mathbb{F}_q^n$, for all $i \in [2^a]$ and all $j \in [a]$, also satisfy the same constraints as in the first variant (for more details, see Section 5.2).

In this variant, all the bottom lists, $L_1, \ldots, L_{2^a-1}$, are of the same sizes, except the rightmost one, L_{2^a}, which is quadratically larger than the others. We thus change our definitions of L_i^f accordingly (see definitions above). In contrast to the first variant, the algorithm does not create the rightmost list. It instead computes and sorts the other lists in lexicographical order on the indices of corresponding J_j, for all $j \in [a]$. For each element of L_{2^a}, it then finds a corresponding element (if one exists) in the top list using an efficient (quantum) routine, for example, Grover's search algorithm. For the rest of the lists, the algorithm use the same merging method as in the first variant (see Section 5.2). An example of the algorithm on three levels is presented below.

Let us now construct the function f as it is required for the CMSD problem. First, let $\mathbf{y}_{2^a}^1, \ldots, \mathbf{y}_{2^a}^Y$ be the elements of L_{2^a}, *i.e.*, the elements of the bottom right list. For a fixed k, we aim to find $\mathbf{y}'_1, \ldots, \mathbf{y}'_{2^a-1}$ that satisfy the following: for $\forall i \in [2^a - 1]$, $\mathbf{y}'_i \in L_i$ and $\sum_i \mathbf{y}'_i + \mathbf{y}_{2^a}^k = \mathbf{s}$. If they exist, for each i, we find the associated $\mathbf{b}_i$ (from the definition of $\mathscr{I}_i$ above) such that $\mathbf{H}\mathbf{b}_i = \mathbf{y}'_i$ and $\mathbf{H}\mathbf{b}_{2^a} = \mathbf{y}_{2^a}^k$. If there are several such combinations, we take the first one according to the lexicographical order. Finally, let us take $\mathbf{e}_k = \sum_i \mathbf{b}_i$, so that we have $\mathbf{H}\mathbf{e}_k = \mathbf{s}$. We then define f as follows:

$$f(k) = \begin{cases} \mathbf{e}_k, \text{ if such a vector exists,} \\ \mathbf{0}, \text{ otherwise.} \end{cases}$$

The function f then can be described as follows. On an input k, f takes $\mathbf{y}_{2^a}^k$, from the list L_{2^a}, and checks if it can be summed with $\mathbf{y}'_{2^a-i}$ from the left neighbouring list, L_{2^a-i}, so that they appear in the solution sum. Again, if we have several such combinations, we take any one of them, for example, the first one in lexicographical order. The function repeats that at each level until it fails (in which case it outputs $\mathbf{0}$), or it arrives to the top list, where it outputs the corresponding $\mathbf{e}_k$.

Proposition 4. [9] *Fix parameters m_0, ω_0, as well as a number of levels, a. Let $s_{\omega_0} = \lim_{n\to\infty} \frac{1}{n} \log_q(S_{n\omega_0}^n)$, $u' = \min\{\frac{s_{\omega_0}}{2^a+1}, m_0/a\}$, and $x = m_0 - (a-1)u$. The second variant of the algorithm on a levels solves the* $\mathrm{CMSD}(N, m_0 N, \omega_0 N, Y, Z)$ *problem in time T_{CMSD}, where*

$$Z = q^{N(3u'-x+o(1))},\ T_{\mathrm{CMSD}} = q^{N(u'+o(1))},\ Y = q^{N(2u'+o(1))},$$

and the $o(1)$ hides an expression that goes to 0 as N goes to $+\infty$.

Proof. We choose lists $L_1, \ldots, L_{2^a-1}$ to be random subsets of size $q^{Nu'}$ of $L_1^f, \ldots, L_{2^a-1}^f$, respectively. We also choose L_{2^a} so that is a random subset of $L_{2^a}^f$ and that it is of size $q^{N2u'}$. Without loss of generality, we choose J_j such that $|J_j| = u'$, for all $j \in [2^{a-1}]$, and $|J_a| = x$. We then have that $Y = q^{2Nu'}$. After the list merging at each level up to the top one, the new lists are of expected size $q^{Nu'}$, except the rightmost one, at each level, that is of expected size $q^{N2u'}$. At the top level, there is one list of the expected size $q^{Nu'}$ and one of the expected size $q^{N2u'}$. Since $|J_a| = x$, the expected size of the top list, that is the expected number of solutions to be find by the algorithm, is $Z = q^{N(3u'-x)}$. The time for which the algorithm finds Z solutions is calculated as follows. Constructing and sorting the lists to compute f take time $q^{N(u'+o(1))}$ (omitting the constant multiplicative term 2^a), but computing f afterwards take polynomial time, so we finally have $T_{CMSD} = q^{N(u'+o(1))}$. The number of k such that $f(k)$ outputs a good solution is actually the size of L^{top}, *i.e.*, $q^{N(3u'-x)}$ and, since $f : [Y] \to \mathbb{F}_q^n$, this proves our proposition. □

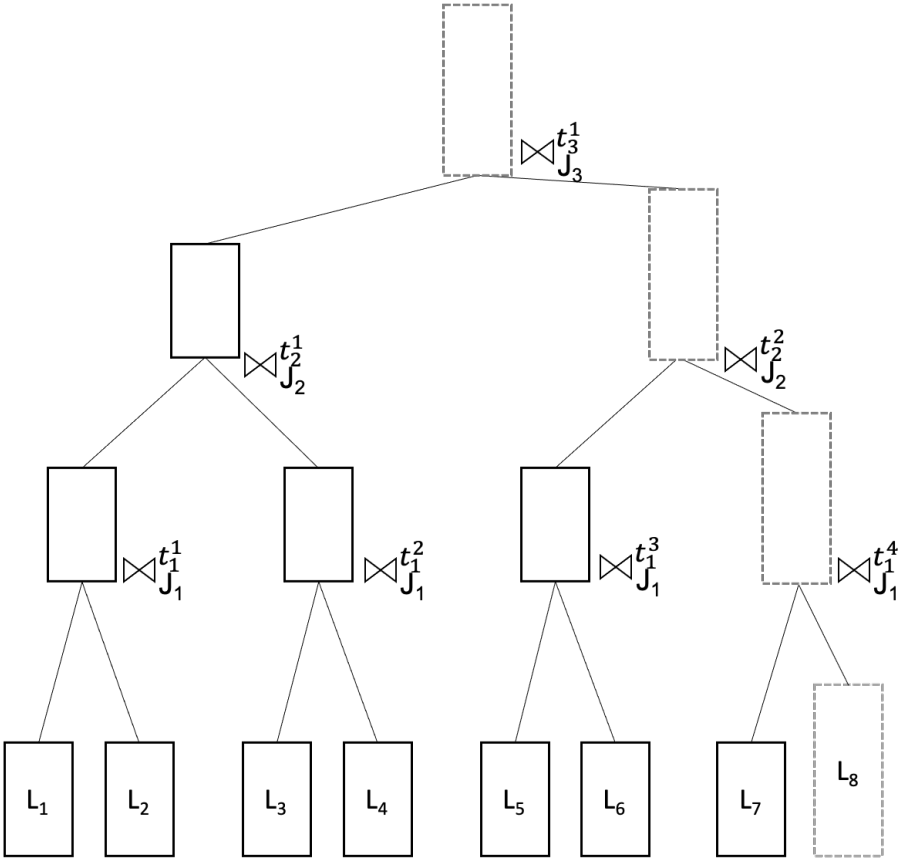

Fig. 2. Second variant of Wagner's based algorithm for $a = 3$.

Final remarks. Our ISD algorithm aims to solve an instance of CMSD$(k+\ell, \ell, p, Y, Z)$. That means we can use the above propositions to calculate the asymptotic running time of the algorithm described in section 4.1. We first define $N = k + \ell, m_0 = \frac{\ell}{k+\ell}$, and $\omega_0 = \frac{p}{k+\ell}$, and then do the following: we plug Proposition 3 into Proposition 1, for the classical case, and plug Proposition 4 into Proposition 2, for the quantum case. We then optimize parameters of our ISD algorithm over k, ℓ, and a by minimizing the algorithm's running. From the values of k, ℓ, and a, we finally extract Y and Z and obtain the asymptotic running time of the algorithm in both the classical and quantum case.

6 Computing Surface Area of a Sphere

We here rely primarily on the combinatorial approach presented in [Ast84]. Some of the other methods are shown in more recent papers as, for example, [GS91], [BB19], [WKH+21]. We decided to use the approach from [Ast84] as it enables us to derive a generic method for calculating the asymptotic value of the sphere surface area independently of the weight function and the alphabet size.

Proposition 5. *Fix a parameter q, and a weight function wt' satisfying Equation 1. Let the set C be defined as follows:*

$$C \stackrel{def}{=} \{\mathbf{c} = (c_1, \cdots, c_q) : i \in [q], c_i \in \mathbb{N}, \sum_{i=1}^{q} c_i = n, \sum_{i=1}^{q} c_i wt'(i) = w\},$$

where $w \in \mathbb{N}, \quad w \leq n \max_{i \in \{1,\cdots,q\}} wt'(i)$. The sphere surface area, and its corresponding asymptotic value when n gows to $+\infty$, are given by the following expressions:

$$S_w^n = \sum_{\mathbf{c} \in C} \binom{n}{\mathbf{c}} \text{ (12)}. \tag{4}$$

$$s_\omega = \lim_{n \to +\infty} \max_{\mathbf{c} \in C} \left(\sum_{i=1}^{q} -\frac{c_i}{n} \log_q \frac{c_i}{n} \right). \tag{5}$$

Proof. Let us first take a multiset of size n where elements are taken from $[q]$, and each element is repeated c_i times, for each $i \in [q]$. The number of permutations of such a multiset is given by the multinomial coefficient, defined as $\binom{n}{c_1,\ldots,c_q} \stackrel{\text{def}}{=} \frac{n!}{c_1! \ldots c_q!}$. This number corresponds to the number of vectors consisting of c_1 ones, c_2 twos, ..., c_q values of q. By the definition of the set C, and the sphere surface area, we thus have $S_w^n = \sum_{\mathbf{c} \in C} \binom{n}{\mathbf{c}}$.

(12) $\binom{n}{\mathbf{c}}$ denotes a multinomial coefficient.

Given the classical combinatorial result for the number of multinomial coefficients for a fixed n and q, the size of a set C, and thus the number of the elements in the sum, is upper bounded by $\binom{n+q-1}{q-1}$. The upper and lower bounds of S_w^n are then given by $\max_{\mathbf{c}\in C}\binom{n}{\mathbf{c}} \leq S_w^n \leq \binom{n+q-1}{q-1}\max_{\mathbf{c}\in C}\binom{n}{\mathbf{c}}$.

Following the same line of reasoning as in [Ast84], *i.e.*, by taking log_q of each part of the equation above, multiplying them by $\frac{1}{n}$, where $n \to +\infty$, and using Stirling's approximation we finally obtain: $s_\omega = \lim_{n\to+\infty} \max_{\mathbf{c}\in C}\left(\sum_{i=1}^{q} -\frac{c_i}{n}\log_q\frac{c_i}{n}\right)$. □

This proposition can be observed as a generalization of the combinatorial approach presented in [Ast84] for any weight function and arbitrary alphabet size. Using the same reasoning, we calculate the asymptotic value of the sphere surface area, s_ω, by reducing the Expression 5 to the following convex optimization problem:

Problem 3. Let $\boldsymbol{\lambda} = (\lambda_1, ..., \lambda_{q-1})$, and $\lambda_i \in \mathbb{R}_+$ for each $i \in [q]$.

- Maximize: $-\sum_{i=1}^{q} \lambda_i \log_q \lambda_i$,
- Subject to: $\sum_{i=1}^{q} \lambda_i = 1, \quad \sum_{i=1}^{q} \lambda_i wt'(i) = \omega$.

It can be easily verified that when replacing the optimization variable λ_i with c_i/n from (5), the optimization problem remains convex. If we denote by $\tilde{\boldsymbol{\lambda}} = (\tilde{\lambda}_0, \tilde{\lambda}_1, ..., \tilde{\lambda}_{q-1})$ the solution of Problem 3, the asymptotic value of the sphere surface area is calculated as $s_\omega = -\sum_{i=0}^{q-1} \tilde{\lambda}_i \log_q \tilde{\lambda}_i$. Notice here that we do not compute only the surface areas but also the typical weight pattern of words of Lee weight w, *i.e.* the $\mathbf{c} \in C$ that maximizes the quantity in Equation 5. This is necessary if we want to use this problem in Stern's signature scheme.

It can be shown that Problem 3 belongs to the subclass of the convex optimization problems, namely the class of conic optimization problems [BV14]. As such, it is susceptible to solving via MOSEK solver [ApS21], so we utilize MOSEK as a primary computational tool. Nevertheless, to be solved via MOSEK, Problem 3 needs to be transformed so that it aligns with the standard form of conic optimization problems, as presented in the following problem:

Problem 4. Let $\lambda \stackrel{\text{def}}{=} (\lambda_1, ..., \lambda_q) \in \mathbb{R}_+^q$ and $\tau \stackrel{\text{def}}{=} (\tau_1, ..., \tau_q) \in \mathbb{R}_+^q$.

- Maximize: $\sum_{i=1}^{q} \tau_i$,
- Subject to: $\sum_{i=1}^{q} \lambda_i = 1, \quad \sum_{i=1}^{q} \lambda_i \; wt'(i) = \omega, \quad (1, \lambda, \tau) \in K_{exp}$.

where the constraint $(1, \lambda, \tau) \in K_{exp}$ means that $\tau_i \leq -\lambda_i \log_q \lambda_i$, for each $i \in [q]$.[13] It can be easily verified that Problem 3 and Problem 4 are equivalent, hence finding a solution of either of the two yields the asymptotic value of the sphere surface area.

7 Results

We use our framework to compare SD with the Hamming and Lee weight. For $q = 2$ and $q = 3$, the weight functions are the same by their definitions. For $q > 3$, however, our numerical results show that the asymptotic complexities of the problem differ in these two cases and that the problem is indeed harder in the Lee weight case. We present here the comparison of the complexities of our classical ISD algorithm in the Lee and Hamming weight setting and in the parameter range that is interesting from the perspective of the hardest instances of the SD problem. It can be easily verified that the complexity of the hardest instances of the Lee SD problem is indeed higher than that of the hardest Hamming instances.

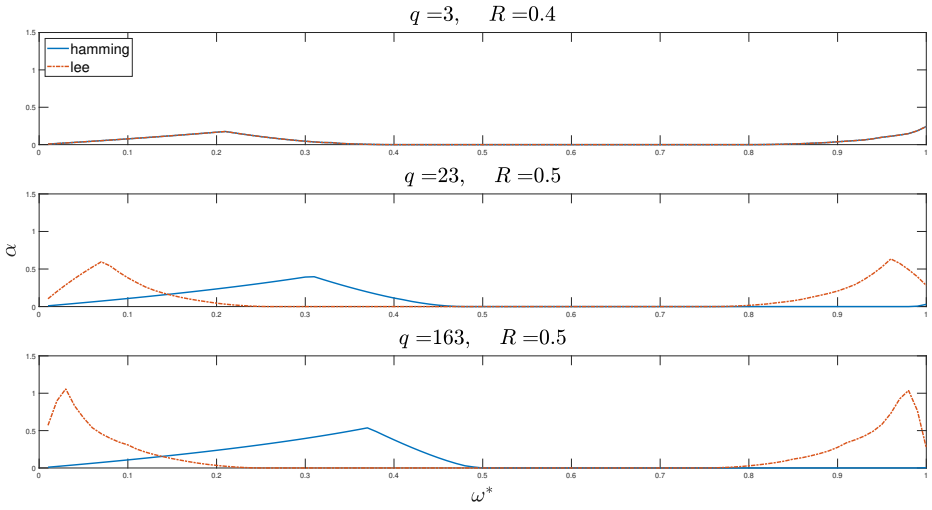

Fig. 3. Comparison of the Hamming and Lee SD problem: for a fixed q and R, the exponents α s.t. $Time = 2^{\alpha n}$ are given as a function of ω^*, where $\omega^* = \omega$ in the Hamming weight case, and $\omega^* = \omega\lfloor q/2 \rfloor$ in the Lee weight case.

[13] The notation K_{exp} comes from the MOSEK optimizer[ApS21] and represents the exponential convex cone.

In the rest of the analysis, we focus on the SD problem in Lee weight. The following plot illustrates some of the numerical results we obtain.

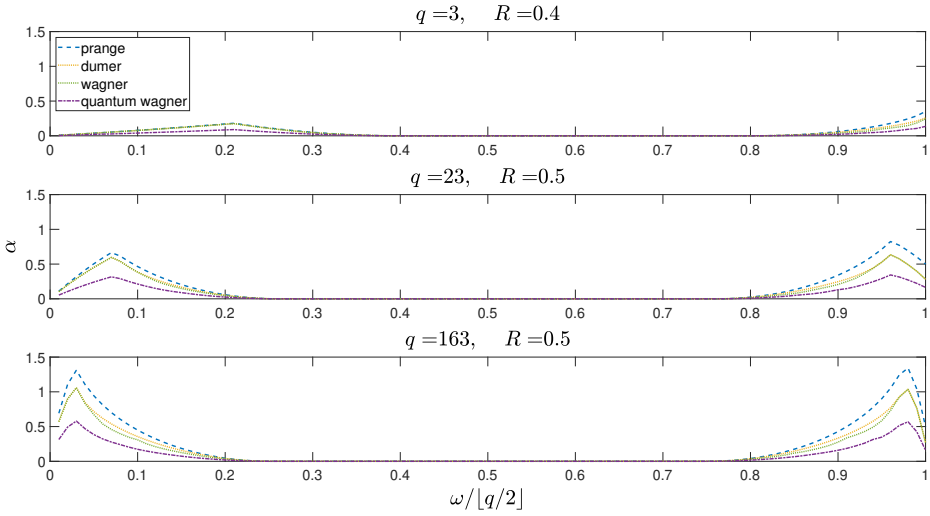

Fig. 4. Hardness of the Lee SD problem: The exponents α of the binary asymptotic complexity, $Time = 2^{\alpha n}$, of four ISD algorithms in Lee weight setting.

We observe that for any fixed q and R, the asymptotic complexity of our algorithms, as a function of ω^*, has 2 local maxima: at some values $\omega^*_- \in [0, x)$ and $\omega^*_+ \in (x, \lfloor \frac{q}{2} \rfloor]$, with $x = \frac{q^2-1}{4q}$ [(14)]. Moreover, these local maxima always satisfy:

$$\omega^*_- = \omega \in [0, x) \text{ st. } s_\omega = (1 - R).^{(15)}$$

$$\omega^*_+ = \begin{cases} \omega \in (x, \lfloor \frac{q}{2} \rfloor] \text{ st. } s_\omega = (1-R)^{(15)} \text{ if such an } \omega \text{ exists} \\ \lfloor \frac{q}{2} \rfloor \text{ otherwise.} \end{cases}$$

This characterization of the local maxima is particularly useful when aiming to obtain the hardest instances of a problem. Namely, for a fixed q, it allows us to find the R that yields the hardest problem and then to check only the 2 corresponding weights, ω_- and ω_+, to obtain the hardest instance. That makes our calculations more efficient, which becomes increasingly important as q increases and the convex optimization part of the calculations becomes costly due to the number of constraints in Problem 3.

[(14)] This value corresponds to the average Lee weight of a vector chosen uniformly at random.

[(15)] In that case, we have $S^n_w = q^{n-k}$, which is the case where we have on average 1 solution to the SD problem on random inputs $H, \mathbf{s}$.

It is also important to notice here that many previous papers only consider the case ω^*_- and miss out on very interesting parameter ranges where, for the lower values of q, the problem is typically the hardest. Nevertheless, we also observe that as q increases, the plots become symmetric between small weight and large weight. Therefore, we can expect that for relatively high values of q the difference between the small and large weights would become negligible. However, we cannot verify this claim due to the high computational cost of such verification.

The properties we observe here hold for all ISD algorithms we consider, in both classical and quantum settings. However, it is worth noticing that while these seem to be a generic property of ISD algorithms, there might be other algorithms for which these properties do not hold.

Parameters for which the problem is the hardest.

To find the hardest instances of the problem, for a given q, we rely on the observation about the local maxima, ω^*_- and ω^*_+, and we optimize over R to obtain the hardest instance. For the sake of simplicity, in Table 1, we present only the results of the analysis of the classical and quantum Wagner's based ISD algorithms and remark that the other two ISD algorithms exhibit similar behaviour.

q	Classical Wagner ISD complexity				Quantum Wagner ISD complexity			
	R	$\omega/\lfloor q/2 \rfloor$	α	$\hat{\alpha}$	R	$\omega/\lfloor q/2 \rfloor$	α	$\hat{\alpha}$
3	0.370	1.000	0.269	0.170	0.369	1.000	0.148	0.093
5	0.572	1.000	0.357	0.154	0.569	1.000	0.206	0.089
13	0.480	0.957	0.522	0.141	0.501	0.962	0.283	0.076
43	0.454	0.954	0.794	0.146	0.472	0.959	0.429	0.079
163	0.442	0.967	1.117	0.152	0.464	0.971	0.607	0.083
331	0.438	0.974	1.291	0.154	0.464	0.978	0.703	0.084

Table 1. Hardest instances of Lee SD problem: the asymptotic complexity exponents, α and $\hat{\alpha}$, correspond to the binary asymptotic complexity, $Time = 2^{\alpha n}$, and q-ary asymptotic complexity, $Time = q^{\hat{\alpha} n}$, respectively.

It can be readily verified that the complexity of a problem, expressed as $2^{n(\alpha+o(1))}$, becomes higher as q increases. That is expected since the inputs' size also increases, and we do not get this extra difficulty for free. If, for example, we want to use this problem in Stern's signature scheme, where the signature size essentially scales with the size of q-ary vectors of size n or $n-k$, this increase

of the input size becomes relevant. Therefore, we propose the scaling where the complexity is of the form $q^{n(\hat{\alpha}+o(1))}$ instead of $2^{n(\alpha+o(1))}$, and we refer to them as q-ary asymptotic complexity and binary asymptotic complexity, respectively. Observing q-ary complexity, the problem now is the hardest for $q = 3$. Intricately, q-ary complexity diminishes and then increases again at some point as q increases. Hence, it would be interesting to calculate the asymptotic q-ary complexity when both q and n grows beyond bounds. We can also observe that while for $q = 3$ and $q = 5$ the optimal values were for $\omega^* = 1$, this property does not hold for larger q. Nevertheless, it remains in the range close to 1 (typically, in the range $(0.95, 1]$). We can see, as well, that the hardest instances of the problem occur at the mid-range code rates and, typically, in the range $(0.35, 0.6)$.

8 Conclusion

This paper analyzes different ISD algorithms, both in the classical and quantum regimes, for solving SD problems with varying sizes of alphabet and different weight functions. In the numerical part of the paper, we focused on analyzing the Hamming and Lee weight cases as representative examples of weight functions.

Our results show that, for a fixed alphabet size $q > 3$, the complexity of the hardest instances of SD problem is higher in the Lee than in the Hamming weight, as well as that the hardest instances occur at high weights. That is true both in the classical and quantum setting. We also show that the problem remains exponentially hard for conveniently chosen parameters both in the classical and quantum setting for the class of the algorithms we consider. Finally, for a fixed alphabet size, we offer a rough estimate of the parameters' ranges for which the SD problem in Lee weight is typically the hardest.

These results have several implications for designers that want classical and quantum security estimates for their code-based schemes using different weight functions as, for example, for WAVE or other recently proposed schemes[BBC+20a]. For the quantum setting, our algorithms have almost a quadratic improvement over the classical setting, so it is important to update the parameters if we want to achieve quantum security.

Acknowledgments. The authors want to thank Nicolas Sendrier and Anthony Leverrier for helpful discussions. S.E. has received funding from the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No 754362.

References

[ApS21] MOSEK ApS. *MOSEK Fusion API for C++. Version Release 9.2.38.*, 2021.

[Ast84] Jaakko Astola. On the asymptotic behaviour of lee-codes. *Discret. Appl. Math.*, 8(1):13–23, 1984.

[Bar97] Alexander Barg. Complexity issues in coding theory. *Electronic Colloquium on Computational Complexity*, October 1997.

[BB19] S. Bhattacharya and A. Banerjee. A method to find the volume of a sphere in the lee metric, and its applications. In *2019 IEEE International Symposium on Information Theory (ISIT)*, pages 872–876, 2019.

[BBB+20] Magali Bardet, Pierre Briaud, Maxime Bros, Philippe Gaborit, Vincent Neiger, Olivier Ruatta, and Jean-Pierre Tillich. An algebraic attack on rank metric code-based cryptosystems. In *Advances in Cryptology - EUROCRYPT 2020*, volume 12107, pages 64–93. Springer, 2020.

[BBC+20a] Marco Baldi, Massimo Battaglioni, Franco Chiaraluce, Anna-Lena Horlemann-Trautmann, Edoardo Persichetti, Paolo Santini, and Violetta Weger. A new path to code-based signatures via identification schemes with restricted errors. *CoRR*, 2020.

[BBC+20b] Magali Bardet, Maxime Bros, Daniel Cabarcas, Philippe Gaborit, Ray A. Perlner, Daniel Smith-Tone, Jean-Pierre Tillich, and Javier A. Verbel. Improvements of algebraic attacks for solving the rank decoding and minrank problems. In *Advances in Cryptology - ASIACRYPT 2020*, volume 12491, pages 507–536. Springer, 2020.

[BCDL19] Rémi Bricout, André Chailloux, Thomas Debris-Alazard, and Matthieu Lequesne. Ternary syndrome decoding with large weights. *SAC 2019*, 2019.

[BH97] Gilles Brassard and Peter Høyer. An exact quantum polynomial-time algorithm for simon's problem. In *Fifth Israel Symposium on Theory of Computing and Systems, ISTCS 1997, Ramat-Gan, Israel, June 17-19, 1997, Proceedings*, pages 12–23. IEEE Computer Society, 1997.

[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.

[BV14] Stephen P. Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2014.

[Coo00] Colin Cooper. On the distribution of rank of a random matrix over a finite field. *Random Struct. Algorithms*, 17:197–212, 10 2000.

[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 *Advances in Cryptology - ASIACRYPT 2019*, LNCS, 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.

[Gro96] Lov K. Grover. A fast quantum mechanical algorithm for database search. In Gary L. Miller, editor, *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 212–219. ACM, 1996.

[GS91] Danièle Gardy and Patrick Solé. Saddle point techniques in asymptotic coding theory. In *Algebraic Coding, First French-Soviet Workshop,*, volume 573, pages 75–81. Springer, 1991.

[KT17] Ghazal Kachigar and Jean-Pierre Tillich. Quantum information set decoding algorithms. In *Post-Quantum Cryptography 2017*, volume 10346 of *LNCS*, Utrecht, The Netherlands, June 2017. Springer.

[MMT11] Alexander May, Alexander Meurer, and Enrico Thomae. Decoding random linear codes in $O(2^{0.054n})$. In *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.

[NC00] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[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.

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

[WKH+21] Violetta Weger, Karan Khathuria, Anna-Lena Horlemann, Massimo Battaglioni, Paolo Santini, and Edoardo Persichetti. On the hardness of the lee syndrome decoding problem. 2021. arXiv quant-ph 2002.12785.