Not Just Regular Decoding: Asymptotics and Improvements of Regular Syndrome Decoding Attacks

Andre Esser[1*] and Paolo Santini[2]

[1] Technology Innovation Institute, UAE
andre.esser@tii.ae
[2] Marche Polytechnic University, Italy
p.santini@univpm.it

Abstract. Cryptographic constructions often base security on structured problem variants to enhance efficiency or to enable advanced functionalities. This led to the introduction of the Regular Syndrome Decoding (RSD) problem, which guarantees that a solution to the Syndrome Decoding (SD) problem follows a particular block-wise structure. Despite recent attacks exploiting that structure by Briaud and Øygarden (Eurocrypt '23) and Carozza, Couteau and Joux (CCJ, Eurocrypt '23), many questions about the impact of the regular structure on the problem hardness remain open.
In this work we initiate a systematic study of the hardness of the RSD problem starting from its asymptotics. We classify different parameter regimes revealing large regimes for which RSD instances are solvable in polynomial time and on the other hand regimes that lead to particularly hard instances. Against previous perceptions, we show that a classification solely based on the uniqueness of the solution is not sufficient for isolating the worst case parameters. Further, we provide an in-depth comparison between SD and RSD in terms of reducibility and computational complexity, identifying regimes in which RSD instances are actually *harder* to solve.
We provide the first asymptotic analyses of the algorithms presented by CCJ, establishing their worst case decoding complexities as $2^{0.141n}$ and $2^{0.135n}$, respectively. We then introduce *regular-ISD* algorithms by showing how to tailor the whole machinery of advanced Information Set Decoding (ISD) techniques from attacking SD to the RSD setting. The fastest regular-ISD algorithm improves the worst case decoding complexity significantly to $2^{0.112n}$. Eventually, we show that also with respect to suggested parameters regular-ISD outperforms previous approaches in most cases, reducing security levels by up to 30 bits.

Keywords: Hardness Classification, Information Set Decoding, Code-Based Cryptography

1 Introduction

The Syndrome Decoding Problem (SDP) is one of the most fundamental problems in coding theory, and as such finds frequent applications as security foundation in cryptographic

* Supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project-ID MA 2536/12

constructions. Given the parity-check matrix $\mathbf{H}$ of a linear code, a vector (the syndrome) $\mathbf{s}$ and an integer w the SDP asks to recover a vector $\mathbf{e}$ of Hamming weight w satisfying $\mathbf{He} = \mathbf{s}$.

In recent years, to enable the design and increase the efficiency of (advanced) cryptographic constructions based on SDP, such as signatures [CCJ23a, CLY+24], efficient MPC [HOSS18], Vector Oblivious Linear Evaluation (VOLE) [BCGI18], Pseudorandom Correlation Generators (PCGs) [BCG+19b, BCG+20] or correlated Oblivious Transfer (OT) [BCG+19a, YWL+20], often a structured version of the problem is considered, known as Regular Syndrome Decoding (RSD) problem. Initially introduced in the context of the FSB hash function [AFS05], in this variant the error vector $\mathbf{e}$ is known to be regular, i.e., it consists of w consecutive, equally sized chunks, each of weight exactly one.

Intuition suggests that the introduction of such regular structure decreases the problem hardness, but this might not hold universally true. For example, in the related LPN (or LWE) setting [HKL+12, LPR10], even after years of study, attacks on structured ring-variants are essentially the same as on their non-structured counterparts. A first attempt at the translation of concepts used in Information Set Decoding (ISD) attacks against SD to the regular case was given in the security analysis of [HOSS18]. However, the authors eventually concluded that those attacks even if tailored to the RSD setting obtain about the same complexity as direct SD attacks. Generally, the security analysis of RSD-based constructions has predominantly been performed in an ad-hoc manner and supported the assumption that the most efficient attacks on RSD remain the same as those on SD. A more focused study was then performed in [LWYY22], which, however, also finds standard ISD attacks to perform best for most of the suggested parameters.

Recently then, Briaud and Øygarden [BØ23] showed that the regular structure allows to model the problem as a quadratic system of equations, similar to a modelling used by Arora and Ge [AG11] in the context of the structured LPN assumption. Briaud and Øygarden further show that the application of algebraic solvers to the resulting system yields for many of the parameters found in the literature the best running times. At the same time Carozza, Couteau and Joux (CCJ) [CCJ23a] designed a new signature scheme exploiting the regular structure of the RSD solution to obtain reduced signature sizes. In the full version of their work [CCJ23b] the authors give a first hardness classification of the RSDP in comparison to the SDP: based on the uniqueness of the solution the authors identify three regimes in which they find either the SDP or the RSDP to be harder, while in the third regime both problems seem incomparable. Additionally, the authors present three new algorithms for solving the RSD problem. The first algorithm is based on a linearization technique initially introduced by Saarinen [Saa07] for attacking the FSB hash function family. Their second approach relies on enumerating the searched error vector in a meet-in-the-middle fashion, while the third algorithm is an extension of the enumeration procedure by a nearest-neighbor procedure. The authors find that this last extension yields the best running times and, therefore, use it for parameter selection. Note that in the analysis of this third algorithm an optimistic runtime assumption for nearest-neighbor search with structured differences is used. Since no constructive algorithm is known that achieves this running time, this third algorithm, while resulting in conservative parameters, remains non-constructive.

Those recent works made big steps towards classifying the hardness of RSD by showing that algorithms can indeed be tailored to exploit the regular structure. However, especially from a theoretical perspective the effect of the regular structure on the problem hardness remains largely unclear. Naturally, there arise questions about the equivalence of both problems, which were briefly touched in [CCJ23b], or about the amount of structure that can be induced until the problem hardness collapses. Further, in terms of computational hardness, Information Set Decoding algorithms are usually the best strategy to solve the SDP. However, so far it is not known to which extend those can be tailored to exploit the regular structure in case of RSD.

Our Contribution In this work we initiate the systematic study of the hardness of the regular syndrome decoding problem. We start this investigation from a theoretical viewpoint that allows us to isolate instances for which RSD is hard and for which its hardness collapses. This first asymptotic treatment then forms the foundation for a more comprehensive comparison between the hardness of SD and RSD for different parameter regimes. In the second part of this work we then show that against previous perceptions [HOSS18, CCJ23b] essentially all advanced techniques from ISD algorithms in the non-regular case, can be tailored to the RSD setting resulting in new *regular-ISD* algorithms, outperforming the state-of-the-art in the asymptotics as well as in concrete numbers.

Towards a rigorous hardness classification. We first present our results on the general hardness of the RSD problem. In particular, we show that there exist large regimes of parameters for which RSD instances are solvable in polynomial time. One of these regimes is a direct consequence of the introduced structure: the known block-wise structure of the solution gives rise to additional linear equations decreasing the dimension of the underlying code. Once the dimension is small enough, one can solve for $\mathbf{e}$ in polynomial time via basic linear algebra. Note that this is in direct contrast to the SD problem which, for constant rates, is known to be exponentially hard for any parameters that yield a unique solution, i.e., parameters below the Gilbert-Varshamov (GV) bound.

This motivates the question for which parameters RSD remains hard and, in particular, for which parameters it reaches the worst case. Carozza et al. [CCJ23b] identified an *RSD uniqueness-bound* similar to the GV-bound in the SD case and gave a classification of RSD's hardness based on this bound. In this context we show that the uniqueness bound alone unlike the GV-bound in the SD case, does not allow to classify the hardest instances in the RSD case. We then study how to isolate worst case instances and are able to identify a regime of parameters which includes the worst case instances, with respect to the proposed *regular-ISD* algorithms.

Furthermore, in our classification we identify large regimes in which RSD instances are actually *harder* to solve than SD instances with respect to the best algorithms. This is in contrast to the intuitive perception that exploiting the regular structure by switching from SD to RSD instances in cryptographic constructions inevitably leads to a decreased security level. Moreover, when comparing the complexity of the best algorithms on SD and RSD for their respective worst case parameters, we find that RSD is harder for all code rates larger than (about) $\frac{1}{2}$ and that RSD instances obtain a far larger absolute complexity. This effect is illustrated in Fig. 1 in which we compare regular-ISD performance on RSD worst case instances against ISD performance on SD worst case instances.

Eventually, we show that the complexity of SD and RSD with respect to the best algorithms converges for small w. More, precisely as soon as w is sublinear in the code length the complexity for solving both problems differs only by second order terms.

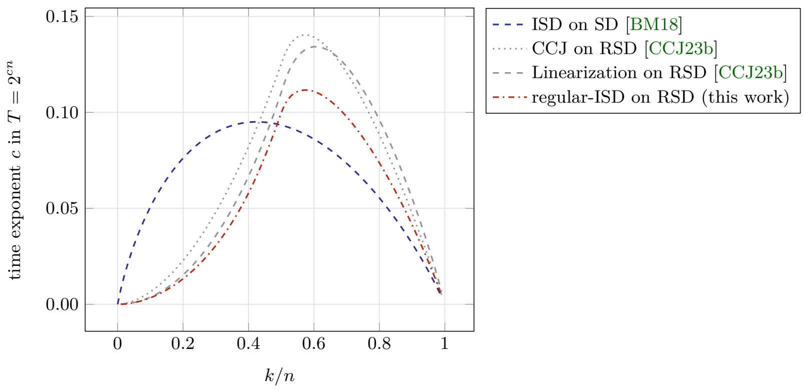


Fig. 1: Comparison of running time of best ISD algorithm on *SD* worst case instances and best regular-ISD algorithm (Section 4.4) and CCJ's algorithms on *RSD* worst case instances.

The regular-ISD approach. The second main contribution of this work is the design of *regular-ISD* algorithms. In all algorithms we first encode the regularity into the parity-check matrix, which essentially results in w additional rows. These linear equations have also been exploited in the algebraic attack by Briaud and Øygarden [BØ23] and in the enumeration routine by Carozza, Couteau and Joux [CCJ23b]. However, we show that the particular structure of these equations requires additional care in the analysis and, actually, prevents the use of some of those equations within the CCJ enumeration algorithm.

We provide the first asymptotic analysis of the CCJ algorithms which establishes the asymptotic baseline for improvements on solving the RSD problem on worst case instances with a complexity of $2^{0.135n}$ (linearization) and $2^{0.141n}$ (enumeration) respectively. We are then able to drastically improve this initial complexity to $2^{0.112n}$ by tailoring the whole machinery of ISD techniques to the regular setting. A comparison between the complexity of the algorithms presented in [CCJ23b] and the regular-ISD approach for all rates is illustrated in Fig. 1.

ISD algorithms usually apply a permutation to the columns of the parity-check matrix to obtain a permuted instance $\mathbf{HP}$ with solution $\mathbf{P}^{-1}\mathbf{e}$ and then hope for a specific weight distribution on $\mathbf{P}^{-1}\mathbf{e}$. However, as observed in [CCJ23b] random permutations destroy the regular structure on the solution and might, hence, prevent further exploitation. We therefore restrict to a specific set of *regular permutations*, similar to a technique described in [HOSS18], which exploits the regular structure to enhance the success probability of obtaining the desired weight distribution. Moreover, any such regular permutation maintains a certain regular structure on the permuted solution, which we then exploit in order to improve the commonly applied enumeration subroutine of ISD procedures. We show that the use of regular permutations enables to leverage all linear equations encoding the regularity of the solution, regardless of their specific structure. Eventually, we show how to incorporate the most advanced concepts of *representations* [MMT11, BJMM12] and nearest-neighbor search [MO15, BM18] into regular-ISD procedures.

We also provide a concrete cost analysis of the regular-ISD techniques as well as the CCJ algorithm and evaluate their complexity on parameter sets provided in the literature.

source	(n, k, w)	previous best	regular-ISD
[HOSS18]	$(1280, 860, 80)$	132	114
[LWYY22]	$(2^{10}, 652, 57)$	90	76
	$(2^{10}, 652, 106)$	129	113
	$(2^{12}, 1589, 172)$	132	109
	$(2^{14}, 3482, 338)$	135	118
	$(2^{16}, 7391, 667)$	139	126
[CCJ23a]	$(1842, 825, 307)$	183	153

Table 1: Bit security for selected instances considering regular-ISD in comparison to previous best approaches.

In Table 1 we give the complexity of the different approaches on selected parameter sets. As can be observed regular-ISD obtains significant bit-security reductions of up-to 30 bits, showing that most of the depicted sets fall below the targeted security threshold. However, generally, we find that constructions based on RSD followed a (over-) conservative parameter selection procedure, such that even considering regular-ISD, most parameter sets still obtain comfortable margins (of up to 270 bits).

Outline. In Section 2 we cover necessary basics on coding theory, the SD and the RSD problem. Subsequently, in Section 3 we provide a systematic classification of the hardness of RSD, in particular in comparison to its non-regular counterpart. In Section 4 we present the regular-ISD algorithms and provide their asymptotic analysis. At the end of that section we also give the analysis of the CCJ algorithm and an asymptotic comparison to previous approaches. Eventually, in Section 5, we present results on the performance of regular-ISD algorithms on suggested parameters.

Artifacts. We provide all used source code, including proof of concept implementations of the regular-ISD algorithms, numerical optimizations for asymptotic exponents as well as implementations of the concrete complexity formulas in the git repository `github.com/Memphisd/Regular-ISD`.

2 Preliminaries

We denote by $\mathbb{F}_2$ the binary finite field. Vectors (resp., matrices) are indicated with bold lowercase (resp., uppercase) letters. Vectors are viewed interchangeably as rows and columns. The Hamming weight of a vector $\mathbf{x}$ corresponds to the number of its non-null entries and is denoted $|\mathbf{x}|$. For two vectors $\mathbf{x}$ and $\mathbf{y}$, we denote their scalar product as $\langle \mathbf{x}; \mathbf{y} \rangle$. For a set of coordinates I we denote by $\mathbf{x}_I$ the projection of $\mathbf{x}$ on the coordinates indexed by I. For an integer a we let $[a] := \{1, \ldots, a\}$. We say an event occurs with *high probability* $p(n)$ if $p(n)$ approaches 1 for $n \to \infty$.

By a linear code $\mathscr{C} \subseteq \mathbb{F}_2^n$, we refer to a linear subspace of the ambient space $\mathbb{F}_2^n$. Typical code parameters are the *length* n, the *dimension* $k = \kappa n$, for $\kappa \in [0; 1]$ being the *code* *rate*, and the *co-dimension* $n - k = (1 - \kappa)n$ (also called redundancy). Any linear code can be compactly represented by a *parity-check matrix*, that is, a full rank $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$ which serves as a basis for the null space of the code: a vector $\mathbf{c} \in \mathbb{F}_2^n$ is a codeword if and only if $\mathbf{Hc} = \mathbf{0}$. Linear codes are invariant under changes of basis, i.e., $\mathbf{H}$ and $\mathbf{SH}$, with $\mathbf{S} \in \mathbb{F}_2^{(n-k)\times(n-k)}$ being non singular, are parity-check matrices for the same code.

We say that a set $J \subseteq \{1, \cdots, n\}$ of size k is an *information set* for a code $\mathscr{C}$ if any two distinct codewords $\mathbf{c}, \mathbf{c}' \in \mathscr{C}$ are different in at least one of the coordinates indexed by J. This is equivalent to the columns of $\mathbf{H}$ which are not indexed by J forming a full rank $(n-k) \times (n-k)$ matrix.

For any vector $\mathbf{e} \in \mathbb{F}_2^n$ which is not a codeword, we have $\mathbf{He} = \mathbf{s} \neq \mathbf{0}$. The vector $\mathbf{s} \in \mathbb{F}_2^{n-k}$ is called a *syndrome*. The same syndrome correspond to multiple vectors: for any non-null $\mathbf{c} \in \mathscr{C}$, $\mathbf{e}$ and $\mathbf{e} + \mathbf{c}$ have the same syndrome. Decoding a given syndrome into an arbitrary vector in $\mathbb{F}_2^n$ can be done by basic linear algebra. However, the problem becomes hard if the vector is required to have low Hamming weight, leading to the Syndrome Decoding Problem (SDP).

Definition 2.1 (Syndrome Decoding (SD)). *Let $k, n, w \in \mathbb{N}$ such that $k, w \leq n$. Given $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$ and $\mathbf{s} \in \mathbb{F}_2^{n-k}$, find a vector $\mathbf{e} \in \mathbb{F}_2^n$ of Hamming weight w such that $\mathbf{He} = \mathbf{s}$. We refer to the SDP problem with parameters (n, k, w) as $\mathcal{SDP}(n, k, w)$.*

Motivated by cryptographic constructions we always assume the existence of at least one solution. Therefore the amount of existing solutions is derived as the maximum of 1 and the expected amount of solutions, giving

$$S = \max\left(1, \frac{\text{Number of weight-}w\text{ vectors}}{\text{Number of syndromes}}\right) = \max\left(1, \binom{n}{w} 2^{-(n-k)}\right).$$

The best known solvers for $\mathcal{SDP}(n, k, w)$ are ISD algorithms. The first algorithm of this class is due to Prange [Pra62] and finds a solutions to the $\mathcal{SDP}(n, k, w)$ in time

$$T = \tilde{\mathcal{O}}\left(\frac{\binom{n}{w}}{S \cdot \binom{n-k}{w}}\right). \quad (1)$$

The hardest instances of SDP are those where w is chosen as the maximum value which still leads to a unique solution. This is the case if w is equal to the Gilbert-Varshamov (GV) bound, which asymptotically gives $w = \omega n$, where $\omega = h^{-1}(1 - \kappa)$ and $h(x) := -x\log_2(x) - (1 - x)\log_2(1 - x)$ is the binary entropy function.

Regular Syndrome Decoding. For a vector $\mathbf{e} \in \mathbb{F}_2^n$ we say it is *b-regular of weight $w \leq \frac{n}{b}$* if it can be written as $\mathbf{e} = \left(\mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e}_{\frac{n}{b}}\right)$, where each $\mathbf{e}_i$ has length b and Hamming weight *at most* one. Furthermore, we simply call a weight-w vector $\mathbf{e}$ regular if $b = \frac{n}{w}$, i.e., each $\mathbf{e}_i$ is of Hamming weight *exactly* one.

Requiring that the solution to SDP is regular, one obtains the Regular Syndrome Decoding Problem (RSDP).

Definition 2.2 (Regular Syndrome Decoding (RSD)). *Let $k, n, w \in \mathbb{N}$ such that $k, w \leq \frac{n}{2}$. Given $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$ and $\mathbf{s} \in \mathbb{F}_2^{n-k}$, find a* regular *vector $\mathbf{e} \in \mathbb{F}_2^n$ of Hamming weight w such that $\mathbf{He} = \mathbf{s}$. We refer to the RSD problem with parameters (n, k, w) as $\mathcal{RSD}(n, k, w)$.*

The constraint $w \leq \frac{n}{2}$ is implied by the fact that the solution has to be regular (equivalently, it must be $b \geq 2$ hence $w = \frac{n}{b} \leq \frac{n}{2}$).

3 Hardness Classification

In this section we present results on the general hardness of the RSD problem and draw direct comparison to the SD problem. We start by recalling and formalizing the required conditions to expect uniqueness of the solution in the RSD case.

3.1 Uniqueness bound

The uniqueness bound for RSD marks, analogous to the GV bound in the SD case, the transition to instances with multiple solutions. Any set of parameters satisfying this bound is expected to have at most one solution. The bound reads

$$\left(\frac{n}{w}\right)^w \leq 2^{n-k},$$

where the left side specifies the amount of regular vectors of weight w, i.e., the search space size, while the probability that a random element of the search space satisfies all parity equations is $2^{-(n-k)}$, leading to the term on the right.

Note that since $k = \kappa n$, $w = \omega n$ and, hence, $n/w = 1/\omega$, the above bound can be rewritten as $-\omega \log_2(\omega) \leq 1 - \kappa$. Therefore, once κ is known, one can determine the maximum value of ω for which uniqueness of the RSD solution (statistically) holds.

Observe that $-\omega \log_2(\omega)$ reaches its maximum at $\omega = e^{-1}$ with a maximum value of $(e \cdot \ln 2)^{-1} \approx 0.5307$. This implies that for any code rate $\kappa \leq 1 - (e \cdot \ln 2)^{-1} \approx 0.4693$, the inequality is satisfied for any value of ω. We summarize this in the following proposition.

Proposition 3.1 (Uniqueness bound for RSD). *Let $k = \kappa n$, $w = \omega n$. We expect a unique solution to $\mathcal{RSD}(n, k, w)$ if*

$$-\omega \log_2(\omega) \leq 1 - \kappa.$$

For $\kappa < 1 - (e \cdot \ln 2)^{-1}$, the inequality is satisfied for any choice of $0 < \omega \leq 0.5$.

The fact that for sufficiently small code rate, uniqueness is expected regardless of ω is a remarkable difference to the GV bound in the SD case. Later we show that, also with respect to the hardness classification of instances, the RSD uniqueness bound and the GV bound in the SD case behave differently. Indeed, while the GV bound classifies the hardest instances, at least with respect to known algorithms, worst RSD parameters do not always match the uniqueness bound.

3.2 Instances Solvable in Polynomial Time

In this section, we identify two large regimes of parameters for which RSD is solvable in polynomial time. One of those regimes is, analogous to the SD case, related to a large amount of existing solutions; instead, the other regime is specific to the regular structure of the RSD problem. Let us start with the RSD specific regime.

Encoding Regularity Note that a solution $\mathbf{e}$ to the RSD problem is a concatenation of w unit vectors of length $b := n/w$. Therefore, for each $i \in \{1, \cdots, w\}$, we can include the parity-check equation $\langle \mathbf{h}_i, \mathbf{e} \rangle = 1$ where

$$\mathbf{h}_i = \big(\underbrace{0, 0, \cdots, 0}_{(i-1)b}, \underbrace{1, 1, \cdots, 1}_{b}, \underbrace{0, 0, \cdots, 0}_{(w-i)b}\big).$$

Considering all these additional equations, we obtain a new parity-check matrix $\mathbf{H}'$ with $n - k + w$ rows and n columns, structured as

$$\mathbf{H}' = \left(\begin{array}{c|c|c|c} \multicolumn{4}{c}{\mathbf{H}} \\ \hline 1\,1 \cdots 1 & & & \\ & 1\,1 \cdots 1 & & \\ & & \ddots & \\ & & & 1\,1 \cdots 1 \\ \underbrace{\hphantom{1\,1 \cdots 1}}_{b} & \underbrace{\hphantom{1\,1 \cdots 1}}_{b} & & \underbrace{\hphantom{1\,1 \cdots 1}}_{b} \end{array}\right) \Big\updownarrow w \tag{2}$$

Analogously, we update the syndrome as $\mathbf{s}' = (\mathbf{s}, 1, 1, \cdots, 1) \in \mathbb{F}_2^{n-k+w}$. This results in a new RSD instance $(\mathbf{H}', \mathbf{s}')$ where the parity-check matrix $\mathbf{H}'$ corresponds to a code of smaller dimension $k' = k - w$ (so, smaller rate $\kappa' = \kappa - w/n$), while maintaining the same co-dimension $n - k$ and same solution $\mathbf{e}$, i.e., it still holds that $\mathbf{H}'\mathbf{e} = \mathbf{s}'$.

This encoding of the regular structure leads to a polynomial regime whenever the new parity check matrix $\mathbf{H}'$ contains more than n rows.

Theorem 3.1. *Whenever $w \geq k$, $\mathcal{RSD}(n, k, w)$ is solvable in polynomial time with high probability.*

Proof. After encoding the regularity, the RSD solution is a solution to a linear system containing $n - k + w$ equations and n unknowns, represented by the parity check matrix and the syndrome $(\mathbf{H}', \mathbf{s}')$. Whenever $n - k + w \geq n$, which is equivalent to $w \geq k$, the system contains more equations than unknowns.

Note that for random $\mathbf{H}'$ such a system is solvable in polynomial time with high probability. Therefore, consider the case of $w = k$, i.e., $\mathbf{H}'$ being a square matrix. The corresponding system is solvable in polynomial time whenever $\mathbf{H}'$ has rank at least $n - \varepsilon$ with $\varepsilon = \mathcal{O}(\log n)$. Fulman and Goldstein [FG15, Eq. (1)] have shown that this happens with high probability.

The statement of the theorem follows by observing that for our precise choice of $\mathbf{H}'$ this probability is even higher. Indeed, we have $\mathbf{H}' = \begin{pmatrix} \mathbf{H} \\ \mathbf{B} \end{pmatrix}$, where the $w \times n$ matrix $\mathbf{B}$ contains the extra parity-check equations and $\mathbf{H}$ is a random $(n - k) \times n$ matrix. Note that $\mathbf{H}'$ is not of full rank if either $\mathbf{B}$ or $\mathbf{H}$ contain linearly dependent rows or if the space generated by the rows of $\mathbf{H}$ intersects with the space generated by the rows of $\mathbf{B}$. However, for our choice of $\mathbf{H}'$, unlike the random case, all rows of $\mathbf{B}$ are linearly independent by construction. $\square$

Remark 3.1 (Full Rank $\mathbf{H}'$). In case when $\mathbf{H}'$ is square, asymptotically, for the considered case of $w = \Theta(n)$, the matrix behaves as a random matrix with respect to invertibility. Indeed, the additional w parity-check equations are linearly independent, i.e., they generate a space $\mathscr{B}$ of dimension w. So, $\mathbf{H}'$ has full rank whenever the rows of $\mathbf{H}$ have full rank $n - k$

and generate a space which intersects trivially with $\mathscr{B}$. This happens with probability

$$\left(\frac{2^n - 2^w}{2^n}\right)\left(\frac{2^n - 2^{w+1}}{2^n}\right)\cdots\left(\frac{2^n - 2^{w+n-k-1}}{2^n}\right) = \prod_{i=1}^{n-w} 1 - 2^{-i} \xrightarrow{n\to\infty} 0.2887,$$

which converges to the same limit as for random $\mathbf{H}'$.

Amount of Solutions Intuitively and similar to the SD case, RSD becomes easy whenever there are too many solutions. We specify this regime of parameters in the following theorem.

Theorem 3.2. *Whenever* $\frac{k}{n} \geq \frac{1}{2}$ *and* $w > n-k$, $\mathcal{RSD}(n,k,w)$ *with* $w \mid k$ *is solvable in expected polynomial time.*

Proof. Note that the existing amount of solutions to a random RSD instance is $S = \max\left(1, \frac{\left(\frac{n}{w}\right)^w}{2^{n-k}}\right)$, as we assume a guaranteed solution. Further, the later introduced Theorem 4.1 states the expected running time for solving $\mathcal{RSD}(n,k,w)$ with $w \mid k$ (up to polynomial factors) as

$$T = \frac{\left(1 - \frac{k-w}{n}\right)^{-w}}{S}.$$

Therefore as long as $T \leq 1$ which is equivalent to

$$\left(1 - \frac{k-w}{n}\right)^w \geq \frac{2^{n-k}}{\left(\frac{n}{w}\right)^w}, \tag{3}$$

RSD is solvable in expected polynomial time. Using $w \geq n-k$ we lower bound the left hand side of Eq. (3) as

$$\left(1 - \frac{k-w}{n}\right)^w \geq \left(1 - \frac{2k-n}{n}\right)^{n-k} = \left(2(1-\frac{k}{n})\right)^{n-k}, \tag{4}$$

as long as $k \geq w$, which follows from $k \geq \frac{n}{2}$ (since $w \leq n/2$), which is equivalent to $\frac{k}{n} \geq \frac{1}{2}$ as stated in the theorem. Now using $w \geq n-k$, we upper bound the right hand side of Eq. (3) as

$$\frac{2^{n-k}}{\left(\frac{n}{w}\right)^w} \leq \frac{2^{n-k}}{\left(\frac{n}{n-k}\right)^{n-k}} = \left(2(1-\frac{k}{n})\right)^{n-k}. \tag{5}$$

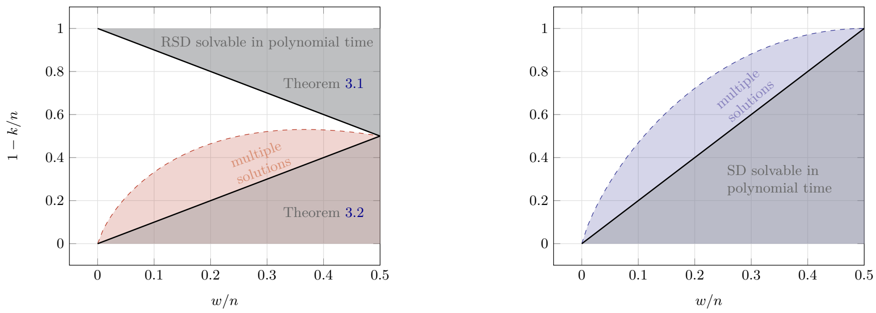
Finally, observe that (4)≥(5) , which is trivially fulfilled, implies Eq. (3). □

Remark 3.2 (Rounding Issues). Note that the requirement of $w \mid k$ stems from the fact that Theorem 4.1 restricts to this case in order to obtain a simplified runtime formula. In Section 4.6, we derive the runtime formula for all combinations of (w,k) and show that in most cases the simplified formula yields a valid upper bound, implying that Theorem 3.2 also holds for those cases.

Comparison to the Non-Regular Case In the case of SD instances over $\mathbb{F}_2$ it is well known that those are solvable in polynomial time whenever

$$\frac{n-k}{2} \leq w \leq \frac{n+k}{2}.$$

The corresponding algorithm simply solves the system restricted to the first $(n-k) \times (n-k)$ submatrix of $\mathbf{H}$. This solution has expected weight $(n-k)/2$. Note that arbitrary columns from the last k columns of $\mathbf{H}$ can be added to the syndrome prior to solving the system in order to increase the solution weight up to a maximum of $(n-k)/2 + k = (n+k)/2$.



(a) RSD parameter regimes

(b) SD parameter regimes

Fig. 2: Parameter regimes for RSD and SD, where gray shaded area marks instances solvable in polynomial time, while colored shaded area marks instances where multiple solutions exist. Dashed lines depict uniqueness bound and GV bound respectively.

In Fig. 2 we compare the regimes of parameters for which instances are solvable in polynomial time in the RSD and the SD case. The figure visualizes that Theorem 3.2 relates to a similar regime as the polynomial instances regime in the SD case, enclosed completely in the region of parameters that give rise to many solutions. In contrast, Theorem 3.1 relates to the whole regime of parameters lying in the upper gray shaded triangle. Note that for all included instances the solution is expected to be unique. This shows that a hardness classification solely based on the uniqueness of the solution is not sufficient in the RSD case. Moreover, this raises the question about the worst case parameters in term of weight for RSD instances.

3.3 Worst Case Instances

In the following we answer the question for the worst case weight of RSD instances. That is, given a dimension $k = \kappa n$ of a linear code, which $w^* = \omega^* n$ leads to the hardest RSD instance $\mathcal{RSD}(n, k, w^*)$.

From the SD case it is known that the hardest instances are those matching the GV bound. However, for RSD we know from Proposition 3.1 that not for every choice of κ a corresponding ω matching the uniqueness bound exists.

Worst Case for SD Instances Note that there is no proof that SD instances matching the GV bound form indeed the worst case. However, there is an intuitive argument, which is further supported by all known algorithms reaching their worst case running time for those instances.

As long as the weight stays below the GV bound the solution is unique. Now, for increasing weight, the search space grows exponentially in n, rendering the problem more difficult. On the other hand if the weight exceeds the GV bound there exist exponentially many solutions, which makes the problem easier. Therefore it is assumed that the hardest instances are those whose weight matches the GV bound.

The Case of RSD In the RSD case the same argument fails, as even if the solution is unique, increasing ω gives rise to more equations encoding the regularity, which may render the problem easier.

In fact, we find that the worst case weight for the regular-ISD algorithms presented in Section 4 grows initially (almost) linear with the dimension as $\omega^* \approx c \cdot \kappa$, with $c \approx 1/2$. This trend continues until $c \cdot \kappa$ exceeds the uniqueness bound, from where ω^* is equal to the uniqueness bound. We visualize the worst case weight ω_{low} for the most basic regular-ISD algorithm in Fig. 3.

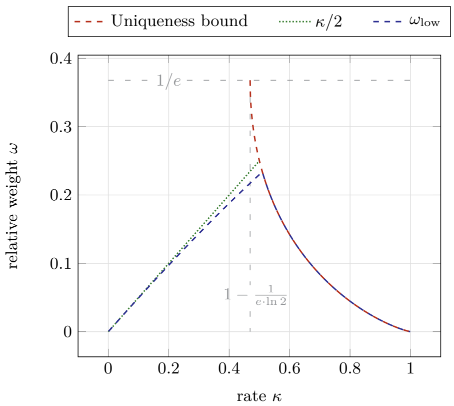


Fig. 3: Worst case weight ω_{low} of most basic regular-ISD algorithm (see Algorithm 1 and Theorem 4.1). From $\kappa \geq 0.51$ onwards ω_{low} matches the uniqueness bound.

Algorithm dependence. Moreover, we find that the precise weight that leads to the worst case depends on the considered algorithm. This behavior is similar to the *worst case rate* in the SD case, where the exact rate which maximizes the running time is algorithm dependent. Usually, it lies around 1/2, but seems to be decreasing for more advanced procedures. Most recent improvements reach their worst case running time for a rate as low as $\kappa^* = 0.42$ [BM18, CDMHT22, Ess23].

We observe a similar behavior for regular-ISD algorithms where the *worst case weight* seems to approach $\kappa/2$ for more advanced procedures in the regime $\kappa \leq 1/2$. The most advanced regular-ISD algorithms almost exactly reach $\omega^* = \kappa/2$ in that regime. Note that the translation of this behavior from the worst case *rate* in the SD case to the worst case *weight* in the RSD case is likely to be explained by the regularity encoding equations. Those set the rate in direct dependence to the weight as the new dimension after encoding regularity is $k' = k - w$.

Worst case weight. Let $UB(\kappa)$ denote the weight $\omega \leq e^{-1}$ matching the RSD uniqueness bound if such an ω exists. We find that the worst case relative weight ω^* for RSD instances for all considered algorithms satisfies

$$\omega_{\text{low}} \leq \omega^* \leq \min\left(\frac{\kappa}{2} + \varepsilon, UB(\kappa)\right),$$

for a very small constant ε. Remember that for $\kappa \geq 0.51$ it holds that $\omega_{\text{low}} = UB(\kappa)$, implying $\omega^* = UB(\kappa)$ (compare to Fig. 3). Generally, considering $\varepsilon = 0$ the right side of the inequality gives a well approximation of the worst case weight of all studied algorithms. This weight is also visualized in Fig. 3.

Note that this shows that even for the regime where there exist parameters matching the uniqueness bound, this bound does not suffice to classify the worst case weight. For example, for a rate of $\kappa = 0.47$, we find that the worst case weight is always smaller than $\omega^* \leq 0.235$ which is far from the uniqueness bound which would imply a weight parameter of roughly 0.349.

3.4 Hardness Comparison to the Non-Regular Case

In this section we compare the hardness of RSD and SD directly. The goal is to identify those parameter regimes in which SD instances are strictly harder to solve than RSD instances and vice versa, if such exist. Also for regimes that do not allow such a strict separation we investigate which problem is harder with respect to known algorithms. In this case we consider for RSD the regular-ISD algorithms presented in the subsequent Section 4.

Classification Based on the Amount of Solutions A first step towards a rigorous hardness classification of RSD was made recently in [CCJ23b], where the authors identify three regimes in dependence on the amount of existing solutions. For each regime the authors argue which problem is harder, identifying one regime in which SD is harder, one in which RSD is harder and one in which both problems are incomparable. However, a proper definition of *problem A being harder than problem B* is missing. For some regimes the authors argue about the existence of polynomial reductions while for others they argue about possible algorithmic speedups. In the following we recall those regimes and give a slightly different categorization.

We compare both problems in those regimes with respect to reducibility as well as the asymptotic complexity of known algorithms to solve those problems.

Unique solutions. This first regime corresponds to parameters which yield a unique solution to both problems, i.e., to a weight below the GV bound.

Note that RSD always polynomially reduces to SD. This reduction permutes the columns of the parity-check matrix to obtain a randomly distributed solution, i.e., it calls the SD solver on input $(\mathbf{HP}, \mathbf{s})$ with solution $\mathbf{P}^{-1}\mathbf{e}$, where $\mathbf{P}$ is a random permutation matrix. On the other hand, there is no polynomial reduction from SD to RSD known in this regime.

Further, since the RSD problem allows to encode the regularity and, as we show in Section 4, allows for many other speedups, algorithms that exploit RSD specifics are strictly faster than those solving plain SD in this regime.

Multiple Solutions. This regime corresponds to parameters which yield multiple solutions to RSD as well as SD. Precisely, this regime corresponds to weights exceeding the RSD uniqueness bound and is visualized as the red shaded area in Fig. 2a. Note that in this regime for any given SD instance there exist regular solutions. Therefore one can apply any RSD solver to the given SD instance finding one of those valid regular solutions. This implies that SD polynomially reduces to RSD in this regime.

However, this reduction works vice-versa as for any RSD instance there exist non-regular solutions and, further, there still exist a reduction from RSD to SD regardless of the amount of solutions. Therefore in terms of polynomial reducibility RSD and SD are equivalent in this regime.

Due to the different amount of solutions to the problems and the effect of the reduction on those amounts, it is not immediately possible to derive a strict separation on the asymptotic complexity of algorithms solving both problems in this regime. Nevertheless, we find that for weights exceeding the uniqueness bound either both problems are solvable in polynomial time, i.e., they are equally hard, or RSD is indeed harder with respect to the best known algorithms.

Unique RSD but multiple SD solutions The third regime corresponds to the case where SD has multiple solutions while the RSD solution is still unique. That means, the weight of those instances exceeds the GV bound but still lies below the uniqueness bound. In this regime RSD reduces to SD via the outlined reduction but not vice versa.

In terms of asymptotic complexity of algorithms solving both problems again a strict separation is not possible. Similar to the regime where both problems have multiple solutions the effect of those can not be quantified directly. We find that this regime contains parameters that lead to harder SD instances as well as harder RSD instances with respect to known algorithms.

Classification Based on Known Algorithms In the following we compare the hardness of RSD and SD based on the asymptotic complexity of known algorithms.

We illustrate in Fig. 4 the different parameter regions and their corresponding hardness classification. We mark the region in which RSD is solvable in polynomial time (see Theorems 3.1 and 3.2) as a gray shaded area framed by a solid gray line. The GV bound and the RSD uniqueness bound are depicted as blue and red dashed lines. The blue shaded area corresponds to weights below the GV bound, in which SD is strictly harder than RSD. The area underneath the red dashed line marks the regime of parameters that yield multiple solutions to both problems. The purple region corresponds to the area in which also SD is solvable in polynomial time.

Finally, the green shaded region marks the regime in which we find RSD to be harder than SD. Here the area extending the purple region to a triangle, separated by the dotted

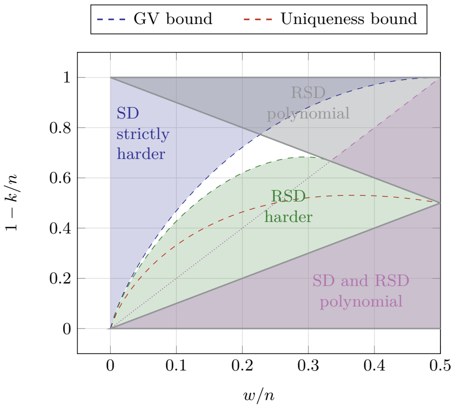


Fig. 4: Hardness comparison between RSD and SD with respect to known algorithms.

line, includes parameters for which SD is solvable in polynomial time, while RSD instances remain exponentially hard. The rest of the green area corresponds to parameters where both problems are exponentially hard and we find RSD instances to be harder than SD instances based on known algorithms. For this classification we compare the running time of the basic ISD algorithm by Prange in the SD case (see Eq. (1)) against the running time of the permutation-based regular-ISD algorithm (see Algorithm 1 and Theorem 4.1). Note that, we also performed this comparison for the most advanced versions of ISD [BM18] and regular-ISD (Algorithm 3) but the picture remains almost the same with a slight increase of the green area. This indicates that the RSD case benefits slightly more from advanced techniques, as an enlarging of the green area corresponds to instances that are closer to the SD worst case (the GV bound) and further away from the RSD worst case, which is for most of the parameters in this region the RSD uniqueness bound.

Note that the white unfilled area between the green and blue area corresponds to parameters where we find SD to be harder with respect to known algorithms.

Comparison for Worst Case Parameters. Fig. 4 shows that there are regimes in which RSD is harder than SD if comparing the problems with the same parameterization. However, in cryptographic constructions parameters are usually chosen considering the individual problem properties. Schemes based on SD would craft parameter sets that are specifically well suited for SD and analogously for RSD. Therefore we compare in following the hardness of SD and RSD for their respective worst case instances. Again, this comparison is with respect to known algorithms, where in Fig. 5 we compare the running time of the fastest ISD algorithm in the SD case by Both-May against the best regular-ISD algorithm (Algorithm 3). We observe that for small rates SD instances following worst case parameters are generally harder with respect to known algorithms. However, interestingly, for all rates $\kappa > 0.49$ RSD instances following worst case parameters are harder to solve.

Comparison for Small Error Weight What is not captured visually in Fig. 4 is that the asymptotic runtime exponents of the time complexity to solve the respective SD and RSD

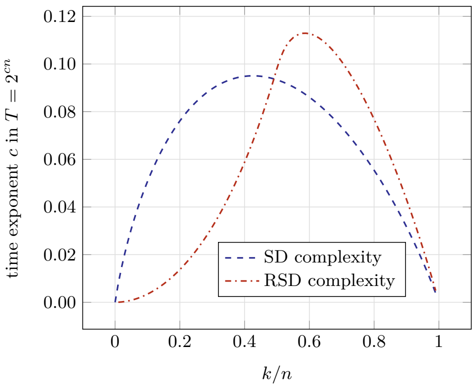


Fig. 5: Comparison of complexity of Both-May algorithm [BM18] on SD instances and regular-ISD algorithm (Algorithm 3) on RSD instances, where instances match respective worst case parameters.

instances converge for $w/n \to 0$, i.e., for $T_{\text{SD}} = 2^{c_{\text{SD}}n+o(n)}$ and $T_{\text{RSD}} = 2^{c_{\text{RSD}}n+o(n)}$ we have $c_{\text{SD}}/c_{\text{RSD}} \xrightarrow{w/n\to 0} 1$. Note that this does not contradict the "SD strictly harder" regime, as for any constant w/n we find $c_{\text{SD}}/c_{\text{RSD}} > 1$. But still, it is an indication that the smaller the solution weight, the closer both problems become in terms of computational complexity. In fact, for w constant, any random permutation of the columns of $\mathbf{H}$ corresponds to a regular solution with inverse polynomial probability. In turn this yields a polynomial reduction from SD to RSD. However, since both problems are solvable in polynomial time if w is a constant the reduction is not very meaningful.

In the following we show that with respect to ISD and regular-ISD algorithms both problems are solvable in same asymptotic running time as long as $w = o(n)$ is sublinear in n. Therefore note that in the SD case for $w = o(n)$, a result by Canto-Torres and Sendrier [TS16] proves that all known ISD algorithms obtain the same asymptotic running time of

$$T_{\text{SD}} = \left(1 - \frac{k}{n}\right)^{-(w+o(w))}.$$

On the other hand Theorem 4.1 states the running time of the permutation-based regular-ISD algorithm as

$$T_{\text{RSD}} = \tilde{\mathcal{O}}\left(\left(1 - \frac{k-w}{n}\right)^{-w}\right).$$

Let $k = \kappa n$ with constant κ and $w = o(n)$ sublinear, then [3]

$$\frac{T_{\text{SD}}}{T_{\text{RSD}}} = \frac{\left(1-\frac{k-w}{n}\right)^w}{\left(1-\frac{k}{n}\right)^w} = \left(1+\frac{w}{n(1-\kappa)}\right)^w = 2^{w\cdot\left(\frac{w}{n(1-\kappa)\ln(2)}+o\left(\frac{w}{n}\right)\right)} = 2^{o(w)}, \tag{6}$$

which shows that both complexities differ only by second order terms.

[3] From Taylor's expansion, for $x = o(1)$, we get $\log_2(1+x) = \frac{x}{\ln(2)} + \mathcal{O}(x^2) = \frac{x}{\ln(2)} + o(x)$. In our case, since $w = o(n)$, we get $w/n = o(1)$.

4 Regular Information Set Decoding

In this section we describe the regular-ISD algorithms that exploit the regular structure of the RSD solution and provide their asymptotic analysis.

The most advanced algorithms following the regular-ISD approach incorporate most modern techniques, including permutations, enumeration, representations and nearest-neighbor search tailored to the regular setting. For didactic reasons we provide an incremental description embedding one technique at a time, following a similar evolution as the ISD literature.

The algorithms we describe use internal parameters of which some must be integers. However, in the initial analysis for the sake of clarity, we ignore all rounding issues by performing computations with non-integer values if necessary. We show in Section 4.6 that these rounding issues have asymptotic effects: they cause a slight increase in the running time in some cases, while for many cases resolving those rounding issues leads to a *decrease* of the the asymptotic complexity. In any case, especially with respect to the worst case complexity exponent, we show that the deviation through these rounding issues is very mild.

4.1 Permutation-Based Regular-ISD

This first algorithm aims to find an error-free information set of the solution $\mathbf{e}$ by random selection or permutation of coordinates.

Error-free information sets. An *error-free* set for $\mathbf{e}$ corresponds to a set J such that $\mathbf{e}_J = \mathbf{0}$. Knowledge of such a set reveals the error vector $\mathbf{e}$ if it is an information set. Therefore, let $\mathbf{P}_J \in \mathbb{F}_2^{n\times n}$ describe any permutation matrix that permutes $\mathbf{e}_J$ to the back of $\mathbf{P}_J\mathbf{e}$, i.e., $\mathbf{P}_J\mathbf{e} = (\mathbf{e}_1, \mathbf{e}_J) \in \mathbb{F}_2^{n-k} \times \mathbb{F}_2^{k}$.[4] In that case we have

$$\mathbf{H}\mathbf{e} = (\mathbf{H}\mathbf{P}_J^{-1})(\mathbf{P}_J\mathbf{e}) = (\mathbf{H}\mathbf{P}_J^{-1})(\mathbf{e}_1, \mathbf{e}_J) = \mathbf{H}_1\mathbf{e}_1 + \mathbf{H}_2\mathbf{e}_J = \mathbf{H}_1\mathbf{e}_1 = \mathbf{s}, \tag{7}$$

where we write $(\mathbf{H}\mathbf{P}_J^{-1}) = (\mathbf{H}_1\mathbf{H}_2)$ with $\mathbf{H}_1 \in \mathbb{F}_2^{(n-k)\times(n-k)}$ and $\mathbf{H}_2 \in \mathbb{F}_2^{k\times(n-k)}$. Note that $\mathbf{H}_2\mathbf{e}_J = \mathbf{0}$ since $\mathbf{e}_J = \mathbf{0}$, as J represents an *error-free* information set. Now, as long as $\mathbf{H}_1$ is invertible we can compute $\mathbf{e}_1 = \mathbf{H}_1^{-1}\mathbf{s}$ and the solution as $\mathbf{e} = \mathbf{P}_J^{-1}(\mathbf{e}_1, 0^k)$. Note that $\mathbf{H}_1$ being invertible is the exact definition of J being an information set, and it happens for random $\mathbf{H}$ with constant probability over the choice of J.

The early ISD algorithm by Prange finds the error-free information set by sampling random permutations $\mathbf{P}$, computing $\mathbf{e}_1' = \mathbf{H}_1^{-1}\mathbf{s}$ and checking if $|\mathbf{e}_1'| = w$, in which case it outputs the solution $\mathbf{P}^{-1}(\mathbf{e}_1', 0^k)$, otherwise it starts over with a new permutation.

Regular Permutations. In general, permutations disregard the regular structure of the error and might lead to a decrease in the probability that $\mathbf{H}_1$ is invertible. Therefore, note that the equations added due to regularity are sparse and any permutation $\mathbf{P}$ that permutes all columns belonging to a single block to $\mathbf{H}_2$, i.e., to the back of the matrix, leads to an $\mathbf{H}_1$ containing zero rows, making it non-invertible.

In order to circumvent this problem and to take advantage of the regular structure of the solution, we modify how permutations are sampled. Namely, we ensure that for every

[4] Recall that J being an information set also implies $|J| = k$.

Algorithm 1: PERM: Permutation-based Regular-ISD

Input : Parity-check matrix $\mathbf{H}' \in \mathbb{F}_2^{(n-k)\times n}$, syndrome $\mathbf{s}' \in \mathbb{F}_2^{n-k}$, weight w
Output : Regular vector $\mathbf{e} \in \mathbb{F}_2^n$ of weight w, such that $\mathbf{He} = \mathbf{s}$

Let $k' := k - w$, $v := k'/w$
Obtain $\mathbf{H} \in \mathbb{F}_2^{(n-k')\times n}, \mathbf{s} \in \mathbb{F}_2^{n-k'}$ from $(\mathbf{H}', \mathbf{s}')$ by encoding regularity
repeat
  Sample random v-regular permutation matrix $\mathbf{P}$
  $\big(\mathbf{H}_1, \mathbf{H}_2\big) \leftarrow \mathbf{HP}$
  if $\mathbf{H}_1$ is non-singular **then**
    $\mathbf{e}_1 \leftarrow \mathbf{H}_1^{-1}\mathbf{s}$
    if $|\mathbf{e}_1| = w$ and $\mathbf{e}_1$ is regular **then**
      return $\mathbf{P}^{-1}(\mathbf{e}_1, 0^{k'})$

permutation the matrix $\mathbf{H}_2$ is formed by selecting $v = \frac{k'}{w} = \frac{k-w}{w}$ columns from each block. Further, to enable later improvements the v columns taken from each block form again a consecutive block in $\mathbf{H}_2$. More formally, we define a v-regular permutation as follows.

Definition 4.1 (Regular Permutation). *Let $\mathbf{e} = (\mathbf{e}_1, \ldots, \mathbf{e}_w) \in (\mathbb{F}_2^b)^w$. For an integer $v \leq b$ and a permutation matrix $\mathbf{P}$ let*

$$\mathbf{Pe} = (\mathbf{e}_1', \ldots, \mathbf{e}_w', \mathbf{e}_1'', \ldots, \mathbf{e}_w''),$$

with $\mathbf{e}_i' \in \mathbb{F}_2^{b-v}$ and $\mathbf{e}_i'' \in \mathbb{F}_2^v$. We call $\mathbf{P}$ a v-regular permutation if each $\mathbf{e}_i'$ and each $\mathbf{e}_i''$ are formed only by coordinates from $\mathbf{e}_i$.

The algorithm now samples permutations uniformly from the set of v-regular permutations until an error-free information set is found. The pseudocode of the overall algorithm is given in Algorithm 1.

Theorem 4.1 (Permutation-based Regular-ISD). *Algorithm 1 solves $\mathcal{RSD}(n, k, w)$ with $w \mid k$ in expected time and memory*

$$T = \tilde{\mathcal{O}}\left(\frac{\left(1 - \frac{k-w}{n}\right)^{-w}}{S}\right) \quad \text{and} \quad M = \tilde{\mathcal{O}}\left(1\right),$$

where S is the amount of existing solutions.

Proof. The correctness follows from the previous argumentation on error-free information sets. Therefore note that as long as $v = k'/w < n/w - 1$ which is equivalent to $k/n \leq 1$ and v being an integer, which follows from $w \mid k$, there always exist v-regular permutations that yield $\mathbf{Pe} = (\mathbf{e}_1, 0^{k'})$.

The expected complexity is the time per iteration divided by the success probability per iteration. First note, that the time spent in each iteration is polynomial, and hence, subsumed in our use of the Landau notation.

An iteration is successful if the chosen permutation distributes the whole support of $\mathbf{e}$ onto the first $n - k'$ coordinates of $\mathbf{Pe}$ and if $\mathbf{H}_1$ is invertible. Note that $\mathbf{H}_1$, due to our restriction to regular permutations has a similar structure as $\mathbf{H}'$ from Eq. (2). Therefore, it is invertible with constant probability as detailed in Remark 3.1. Further, for $b = n/w$ the probability of the permutation distributing the weight as desired is

$$q := \Pr\left[\mathbf{Pe} = (\mathbf{e}_1, 0^{k'})\right] = \left(\frac{\binom{b-1}{v-1}}{\binom{b}{v}}\right)^w = \left(1 - \frac{v}{b}\right)^w = \left(1 - \frac{k'}{n}\right)^w. \tag{8}$$

Therefore note that the permutation has to distribute the single non-zero entry per block to the $\mathbf{H}_1$ part, which happens with probability $\frac{\binom{b-1}{v-1}}{\binom{b}{v}} = 1 - \frac{v}{b}$ per block, while there are w blocks. Note that in case of $S \geq 1$ existing solutions the probability of the permutation distributing the weight as desired for at least one of the solutions is about $q \cdot S$.

Therefore the running time of Algorithm 1 is

$$T = \tilde{\mathcal{O}}\left(\frac{\left(1 - \frac{k-w}{n}\right)^{-w}}{S}\right),$$

since $k' := k - w$. All stored objects are matrices of size polynomial in n, therefore the memory complexity is polynomial, i.e., $M = \tilde{\mathcal{O}}(1)$. □

4.2 Enumeration-Based Regular-ISD

We now extend Algorithm 1 by a meet-in-the-middle enumeration procedure that further exploits the regular structure of the error. Therefore we allow for a certain weight p, later to be optimized, within the coordinates of the information set J, i.e., we search for a set J such that $|\mathbf{e}_J| = p$.

Finiasz-Sendrier modelling We follow a modeling by Finiasz and Sendier [FS09] that increases the size of the set J to $|J| = k + \ell$ with ℓ being another optimization parameter. Let $\mathbf{P}_J$ again be a permutation matrix such that $\mathbf{P}_J\mathbf{e} = (\mathbf{e}_1, \mathbf{e}_J) \in \mathbb{F}_2^{n-k-\ell} \times \mathbb{F}_2^{k+\ell}$, now with $|\mathbf{e}_J| = p$. Further let $\mathbf{Q} \in \mathbb{F}_2^{(n-k)\times(n-k)}$ be a matrix such that

$$\mathbf{QHP}_J^{-1} = \begin{pmatrix} \mathbf{I}_{n-k-\ell} & \mathbf{H}_1 \\ \mathbf{0} & \mathbf{H}_2 \end{pmatrix}, \text{ with } \mathbf{H}_1 \in \mathbb{F}_2^{(n-k-\ell)\times(k+\ell)} \text{ and } \mathbf{H}_2 \in \mathbb{F}_2^{\ell\times(k+\ell)}.$$

This modeling allows to re-write the syndrome equation as

$$\mathbf{QHe} = \mathbf{QHP}_J^{-1}\mathbf{P}_J\mathbf{e} = (\mathbf{e}_1 + \mathbf{H}_1\mathbf{e}_J, \mathbf{H}_2\mathbf{e}_J) = (\mathbf{s}_1, \mathbf{s}_2) = \mathbf{Qs}, \tag{9}$$

with $\mathbf{s}_1 \in \mathbb{F}_2^{n-k-\ell}$ and $\mathbf{s}_2 \in \mathbb{F}_2^{\ell}$.

This implies that once such a set J is known the solution can be found by enumerating all possible values for $\mathbf{e}_J$, i.e., all vectors $\mathbf{x} \in \mathbb{F}_2^{k+\ell}$ of weight p that satisfy $\mathbf{H}_2\mathbf{x} = \mathbf{s}_2$. Among those values there must be one $\mathbf{x} = \mathbf{e}_J$ for which $\mathbf{H}_1\mathbf{x} + \mathbf{s}_2 = \mathbf{e}_1$. Even if $\mathbf{e}_1$ is not known, this $\mathbf{x}$ can be efficiently determined by checking if $|\mathbf{H}_1\mathbf{x} + \mathbf{s}_2| = |\mathbf{e}_1| = w - p$. Eventually the solution can again be reconstructed as $\mathbf{P}_J^{-1}(\mathbf{e}_1, \mathbf{e}_J)$.

Algorithm 2: ENUM: Enumeration-based Regular-ISD

Parameters : $\ell \leq n-k+w$ and $p \leq w$
Input : Parity-check matrix $\mathbf{H}' \in \mathbb{F}_2^{(n-k)\times n}$, syndrome $\mathbf{s}' \in \mathbb{F}_2^{n-k}$
Output : Regular vector $\mathbf{e} \in \mathbb{F}_2^n$ of weight w, such that $\mathbf{He} = \mathbf{s}$

let $k' := k - w$, $v := (k' + \ell)/w$
Obtain $\mathbf{H} \in \mathbb{F}_2^{(n-k')\times n}$, $\mathbf{s} \in \mathbb{F}_2^{n-k'}$ from $(\mathbf{H}', \mathbf{s}')$ by encoding regularity
repeat
  // Permutation and Gaussian Elimination
  Sample a random v-regular permutation matrix $\mathbf{P}$
  $\begin{pmatrix} \mathbf{I}_{n-k'-\ell} & \mathbf{H}_1 \\ \mathbf{0} & \mathbf{H}_2 \end{pmatrix} \leftarrow \mathbf{QHP}^{-1}$, with $\mathbf{H}_1 \in \mathbb{F}_2^{(n-k'-\ell)\times(k'+\ell)}$, $\mathbf{H}_2 \in \mathbb{F}_2^{\ell\times(k'+\ell)}$
  $(\mathbf{s}_1, \mathbf{s}_2) \leftarrow \mathbf{Qs}$ with $\mathbf{s}_1 \in \mathbb{F}_2^{n-k'-\ell}$ and $\mathbf{s}_2 \in \mathbb{F}_2^{\ell}$
  // Enumeration
  $L_i \leftarrow \{\mathbf{z}_i \mid \mathbf{z}_i \in \mathbb{F}_2^{(k'+\ell)/2}$ is v-regular of weight $p/2\}$, $i = 1, 2$
  $L \leftarrow \{(\mathbf{z}_1, \mathbf{z}_2) \in L_1 \times L_2 \mid \mathbf{H}_2(\mathbf{z}_1, \mathbf{z}_2) = \mathbf{s}_2\}$
  for $\mathbf{e}_2 \in L$ **do**
    $\mathbf{e}_1 \leftarrow \mathbf{H}_1\mathbf{e}_2 + \mathbf{s}_1$
    if $|\mathbf{e}_1| = w$ and $\mathbf{e}_1$ is regular **then**
      return $\mathbf{P}^{-1}(\mathbf{e}_1, \mathbf{e}_2)$

Regular Enumeration. For finding a suitable permutation that encodes such a set J we again use the sampling of regular permutations as in Algorithm 1. Recall, that this ensures that any permutation $\mathbf{P}$ permutes exactly $v := (k' + \ell)/w$ columns of each block to a consecutive block within the last $k' + \ell$ columns of $\mathbf{HP}^{-1}$. Note that such permutations $\mathbf{P}$ reflect the regular structure to $\mathbf{Pe} = (\mathbf{e}_1, \mathbf{e}_2) \in \mathbb{F}_2^{n-k'-\ell} \times \mathbb{F}_2^{k'+\ell}$ in the sense that $\mathbf{e}_2$ consists of w blocks of length v each having either weight 0 or weight 1. Put differently, if $\mathbf{P}$ encodes such a set J, $\mathbf{e}_2$ is v-regular of weight p (see Definition 4.1). Similarly, in that case $\mathbf{e}_1$ is $(n/w - v)$-regular of weight $w - p$.

We enumerate possible candidates for $\mathbf{e}_J$ in a meet-in-the-middle fashion taking into account the implied regularity. Therefore we write $\mathbf{e}_J = (\mathbf{z}_1, \mathbf{z}_2)$, with $\mathbf{z}_1, \mathbf{z}_2 \in \mathbb{F}_2^{(k'+\ell)/2}$, allowing to rewrite $\mathbf{H}_2\mathbf{e}_J = \mathbf{s}_2$ as

$$\mathbf{H}_2'\mathbf{z}_1 = \mathbf{H}_2''\mathbf{z}_2 + \mathbf{s}_2, \tag{10}$$

where $\mathbf{H}_2 = (\mathbf{H}_2', \mathbf{H}_2'')$. We then enumerate all possible $\mathbf{z}_i$ in lists L_i, where we only consider those of certain regularity. The full procedure is detailed in pseudocode in Algorithm 2.

Analysis of Algorithm 2. The correctness of the algorithm follows again from the previous argumentation. Therefore observe that once the permutation encodes a set J, $\mathbf{z}_1, \mathbf{z}_2$ from Eq. (10) form v-regular vectors of length- $(k' + \ell)/2$ and weight $p/2$, which are exhaustively enumerated in the lists L_1, L_2.

The expected running time of the algorithm is, analogous to Algorithm 1, the time spend in each iteration divided by the success probability per iteration. The time per iteration is dominated by the construction of the three lists L_1, L_2 and L. Lists L_1 and L_2 enumerate all v-regular vectors $\mathbf{z}_1, \mathbf{z}_2$ of length $(k'+\ell)/2$ and weight $p/2$, where $v := (k'+\ell)/w$. This implies that $|L_1| = |L_2| = \binom{w/2}{p/2} v^{p/2}$. The list L then contains all pairs of elements $(\mathbf{z}_1, \mathbf{z}_2) \in L_1 \times L_2$ that satisfy Eq. (10). This list can be constructed in time linear in the involved list sizes, leading to a time per iteration of $T_{\text{it}} = \tilde{\mathcal{O}}\left(\max(|L_1|, |L_2|, |L|\right)$. The expected size of L is $\mathbb{E}[|L|] = \frac{|L_1 \times L_2|}{2^\ell}$ (see Remark 4.1).

An iteration is successful whenever the permutation distributes the weight on $\mathbf{Pe} = (\mathbf{e}_1, \mathbf{z}_1, \mathbf{z}_2)$ such that both $\mathbf{z}_1$ and $\mathbf{z}_2$ are of weight $p/2$. This happens with probability

$$q = \Pr\left[|\mathbf{z}_1| = |\mathbf{z}_2| = p/2\right] = \binom{w/2}{p/2}^2 \left(\frac{v}{b}\right)^p \left(1 - \frac{v}{b}\right)^{w-p}. \tag{11}$$

Therefore note that the probability of a block of length v in $\mathbf{z}_1$, resp. $\mathbf{z}_2$, being of weight one is $\frac{\binom{b-1}{v-1}}{\binom{b}{v}} = \frac{v}{b}$ and correspondingly it is of weight zero with probability $\left(1 - \frac{v}{b}\right)$. Now there are $\binom{w/2}{p/2}$ ways how the $p/2$ weight-1 blocks can be distributed among the $w/2$ blocks of $\mathbf{z}_1$ or $\mathbf{z}_2$, respectively.

In total this leads to a running time of

$$T = T_{\text{it}}/q = \tilde{\mathcal{O}}\left(\frac{\max\left(\binom{w/2}{p/2} v^{p/2}, \binom{w}{p} v^p / 2^\ell\right)}{\binom{w}{p}\left(\frac{v}{b}\right)^p \left(1 - \frac{v}{b}\right)^{w-p}}\right). \tag{12}$$

Note that here we use the fact that $\binom{w/2}{p/2}^2 = \tilde{\Theta}\left(\binom{w}{p}\right)$. The memory complexity is $M = \tilde{\mathcal{O}}\left(|L_1|\right) = \tilde{\mathcal{O}}\left(\binom{w/2}{p/2} v^{p/2}\right)$, as elements of $|L|$ can be checked on the fly for leading to a solution.

Remark 4.1 (Expected Size of L). Note that for a random matrix $\mathbf{H}_2$ the probability of any vector $\mathbf{v} = (\mathbf{z}_1, \mathbf{z}_2)$ satisfying $\mathbf{H}_2 \mathbf{v} = \mathbf{s}_2$ is $2^{-\ell}$. In particular this follows from the fact that then for each row $\mathbf{h}_i$ of $\mathbf{H}_2$ it holds $\Pr\left[\langle \mathbf{h}_i, \mathbf{v}\rangle = (s_2)_i\right] = \frac{1}{2}$.

In our particular case, we know that the matrix $\begin{pmatrix}\mathbf{H}_1 \\ \mathbf{H}_2\end{pmatrix}$ is not random, but it is spanned by $n - k' - w$ random vectors as well as all vectors $\mathbf{r}_i = (0^{(i-1)v}, 1^v, 0^{(w-i)v})$, $i = 1, \ldots, w$, due to the fact that the parity-check matrix encodes the regularity and we chose v-regular permutations. Now, while for any v-regular vector $\mathbf{v}$ we have $\Pr\left[\langle \mathbf{r}_i, \mathbf{v}\rangle = 1\right] = 1$, it still holds

$$\Pr\left[\langle \mathbf{r}_i + \mathbf{h}, \mathbf{v}\rangle = 1 + s\right] = \Pr\left[\langle \mathbf{h}, \mathbf{v}\rangle = s\right] = \frac{1}{2}$$

for any random parity equation $(\mathbf{h}, s)$. Therefore, as long as $\mathbf{r}_i \notin \text{span}(\mathbf{H}_2)$ for all $i = 1, \ldots, w$, it still holds $\Pr\left[\mathbf{H}_2 \mathbf{v}\right] = 2^{-\ell}$. Finally, the probability that none of the $\mathbf{r}_i$ are in the span of $\mathbf{H}_2$ is

$$\bigcup_{i=1}^{w} \Pr\left[\mathbf{r}_i \notin \text{span}(\mathbf{H}_2)\right] \leq w \cdot \Pr\left[\mathbf{r}_1 \notin \text{span}(\mathbf{H}_2)\right] = \frac{w 2^\ell}{2^{n-k}} = \frac{w}{2^{n-k-\ell}}$$

and therefore negligible as long as $\ell << n - k$.

Experimental Verification. We also provide a proof of concept implementation of Algorithm 2. Using that implementation, we ran experiments to confirm the validity of the above remark and, ultimately, the validity of the analysis of the whole algorithm. In all considered experiments, we verified that list sizes match the expectation from the uniform random case on reduced size instances, as indicated by the above remark. Details on the experiments we ran can be found in Appendix Appendix B. For details on the comprehensive tests and all used parameters we refer to `github.com/Memphisd/Regular-ISD`.

4.3 Representation-based Regular-ISD

Contrary to previous assumptions [CCJ23b], we show in this section how the enumeration procedure can be further improved by the use of a technique known as *representations*.

Representations. While we assume a certain familiarity of the reader with the representation technique[5], let us briefly recall its main idea in the non-regular case. A vector $\mathbf{e} \in \mathbb{F}_2^n$ of weight-p satisfying $\mathbf{He} = \mathbf{s}$ is split in the sum of two vectors $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{F}_2^n$ of weight $p_\mathbf{x} := p/2 + \varepsilon_\mathbf{x}$ for a small $\varepsilon_\mathbf{x}$ that has to be optimized. Therefore we write $\mathbf{e} = \mathbf{x}_1 + \mathbf{x}_2$. Note that there are multiple choices for the $\mathbf{x}_1, \mathbf{x}_2$ in that equation, called *representations* of $\mathbf{e}$. Precisely, there are

$$R_\mathbf{x} = \binom{p}{p/2}\binom{n-p}{\varepsilon_\mathbf{x}}$$

many such representations. The technique builds on the observation that finding one of those representations reveals $\mathbf{e}$ and, hence, aims at enumerating a $1/R_\mathbf{x}$-fraction of all representations. This is achieved by only considering those representations which satisfy

$$\mathbf{Hx}_1 = \mathbf{r}_\mathbf{x} \quad \text{and} \quad \mathbf{Hx}_2 = \mathbf{s}_{[\ell_\mathbf{x}]} + \mathbf{r}_\mathbf{x}, \tag{13}$$

where $\ell_\mathbf{x} := \log R_\mathbf{x}$ and $\mathbf{r}_\mathbf{x} \in \mathbb{F}_2^{\ell_\mathbf{x}}$ is chosen arbitrarily. Note that the probability for any representation satisfying Eq. (13) is $\Pr[\mathbf{Hx}_1 = \mathbf{r}_\mathbf{x}] = 2^{-\ell_\mathbf{x}}$, as $\mathbf{Hx}_1 = \mathbf{r}_\mathbf{x}$ together with $\mathbf{He} = \mathbf{s}$ implies the second part of the equation.

While the values for $\mathbf{x}_1$ and $\mathbf{x}_2$ could now, as in the previous section, be enumerated in a meet-in-the-middle fashion based on the identities from Eq. (13), the technique becomes most effective if applied recursively. Therefore, the $\mathbf{x}_i$ are split again into the sum of vectors $\mathbf{y}_j$. More precisely, we write $\mathbf{x}_1 = \mathbf{y}_1 + \mathbf{y}_2$ and $\mathbf{x}_2 = \mathbf{y}_3 + \mathbf{y}_4$, where the $\mathbf{y}_i \in \mathbb{F}_2^n$ are of weight $p_\mathbf{y} = p_\mathbf{x}/2 + \varepsilon_\mathbf{y}$ for a $\varepsilon_\mathbf{y}$ that has to be optimized. Again each such $\mathbf{x}_1$ and $\mathbf{x}_2$ have $R_\mathbf{y} = \binom{p_\mathbf{x}}{p_\mathbf{x}/2}\binom{n-p_\mathbf{x}}{\varepsilon_\mathbf{y}}$ representations as $(\mathbf{y}_1, \mathbf{y}_2)$ and $(\mathbf{y}_3, \mathbf{y}_4)$ respectively. Therefore again constraints on the exact form of the matrix-vector product $\mathbf{Hy}_i$ are introduced to only enumerate a $1/R_\mathbf{y}$ fraction of those representations, still ensuring all $\mathbf{x}_i$ are constructed. Precisely, letting $\ell_\mathbf{y} := \log R_\mathbf{y}$ and $\mathbf{r}_\mathbf{y} \in \mathbb{F}_2^{\ell_\mathbf{y}}$ the algorithm enforces

$$\mathbf{Hy}_1 = \mathbf{r}_\mathbf{y} \quad \text{and} \quad \mathbf{Hy}_2 = (\mathbf{r}_\mathbf{x})_{[\ell_\mathbf{y}]} + \mathbf{r}_\mathbf{y} \qquad \text{, as well as} \tag{14}$$

$$\mathbf{Hy}_3 = \mathbf{r}_\mathbf{y} \quad \text{and} \quad \mathbf{Hy}_4 = (\mathbf{s} + \mathbf{r}_\mathbf{x})_{[\ell_\mathbf{y}]} + \mathbf{r}_\mathbf{y}. \tag{15}$$

[5] For an introduction we refer to [HJ10, MMT11].

Analogously to the first application of the technique, for any representation $(\mathbf{y}_1, \mathbf{y}_2)$ Eq. (14) is satisfied with probability $2^{-\ell_{\mathbf{y}}}$ giving the desired fraction of representations, as the second part of each equation is implied by the first part in conjunction with Eq. (13). Analogously, the same holds for a representation $(\mathbf{y}_3, \mathbf{y}_4)$ and Eq. (15).

Regular Representations. The representation enhanced algorithm for the regular case follows a similar initial construction as Algorithm 2. As such, it uses the same modelling by Finiasz-Sendrier and samples the same kind of regular permutations. The adaptation lies in the enumeration procedure, i.e., the way we enumerate $\mathbf{e}_J$. Recall that $\mathbf{e}_J$ is a v-regular vector of length $k' + \ell$ and weight p, with $v := (k' + \ell)/w$, satisfying the identity $\mathbf{He}_J = \mathbf{s}_2$. In order to embed representations into the regular enumeration we split $\mathbf{e}_J = \mathbf{x}_1 + \mathbf{x}_2$ in the sum of two v-regular vectors of same length $k' + \ell$ but weight $p_{\mathbf{x}} = p/2 + \varepsilon_{\mathbf{x}}$. Therefore we maintain the regular structure in the addends $\mathbf{x}_i$, which reduces the search space for the enumeration, while still obtaining multiple representations. Note that the number of representations in such a modelling becomes

$$R_{\mathbf{x}} = \binom{p}{p/2}\binom{w-p}{\varepsilon_{\mathbf{x}}} \cdot v^{\varepsilon_{\mathbf{x}}}. \tag{16}$$

As in the non-regular case the first binomial coefficient counts the possibilities how the p weight-1 blocks can be distributed equally over the addends $\mathbf{x}_1$ and $\mathbf{x}_2$. The second coefficient counts the possibilities to position the remaining $\varepsilon_{\mathbf{x}}$ non-zero blocks in which the weight cancels. Note that each canceling block offers v possibilities to place its weight, which leads to the final factor. We apply the technique recursively, splitting $\mathbf{x}_1 = \mathbf{y}_1 + \mathbf{y}_2$ and $\mathbf{x}_2 = \mathbf{y}_3 + \mathbf{y}_4$, where we let $\mathbf{y}_i$ be v-regular vectors of length $k' + \ell$ and weight $p_{\mathbf{y}} = p_{\mathbf{x}}/2 + \varepsilon_{\mathbf{y}}$. The vectors $\mathbf{y}_i$ are then enumerated in a meet-in-the-middle fashion based on the identities from Eqs. (14) and (15). A pseudocode description of the full procedure is given in Algorithm 3.

Analysis of Algorithm 3. The correctness of the procedure follows from the correctness of Algorithm 2 and the argumentation above. Therefore observe that for the enumeration procedure we enforce the necessary constraints $\mathbf{r}_{\mathbf{x}}, \mathbf{r}_{\mathbf{y}}$ of correct length $\ell_{\mathbf{x}} := \log R_{\mathbf{x}}, \ell_{\mathbf{y}} := \log R_{\mathbf{y}}$, where

$$R_{\mathbf{y}} = \binom{p_{\mathbf{x}}}{p_{\mathbf{x}}/2}\binom{w-p_{\mathbf{x}}}{\varepsilon_{\mathbf{y}}} \cdot v^{\varepsilon_{\mathbf{y}}}, \tag{17}$$

analogous to Eq. (16). Hence, the enumeration recovers all candidates for $\mathbf{e}_J$ in the final lists, i.e., all v-regular vectors of weight p satisfying $\mathbf{He}_J = \mathbf{s}_2$, implying that $\mathbf{e}_J$ is found once a correct permutation is encountered.

The running time is the number of iterations divided by the success probability of each iteration. An iteration is successful if the permutation distributes the weight as desired, which we already saw happens with probability q (see Eq. (11)).

The cost per iteration is dominated by the enumeration procedure, that is the construction of the lists. In the initial lists L_i, $i = 1, \ldots, 8$ all v-regular vectors of length $(k' + \ell)/2$ and weight $p_{\mathbf{y}}/2$ are enumerated. Therefore those lists are of size $|L_i| = \binom{w/2}{p_{\mathbf{y}}/2} v^{p_{\mathbf{y}}/2}$. The lists $L_{\mathbf{y}_j}$ are constructed from pairs from lists L_{2j-1}, L_{2j} by enforcing a constraint on $\ell_{\mathbf{y}}$ coordinates and is therefore of expected size $|L_{\mathbf{y}_j}| = |L_i|^2/2^{\ell_{\mathbf{y}}}$. The construction of this list is linear in the size of the L_i and the list $L_{\mathbf{y}_j}$ itself, hence it can be performed in time

$$T_{\mathbf{y}} = \tilde{\mathcal{O}}\left(\max(|L_i|, |L_{\mathbf{y}_i}|)\right).$$

Algorithm 3: Rep: Representation-based Regular-ISD

Parameters: $\ell \leq n-k+w$, $p \leq w$, $\varepsilon_{\mathbf{x}} \leq k-w+\ell-p$ and $\varepsilon_{\mathbf{y}} \leq k-w+\ell-p/2-\varepsilon_{\mathbf{x}}$
Input : Parity-check matrix $\mathbf{H}' \in \mathbb{F}_2^{(n-k)\times n}$, syndrome $\mathbf{s}' \in \mathbb{F}_2^{n-k}$
Output : Regular vector $\mathbf{e} \in \mathbb{F}_2^n$ of weight w, such that $\mathbf{He} = \mathbf{s}$

let $k' := k-w$, $v := (k'+\ell)/w$, $p_{\mathbf{x}} := p/2+\varepsilon_{\mathbf{x}}$ and $p_{\mathbf{y}} := p_{\mathbf{x}}/2+\varepsilon_{\mathbf{y}}$
let $\ell_{\mathbf{x}} = \log R_{\mathbf{x}}$ and $\ell_{\mathbf{y}} = \log R_{\mathbf{y}}$ for $R_{\mathbf{x}}, R_{\mathbf{y}}$ as in Eqs. (16) and (17)
Obtain $\mathbf{H} \in \mathbb{F}_2^{(n-k')\times n}, \mathbf{s} \in \mathbb{F}_2^{n-k'}$ from $(\mathbf{H}', \mathbf{s}')$ by encoding regularity
repeat
  `// Permutation and Gaussian Elimination`
  Sample a regular permutation matrix $\mathbf{P} \in R_v$
  $\begin{pmatrix} \mathbf{I}_{n-k'-\ell} & \mathbf{H}_1 \\ \mathbf{0} & \mathbf{H}_2 \end{pmatrix} \leftarrow \mathbf{QHP}^{-1}$, with $\mathbf{H}_1 \in \mathbb{F}_2^{(n-k'-\ell)\times(k'+\ell)}$, $\mathbf{H}_2 \in \mathbb{F}_2^{\ell\times(k'+\ell)}$
  $(\mathbf{s}_1, \mathbf{s}_2) \leftarrow \mathbf{Qs}$ with $\mathbf{s}_1 \in \mathbb{F}_2^{n-k-\ell}$ and $\mathbf{s}_2 \in \mathbb{F}_2^{\ell}$
  `// Enumeration`
  sample $\mathbf{r}_{\mathbf{x}} \in \mathbb{F}_2^{\ell_{\mathbf{x}}}$,$\mathbf{r}_{\mathbf{y}} \in \mathbb{F}_2^{\ell_{\mathbf{y}}}$
  $L_i \leftarrow \{\mathbf{z}_i \mid \mathbf{z}_i \in \mathbb{F}_2^{(k'+\ell)/2}$ is v-regular of weight $p_{\mathbf{y}}/2\}$, $i = 1, \ldots, 8$
  let $\mathbf{r}_{\mathbf{y}_1} = \mathbf{r}_{\mathbf{y}_3} := \mathbf{r}_{\mathbf{y}}$, $\mathbf{r}_{\mathbf{y}_2} = (\mathbf{r}_{\mathbf{x}})_{[\ell_{\mathbf{y}}]} + \mathbf{r}_{\mathbf{y}}$ and $\mathbf{r}_{\mathbf{y}_4} = (\mathbf{s}_2 + \mathbf{r}_{\mathbf{x}})_{[\ell_{\mathbf{y}}]} + \mathbf{r}_{\mathbf{y}}$
  $L_{\mathbf{y}_i} \leftarrow \{(\mathbf{z}_{2i-1}, \mathbf{z}_{2i}) \in L_{2i-1} \times L_{2i} \mid \mathbf{H}_2(\mathbf{z}_{2i-1}, \mathbf{z}_{2i}) = \mathbf{r}_{\mathbf{y}_i}\}$, $i = 1,2,3,4$
  let $\mathbf{r}_{\mathbf{x}_1} := \mathbf{r}_{\mathbf{x}}$ and $\mathbf{r}_{\mathbf{x}_2} = (\mathbf{s}_2)_{[\ell_{\mathbf{x}}]} + \mathbf{r}_{\mathbf{x}}$
  $L_{\mathbf{x}_i} \leftarrow \left\{\mathbf{y}_{2i-1} + \mathbf{y}_{2i} \mid \mathbf{y}_j \in L_{\mathbf{y}_j} \wedge \mathbf{H}_2(\mathbf{y}_{2i-1} + \mathbf{z}_{2i}) = \mathbf{r}_{\mathbf{y}_i}\right\}$, $i = 1,2$
  $L_{\mathbf{e}_J} \leftarrow \left\{\mathbf{x}_1 + \mathbf{x}_2 \mid \mathbf{x}_j \in L_{\mathbf{x}_j} \wedge \mathbf{H}_2(\mathbf{x}_1 + \mathbf{x}_2) = \mathbf{s}_2\right\}$
  for $\mathbf{e}_J \in L$ **do**
    $\mathbf{e}_1 \leftarrow \mathbf{H}_1\mathbf{e}_J + \mathbf{s}_1$
    if $|\mathbf{e}_1| = w$ and $\mathbf{e}_1$ is regular **then**
      return $\mathbf{P}^{-1}(\mathbf{e}_1, \mathbf{e}_J)$

Lists $L_{\mathbf{x}_j}$ are similarly constructed from pairs of the lists $L_{\mathbf{y}_{2j-1}}, L_{\mathbf{y}_{2j}}$ by enforcing a constraint on $\ell_{\mathbf{x}}$ coordinates. However, note that since every such pair satisfies Eq. (14) (resp. Eq. (15)) it already satisfies the first (resp. second) part of Eq. (13) on $\ell_{\mathbf{y}}$ coordinates. Hence only a new constraint on $\ell_{\mathbf{x}} - \ell_{\mathbf{y}}$ coordinates is introduced. The construction of list $L_{\mathbf{x}_j}$ therefore comes at a cost of

$$T_{\mathbf{x}} = \tilde{\mathcal{O}}\left(\max(|L_{\mathbf{y}_i}|, |L_{\mathbf{y}_i}|^2/2^{\ell_{\mathbf{x}}-\ell_{\mathbf{y}}})\right).$$

Observe that not all the $|L_{\mathbf{y}_i}|^2/2^{\ell_{\mathbf{x}}-\ell_{\mathbf{y}}}$ constructed elements satisfying the corresponding part of Eq. (13), have the correct number of non-zero blocks $p_{\mathbf{x}}$. This happens in the case when not exactly $\varepsilon_{\mathbf{x}}$ blocks cancel in the addition of elements from the lists $L_{\mathbf{y}_{2j-1}}, L_{\mathbf{y}_{2j}}$. We then discard elements that do not form v-regular vectors of weight $p_{\mathbf{x}}$. Due to the choice of constraint-sizes, which ensures that each v regular vector of weight $p_{\mathbf{x}}$ satisfying Eq. (13) is constructed, after discarding ill-formed elements the lists $L_{\mathbf{x}_j}$ are of size $|L_{\mathbf{x}_j}| = \binom{w}{p_{\mathbf{x}}} v^{p_{\mathbf{x}}}/2^{\ell_{\mathbf{x}}}$

Eventually, the final list $L_{\mathbf{e}_J}$ can analogously to list $L_{\mathbf{x}_j}$ be constructed in time

$$T_{\mathbf{e}_J} = \tilde{\mathcal{O}}\left(\max(|L_{\mathbf{x}_j}|, |L_{\mathbf{x}_j}|^2/2^{\ell-\ell_x}\right).$$

Therefore note that any sum $\mathbf{e}'_J$ of elements from lists $L_{\mathbf{x}_1}, L_{\mathbf{x}_2}$ already satisfies the equation $\mathbf{H}_2\mathbf{e}'_J = \mathbf{s}_2$ on $\ell_{\mathbf{x}}$ out of ℓ coordinates (compare to Eq. (13)).

The time complexity of the full algorithm is therefore given by

$$\begin{aligned} T &= \tilde{\mathcal{O}}\left(q^{-1} \cdot \max(T_{\mathbf{y}}, T_{\mathbf{x}}, T_{\mathbf{e}_J})\right) \\ &= \tilde{\mathcal{O}}\left(\frac{\max\left(\binom{w/2}{p_{\mathbf{y}}/2}v^{p_{\mathbf{y}}/2}, \binom{w}{p_{\mathbf{y}}}v_{\mathbf{y}}^{p}/2^{\ell_{\mathbf{y}}}, \binom{w}{p_{\mathbf{y}}}^2 v^{2p_{\mathbf{y}}}/2^{\ell_{\mathbf{x}}+\ell_{\mathbf{y}}}, \binom{w}{p_{\mathbf{x}}}^2 v^{2p_{\mathbf{x}}}/2^{\ell+\ell_{\mathbf{x}}}\right)}{\binom{w}{p}\left(\frac{v}{b}\right)^p\left(1-\frac{v}{b}\right)^{w-p}}\right), \end{aligned}$$

while the memory complexity is equal to the sizes of $L_i, L_{\mathbf{y}}, L_{\mathbf{x}}$, i.e., $M = \tilde{\mathcal{O}}\left(\max(|L_i|, |L_{\mathbf{y}}|, |L_{\mathbf{x}}|)\right)$.

Since for ISD algorithms in the SD context the optimal recursion depth of the representation technique differs, we also computed the complexity increasing the depth by one. However, this variant did not allow to obtain further improvements.

4.4 Nearest-Neighbor-Based Regular-ISD

Finally, we achieve a further improvement over Algorithm 3 by leveraging nearest-neighbor search techniques similar to [MO15].

Nearest-Neighbor-based ISD. Therefore note, that so far Algorithm 3 exploits only the relation $\mathbf{H}_2\mathbf{e}_J = \mathbf{s}_2$ to find the solution, i.e., the second part of Eq. (9). May and Ozerov observed, that the first part of this equation, namely $\mathbf{H}_1\mathbf{e}_J + \mathbf{e}_1 = \mathbf{s}_1$, can be interpreted as a nearest-neighbor identity and in turn be exploited in the candidate search for $\mathbf{e}_J$. Therefore note that if $\mathbf{e}_J = \mathbf{x}_1 + \mathbf{x}_2$, then the equation can be rewritten as

$$\mathbf{H}_1\mathbf{x}_1 = \mathbf{H}_1\mathbf{x}_2 + \mathbf{s}_1 + \mathbf{e}_1.$$

In other words $\mathbf{H}_1\mathbf{x}_1$ is equal to $\mathbf{H}_1\mathbf{x}_2 + \mathbf{s}_1$ up to the addition of $\mathbf{e}_1$. Now, since $\mathbf{e}_1$ is known to have low Hamming weight those values are actually close.

Therefore, once two lists $L_{\mathbf{x}_1}, L_{\mathbf{x}_2}$ with candidates for $\mathbf{x}_1, \mathbf{x}_2$ are constructed one can apply a nearest neighbor search to find those elements $\mathbf{x}'_1, \mathbf{x}'_2$ for which it holds that $\mathbf{H}_1\mathbf{x}'_1 \approx \mathbf{H}_1\mathbf{x}'_2 + \mathbf{s}_1$.

Integrating Nearest-Neighbor search into Algorithm 3. We embed the nearest-neighbor search in the same way. Therefore we exchange line 14 in Algorithm 3 with an application of a nearest neighbor search routine that computes $L_{\mathbf{e}_J}$ as

$$L_{\mathbf{e}_J} = \{\mathbf{x}_1 + \mathbf{x}_2 \colon \mathbf{x}_i \in L_{\mathbf{x}_i} \wedge |\mathbf{H}_1(\mathbf{x}_1 + \mathbf{x}_2) + \mathbf{s}_2| = w - p\}.$$

Note that we do not require the identity $\mathbf{H}_2\mathbf{e}_J = \mathbf{s}_2$ in the construction of $L_{\mathbf{e}_J}$ anymore. Therefore, we set $\ell = \ell_{\mathbf{x}}$, which ensures that the identity is already satisfied on all coordinates for any sum of elements from $L_{\mathbf{x}_1}, L_{\mathbf{x}_2}$. The only change in the complexity of the adapted algorithm in comparison to Algorithm 3 is with regards to the time to construct $L_{\mathbf{e}_J}$, which now becomes

$$T_{\mathbf{e}_J} = \tilde{\mathcal{O}}\left(\max(|L_{\mathbf{x}_j}|, \mathcal{N}(|L_{\mathbf{x}_1}|, n-k-\ell, w-p))\right).$$

Here $\mathcal{N}(|L_{\mathbf{x}_1}|, n-k'-\ell, w-p)$ describes the complexity of the nearest neighbor routine by May-Ozerov [MO15, Theorem 1] (or see [EKZ21, Theorem 1] for a generalization). In our case this routine finds all pairs of elements from the two lists of size $|L_{\mathbf{x}_1}|$ containing length-$(n-k'-\ell)$ vectors that sum to weight-$(w-p)$ vectors.

Remark 4.2 (Regular Nearest-Neighbor Search). We note that the difference between $\mathbf{H}_1\mathbf{x}_1$ and $\mathbf{H}_1\mathbf{x}_2 + \mathbf{s}_1$ is $\mathbf{e}_1$, i.e., a $(n-k'-\ell)/w$-regular vector of weight $w-p$. So far we do not exploit this structure in the nearest neighbor search. Internally, the May-Ozerov nearest neighbor search already applies a permutation to both lists that ensures a certain regular distribution of the difference. More precisely, the algorithm ensures that the difference $\mathbf{e}_1$ is formed by blocks of length v of same weight, where the precise choice of v depends on the analysis, but a common choice is $\log^2(k'+\ell)$. Note that enforcing such a distribution comes at a subexponential runtime overhead subsumed in the asymptotic notation (since lists are of exponential size). How to further exploit the regular structure when v is constant to obtain an asymptotic speedup is non-obvious. However, the second order terms in the complexity of the May-Ozerov algorithm can certainly be reduced if the difference is guaranteed to be regular.

4.5 Asymptotic Complexity Comparison

In this section we compare the theoretical runtime exponent of the different improvements against the state-of-the-art. Therefore note, that the asymptotic complexity of all presented algorithms can be expressed as $2^{c(\kappa,\omega)n}$, where c is a constant depending on the constant rate κ and the constant relative weight ω, where $k = \kappa n$ and $w = \omega n$.

Obtaining the runtime exponent. We obtain the constant c by a standard procedure in the context of ISD algorithms. That is by straightforward application of the well-known approximation $\binom{a}{b} = \Theta\left(2^{ah(b/a)}\right)$, to the previously stated runtime formulas, where h denotes the binary entropy function. Technically, the function c also depends on the optimization parameters, such as ℓ and p in the case of Algorithm 2. We then model those parameters to be linear in n, i.e. $\ell = c_\ell n$ and $p = c_p n$, where c_ℓ, c_p are constants. For given κ, ω, we then numerically minimize $c(\kappa, \omega)$ by the choice of c_ℓ, c_p. The values of all optimal parameters for considered parameters as well as a a python script performing the necessary numerical optimization for arbitrary inputs is available at `github.com/Memphisd/Regular-ISD`.

State-of-the-art. Recently, Briaud and Øygarden [BØ23] presented an algorithm modelling the RSD problem as a multivariate-system. Their algorithm is shown to outperform other approaches for many parameters in the low rate, low weight regime. However, so far the algorithm has not been analyzed in the constant rate, constant relative weight regime. We therefore postpone the comparison against this approach to Section 5 where we consider concrete complexities in various rate and weight settings. At the same time Carozza, Couteau and Joux [CCJ23b] presented an algorithm that is based on an enumeration approach, as well as an algorithm using a linearization technique introduced by Saarinen [Saa07] to attack the

FSB hash function. In the following, we briefly recall the enumeration as well as linearization algorithms and establish their asymptotic complexities.

Asymptotic of the Linearization Algorithm Instead of exploiting the regularity of the solution in form of adding extra equations, i.e. by encoding the regularity, the linearization technique leverages the structure differently. The chosen preprocessing reduces the number of columns of the parity-check matrix as well as the weight of the solution, while still ensuring that the solution to the reduced instance directly reveals the original solution.

Therefore, the first column from each of the w blocks of the parity-check matrix $\mathbf{H}$ is added to all columns within the corresponding block, as well as to the syndrome $\mathbf{s}$. This results in an altered instance $(\mathbf{H}', \mathbf{s}')$ for which it still holds that $\mathbf{H}'\mathbf{e} = \mathbf{s}'$, where $\mathbf{e}$ is the original solution, satisfying $\mathbf{He} = \mathbf{s}$. This is because each block contains exactly one non-zero entry. Further note that the first column of each block of $\mathbf{H}'$ is now zero, and can therefore be discarded, resulting in a matrix $\tilde{\mathbf{H}} \in \mathbb{F}_2^{(n-k)\times(n-w)}$. It then holds $\tilde{\mathbf{H}}\mathbf{e}' = \mathbf{s}'$, where $\mathbf{e}' \in \mathbb{F}_2^{n-w}$ is constructed from $\mathbf{e}$ by deleting the first coordinate of every block.

The solution $\mathbf{e}'$ to the resulting instance $(\tilde{\mathbf{H}}, \mathbf{s}')$ is not exactly regular anymore, but any block of length $n/w - 1$ of $\mathbf{e}'$ is known to have weight *at most* 1, as the deletion might have lowered the weight. After this preprocessing step, the algorithm essentially samples $(k/w-1)$-regular permutations $\mathbf{P}$ with the goal to sample a permutation such that $\mathbf{Pe}' = (\mathbf{e}_1, 0^{k-w})$. Once such a permutation is found, the error can be recovered by linear algebra (compare to Eq. (7)). The Linearization algorithm can therefore be seen as a version of Perm (Algorithm 1), with a different preprocessing step. The complexity of the algorithm was already derived in [CCJ23b] as

$$\tilde{\mathcal{O}}\left(\left(\binom{n/w-1}{n/w-1-\frac{n-k}{w}}\Big/\binom{n/w-2}{n/w-1-\frac{n-k}{w}}\right)^{w(1-w/n)}\right),$$

which using $\binom{a}{b}/\binom{a-1}{b} = \frac{a}{a-b}$ can be rewritten as

$$T_{\text{Linearization}} = \tilde{\mathcal{O}}\left(\left(\frac{n-w}{n-k}\right)^{w(1-w/n)}\right)$$

Asymptotic of the Carozza-Couteau-Joux Algorithm The CCJ algorithm follows a pure enumeration strategy. Therefore, the algorithm relies on a similar technique as the Finiasz-Sendier modelling. The solution $\mathbf{e} = (\mathbf{e}_1, \mathbf{e}_2) \in \mathbb{F}_2^{n-k-\ell} \times \mathbb{F}_2^{k+\ell}$ is split in two parts, for some optimization parameter ℓ. Analogously to Eq. (9) prior to the enumeration a change of basis $\mathbf{Q} \in \mathbb{F}_2^{n-k\times n-k}$ is applied that allows to rewrite the syndrome equation as

$$\mathbf{QHe} = \begin{pmatrix} \mathbf{I}_{n-k-\ell} & \mathbf{H}_1 \\ \mathbf{0} & \mathbf{H}_2 \end{pmatrix}(\mathbf{e}_1, \mathbf{e}_2) = (\mathbf{H}_1\mathbf{e}_2 + \mathbf{e}_1, \mathbf{H}_2\mathbf{e}_2) = (\mathbf{s}_1, \mathbf{s}_2) =: \mathbf{Qs}. \tag{18}$$

The algorithm then enumerates all candidates $\mathbf{x}$ for $\mathbf{e}_2$ exploiting its regularity. Among those candidates it recovers the one $\mathbf{x} = \mathbf{e}_2$ that satisfies $|\mathbf{H}_1\mathbf{x} + \mathbf{s}_1| = |\mathbf{e}_1| = w(1 - \frac{k+\ell}{n})$.

For the enumeration the algorithm uses a meet-in-the-middle strategy, enumerating n/w-regular vectors $\mathbf{e}_2', \mathbf{e}_2''$ of length $(k+\ell)/2$, where $\mathbf{e}_2 = (\mathbf{e}_2', \mathbf{e}_2'')$. Therefore the enumerated lists are of size $L = b^{\frac{k+\ell}{2b}} = \binom{n}{w}^{\frac{w(k+\ell)}{2n}}$. Similar to the Finiasz-Sendrier modelling the algorithm then finds all pairs between the lists that satisfy

$$\mathbf{H}_2(\mathbf{e}_2', \mathbf{0}) = \mathbf{H}_2(\mathbf{0}, \mathbf{e}_2'') + \mathbf{s}_2.$$

On expectation there are $L^2/2^\ell$ pairs satisfying the above identity, leading to a running time of

$$T_{\text{CCJ}} = \tilde{\mathcal{O}}\left(\max\left(L, L^2/2^\ell\right)\right) = \tilde{\mathcal{O}}\left(\max\left(\binom{n}{w}^{\frac{w(k+\ell)}{2n}}, \binom{n}{w}^{\frac{w(k+\ell)}{2n}}/2^\ell\right)\right), \quad (19)$$

while the memory complexity is linear in the list size, i.e., $M_{\text{CCJ}} = \tilde{\mathcal{O}}(L) = \tilde{\mathcal{O}}\left(\binom{n}{w}^{w\frac{(k+\ell)}{2n}}\right)$.

Improvement by adding regularity encoding equations. The authors of [CCJ23b] suggest to add the regularity encoding equations prior to the modelling from Eq. (18). By doing this, the new parity-check matrix would have $n-k+w = n-k'$ rows and the list size would get reduced to $L = \binom{n}{w}^{\frac{w(k'+\ell)}{2n}} = \binom{n}{w}^{\frac{w(k-w+\ell)}{2n}}$. However, we note that adding equations that have support only on their last $k+\ell$ coordinates does not improve the enumeration procedure. This is because the enumeration on those coordinates is already restricted to regular vectors which satisfy those equations by construction. Put differently, the expected number of collisions would be greater than $L^2/2^\ell$; we prove this fact in the following proposition.

Proposition 4.1. *Assume the parity check matrix prior to the transformation in Eq. (18) is extended by the w regularity encoding equations. Further let $\mathbf{e}_2', \mathbf{e}_2'' \in \mathbb{F}_2^{\frac{k'+\ell}{2}}$ be n/w-regular vectors. Then, the probability of any such pair $(\mathbf{e}_2', \mathbf{e}_2'')$ satisfying*

$$\mathbf{H}_2(\mathbf{e}_2', \mathbf{0}) = \mathbf{H}_2(\mathbf{0}, \mathbf{e}_2'') + \mathbf{s}_2$$

is $1/2^{\ell-u}$, where $u := \frac{k-w+\ell}{w}$.

Proof. We denote by $\mathbf{H}'$ the matrix obtained by appending to $\mathbf{H}$ the regularity encoding parity-check equations $\mathbf{r}_i = \left(0^{(i-1)b}, 1^b, 0^{(w-i)b}\right)$, with $i \in \{1, \cdots, w\}$ and let $\mathbf{s}' = (\mathbf{s}, 1^w)$ be the syndrome after that encoding. The CCJ algorithm applies a change of basis $\mathbf{Q} \in \mathbb{F}_2^{(n-k')\times(n-k')}$ such that

$$\mathbf{QH}' = \begin{pmatrix} \mathbf{I}_{n-k'-\ell} & \mathbf{H}_1 \\ \mathbf{0} & \mathbf{H}_2 \end{pmatrix}.$$

We now observe that, for each $\mathbf{r}_i$ having support only in $\{n-k'-\ell+1, \cdots, n\}$, it holds that $\mathbf{r}_i \in \text{Span}\big((\mathbf{0}, \mathbf{H}_2)\big)$. Therefore observe that linear combinations of the rows of $\mathbf{QH}'$ can be expressed as

$$(\mathbf{x}_1, \mathbf{x}_2)\begin{pmatrix} \mathbf{I}_{n-k'-\ell} & \mathbf{H}_1 \\ \mathbf{0} & \mathbf{H}_2 \end{pmatrix} = (\mathbf{x}_1, \mathbf{x}_1\mathbf{H}_1 + \mathbf{x}_2\mathbf{H}_2).$$

Whenever $\mathbf{x}_1$ is not null, we obtain a vector that has at least one non-null entry among the first $n-k'-\ell$ positions. However, since the considered equations $\mathbf{r}_i$ have no support in $\{1, \cdots, n-k'-\ell\}$, they can only be obtained if $\mathbf{x}_1$ is all-zero and, consequently, they are linear combinations of the rows of $(\mathbf{0}, \mathbf{H}_2)$. This implies that the regularity encoding equations $\mathbf{r}_i$ indexed by $i = w - u + 1, \ldots, w$, with $u = \frac{k'+\ell}{w} = \frac{k+\ell}{w} - 1$ live in the span of $(\mathbf{0}, \mathbf{H}_2)$.

For convenience, we ignore the initial zero columns in the following and let $\mathbf{r}'_i$ for $i = w - u + 1, \ldots, w$ be $\mathbf{r}_i$ restricted to its last $k' + \ell$ entries. Since $\mathbf{r}'_i \in \mathrm{Span}(\mathbf{H}_2)$, there exist an invertible $\mathbf{Q}'$ such that

$$\mathbf{Q}'\mathbf{H}_2 = \begin{pmatrix} \mathbf{H}'_2 \\ \mathbf{R} \end{pmatrix},$$

where $\mathbf{R} \in \mathbb{F}_2^{u \times k'+\ell}$ contains the $\mathbf{r}'_i$ as rows and $\mathbf{H}'_2 \in \mathbb{F}_2^{\ell-u \times k'+\ell}$. Correspondingly, it holds $\mathbf{Q}'\mathbf{s}_2 = (\mathbf{s}'_2, 1^u)$, as the syndrome entries added for regular equations are all one.

Now, we have for any pair of regular vectors $\mathbf{e}'_2, \mathbf{e}''_2$

$$\mathbf{H}_2(\mathbf{e}'_2, \mathbf{0}) = \mathbf{H}_2(\mathbf{0}, \mathbf{e}''_2) + \mathbf{s}_2 \;\Leftrightarrow\; \mathbf{Q}'\mathbf{H}_2(\mathbf{e}'_2, \mathbf{e}''_2) = \mathbf{Q}'\mathbf{s}_2 \;\Leftrightarrow\; \begin{pmatrix} \mathbf{H}'_2 \\ \mathbf{R} \end{pmatrix}(\mathbf{e}'_2, \mathbf{e}''_2) = (\mathbf{s}'_2, 1^u).$$

Observe that any regular vector $\mathbf{e} = (\mathbf{e}'_2, \mathbf{e}''_2)$ satisfies $\mathbf{R}\mathbf{e} = 1^u$ by construction. Hence we have

$$\Pr\left[\begin{pmatrix} \mathbf{H}'_2 \\ \mathbf{R} \end{pmatrix}(\mathbf{e}'_2, \mathbf{e}''_2) = (\mathbf{s}'_2, 1^u)\right] = \Pr\left[\mathbf{H}'_2(\mathbf{e}'_2, \mathbf{e}''_2) = \mathbf{s}'_2\right] = 2^{-(\ell-u)}.$$

□

Note that the proof of the above proposition shows that the expected list size deviates from the random case as soon as regularity encoding parity-check equations are added whose support entirely lies in the last $k' + \ell$ coordinates. To avoid any such deviations, we need to ensure that

$$n - (k + \ell - i) \le i \cdot b = i \cdot \frac{n}{w} \quad \Leftrightarrow \quad i \le \frac{n-k}{n/w - 1} = \left(1 - \frac{w}{n}\right)^{-1}\left(1 - \frac{k+\ell}{n}\right) w,$$

where i is the number of regularity encoding equations added to the parity check matrix. This inequality is equivalent to all added vectors having support only on their first $n - k - \ell + i$ coordinates, i.e., the part which is not enumerated. The updated parity check matrix therefore corresponds to a code with dimension $\tilde{k} = k - \left(1 - \frac{w}{n}\right)^{-1}\left(1 - \frac{k+\ell}{n}\right) w$. The runtime formula after encoding regularity for the CCJ algorithm is then given by T_{CCJ} form Eq. (19) by substituting k by $\tilde{k}$.

Improvement by Nearest-Neighbors. Carozza, Couteau and Joux further applied the idea of nearest neighbor search in the spirit of [MO15]. In particular, this corresponds to the choice of $\ell = 0$ while using only the nearest neighbor identity $\mathbf{H}_1\mathbf{e}_2 = \mathbf{e}_1 + \mathbf{s}$ from the modelling in Eq. (18) for the subsequent matching of lists. In the parameter selection of their signature scheme they assume this regular nearest-neighbor search can be performed at no cost. While this leads to conservative parameters it does not yield a constructive algorithm. In the following we use the May-Ozerov nearest-neighbor search to perform this matching. Even though we do not exploit the regular structure, we obtain improvements over the pure enumeration variant. Note that the equations that can be added in this case are $k - \tilde{k}$ as

Algorithm	κ	ω	$c_T(\kappa, \omega)$	$c_M(\kappa, \omega)$	c_T^*
CCJ	0.57	0.1659	0.1404	0.1404	0.1404
Linearization	0.61	0.1350	0.1342	**0.0000**	–
CCJ-MO	0.58	0.1575	0.1281	0.1054	0.1281
Perm	0.60	0.1421	0.1256	**0.0000**	0.1268
Enum	0.60	0.1421	0.1225	0.0287	0.1237
Rep	0.59	0.1496	0.1130	0.0714	0.1134
Rep-MO	0.57	0.1659	**0.1117**	0.0852	**0.1119**

Table 2: Maximum runtime and corresponding memory exponents for different RSD algorithms. Runtime resp. memory are of the form $2^{c_T(\kappa,\omega)n}$ and and $2^{c_M(\kappa,\omega)n}$. Maximum time is obtained for stated κ and ω. Maximum time exponent after resolving rounding issues is given as c_T^*

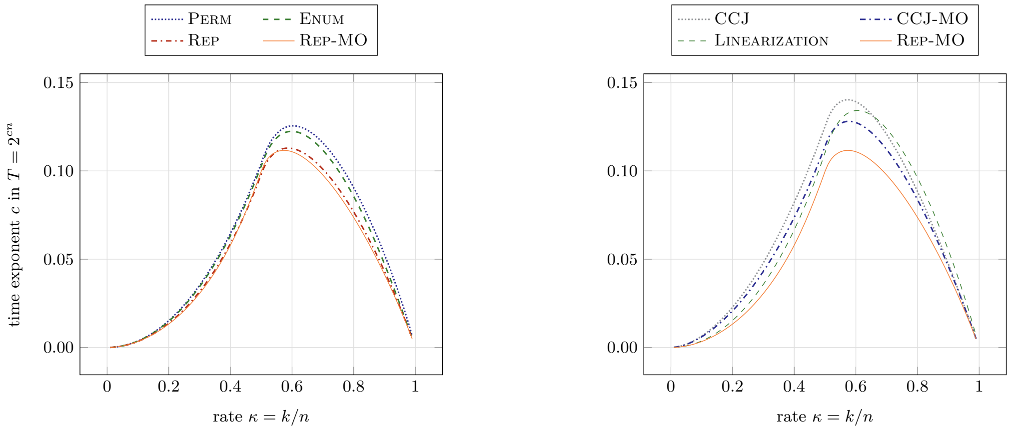
before, now with the choice $\ell = 0$. Further, the nearest-neighbor search comes at a cost of $N := \mathcal{N}\left(L, n - \tilde{k}, \left(1 - \frac{\tilde{k}}{n}\right) w\right)$ (compare to Section 4.4), which results in a runtime of

$$T_{\mathrm{NN}} = \tilde{\mathcal{O}}\left(\max\left(L, N\right)\right). \tag{20}$$

Asymptotic Comparison In the following we compare the worst case decoding complexity of the different regular-ISD algorithms from Sections 4.1 to 4.4 against the state-of-the-art. The worst case complexity corresponds to the runtime exponent maximized over all rates for the respective worst case weight ω^*, i.e., $\max_\kappa c(\kappa, \omega^*)$. In the comparison we refer to the regular-ISD algorithms as Perm (Section 4.1), Enum (Section 4.2), Rep (Section 4.3) and Rep-MO (Section 4.4), where Rep-MO is the nearest neighbor enhanced variant of Rep. Further we denote the linearization algorithm by Linearization, while by CCJ we refer to the Carozza-Couteau-Joux enumeration algorithm [CCJ23b] and by CCJ-MO to the nearest-neighbor enhanced variant of CCJ instantiated via the May-Ozerov nearest-neighbor search routine. In Table 2 we compare the obtained worst case time complexity exponents (and the corresponding memory) for the different algorithms.

We find that CCJ-MO improves on CCJ as well as Linearization in terms of time complexity. Interestingly, we find that Linearization outperforms plain CCJ, contrary to the findings in [CCJ23b]. This is related to the fact that CCJ is not able to make use of all regularity encoding equations (see Proposition 4.1). Overall, we find that regular-ISD algorithms obtain the best time complexities. Notably, Perm offers a polynomial memory instantiation while outperforming all non-regular-ISD approaches. Among regular-ISD approaches we observe a similar behavior as for ISD algorithms in the SD case, where advanced algorithms obtain the time improvements by spending higher amounts of memory. However, even the fastest regular-ISD instantiation uses much less memory than the CCJ-style algorithms.

As indicated in the beginning of Section 4 the analysis of regular-ISD algorithms (as well as Linearization) neglected rounding issues till now. However, we show in the next section, how to resolve these rounding issues for the new algorithms, leading to a slight increase in the worst case decoding exponent. This updated exponent accounting for rounding deviations is illustrated in Table 2 as c_T^*. Note that CCJ-style algorithms do not suffer any rounding issues,



(a) Comparison of regular-ISD from Section 4.

(b) Comparison against other algorithms.

Fig. 6: Comparison of worst case time complexity exponents for RSD algorithms via approximate complexity exponents.

and therefore it holds $c_T^* = c_T(\kappa, \omega)$. Observe that the deviation of the worst case exponent due to rounding issues for all regular-ISD algorithms lies below 0.98% with a minimum of 0.19% in the case of Rep-MO (more details are given in the following Section 4.6).

In Fig. 6 we compare the time complexity exponent for all possible choices of the rate (for the respective worst case weight). On the left we compare the regular-ISD algorithms. Interestingly for rates $0.45 \leq \kappa \leq 0.55$ Rep outperforms Rep-MO. Since this is in contrast to similar nearest-neighbor search improvements in the SD case, it is a strong indication that the used nearest-neighbor routine is suboptimal for the regular case. We pose it as a further research direction to investigate advanced procedures when the distance is known to be regular.

On the right, i.e. in Fig. 6b, we compare the CCJ-style algorithms as well as Linearization against Rep-MO.

4.6 The Asymptotic Effect of Rounding Issues

In the previous sections, for the sake of clarity we ignored all rounding issues. Precisely, we performed computations with a block size of $b = n/w$ and we assumed that the (information) set J is formed by the same amount of $v = \frac{|J|}{w}$ coordinates from each block. However, since $|J|$ and w are linear in n and, hence, b and v are constant, this actually requires $\frac{n}{w}$ and $\frac{|J|}{w}$ to be integers.

As both constants are raised to a power linear in n, compare e.g. to Eq. (8), the rounding might affect the asymptotics. In the following we classify the asymptotic effect of rounding issues by tweaking the analysis to work entirely with integer values. We show that this tweak in many cases leads to a decrease in the asymptotic runtime exponent, which does not affect the validity of the upper bounds derived in the previous sections. While on the other hand increases of the exponents are rather mild, and getting even smaller for more advanced procedures, with Rep-MO obtaining a maximum increase by 0.79%.

Obtaining integer block size. First, for given integers $k = \kappa n$ and corresponding worst case weight $w^* = \omega^* n$ we obtain a valid $\mathcal{RSD}(\tilde{n}, \tilde{k}, w^*)$ instance with integer block size and unique solution by letting

$$b = \left\lceil \frac{1}{\omega} \right\rceil, \quad \tilde{n} = w^* \cdot b \quad \text{and} \quad \tilde{k} = \lceil \kappa \cdot \tilde{n} \rfloor. \tag{21}$$

The new instance obtains rate $\tilde{k}$ very close to κ but has a slightly lower relative weight $\frac{w^*}{n'} \leq \omega^*$, yielding a unique solution with high probability. Overall, we expect this to decrease the obtained complexity exponent slightly as the relative weight does not match the worst case anymore.

Remark 4.3 (Worst Case Weight). We find that resolving rounding issues in some cases leads to a slightly shift of the worst case weight in comparison to the value determined in the previous sections. Namely, using the adaptation from Eq. (21) with $b = 1/\left(\lceil \frac{1}{\omega} \rceil \pm 1\right)$ leads in some cases to a worse complexity. We therefore maximized the complexity exponents in our comparison over those choices.

Sampling the information set. In case w does not divide k, $v = \frac{k'}{w} = \frac{k}{w} - 1$ is not an integer and, hence, the information set J cannot be formed by the same amount of coordinates from all blocks. In such cases, we select $\left\lfloor \frac{k'}{w} \right\rfloor$ coordinates from some blocks, while $\left\lceil \frac{k'}{w} \right\rceil$ from others. Consequently the success probability of sampling an error-free information set J changes.

Let w_{f} and w_{c} denote the number of blocks from which we select $\left\lfloor \frac{k'}{w} \right\rfloor$ and $\left\lceil \frac{k'}{w} \right\rceil$ coordinates, respectively. Since our selection involves all blocks, and the total amount of selected coordinates needs to add to $|J| = k'$, we obtain the relations

$$w_{\mathrm{f}} + w_{\mathrm{c}} = w \quad \text{and} \quad w_{\mathrm{f}} \left\lfloor \frac{k'}{w} \right\rfloor + w_{\mathrm{c}} \left\lceil \frac{k'}{w} \right\rceil = k',$$

which together imply $w_{\mathrm{f}} = w - \left(k' - w \left\lfloor \frac{k'}{w} \right\rfloor\right)$ and $w_{\mathrm{c}} = k' - w \left\lfloor \frac{k'}{w} \right\rfloor$. The modified success probability then becomes (compare to Eq. (8))

$$\tilde{q} = \left(1 - \frac{\lfloor k'/w \rfloor}{b}\right)^{w_{\mathrm{f}}} \left(1 - \frac{\lceil k'/w \rceil}{b}\right)^{w_{\mathrm{c}}}. \tag{22}$$

This leads to a modified time complexity of $\tilde{T} = \tilde{\mathcal{O}}\left(\frac{1}{\tilde{q}}\right) = 2^{\tilde{c}n}$. Let $\left\lfloor \frac{k'}{w} \right\rfloor = \frac{k'}{w} - \varepsilon$, for some $\varepsilon \in [0; 1]$ and, hence, $\left\lceil \frac{k'}{w} \right\rceil = \frac{k'}{w} - \varepsilon + 1$. Then we can rewrite $\tilde{q}$ as follows

$$\begin{aligned} \tilde{q} &= \left(1 - \frac{k'/w - \varepsilon}{b}\right)^{w - \left(k' - w\left(\frac{k'}{w} - \varepsilon\right)\right)} \left(1 - \frac{k'/w - \varepsilon + 1}{b}\right)^{k' - w\left(\frac{k'}{w} - \varepsilon\right)} \\ &= (1 - \kappa + (1+\varepsilon)\omega)^{w(1-\varepsilon)} \left(1 - \kappa + \varepsilon\omega\right)^{\varepsilon w}. \end{aligned}$$

Hence, it follows that

$$\tilde{c} = \frac{1}{n} \log_2(1/\tilde{q}_1) = \omega(1-\varepsilon) \log_2\left(1 - \kappa + (1+\varepsilon)\omega\right) + \varepsilon\omega \log_2(1 - \kappa + \varepsilon\omega). \tag{23}$$

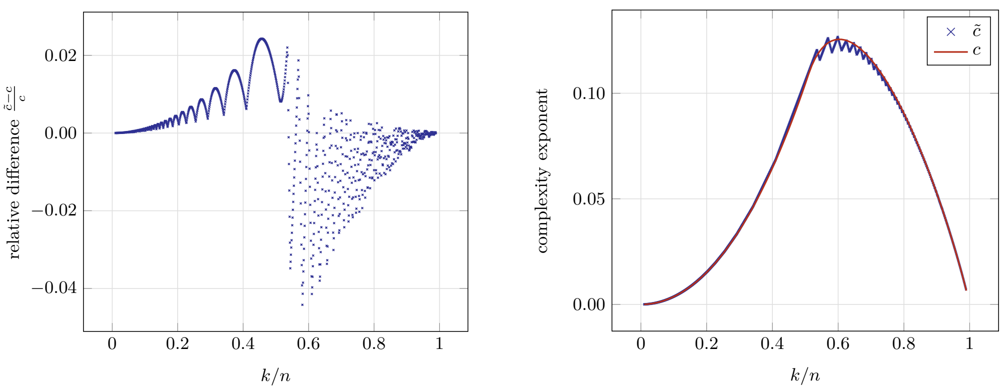
The joint effect on the runtime exponent of Perm. We now compare the exponent obtained in the theoretical analysis and the exponent obtained after resolving rounding issues. The theoretical exponent can be obtained from Theorem 4.1 as

$$c := \frac{1}{n} \log_2 T = \omega \log_2(1 - \kappa + \omega).$$

For obtaining the updated exponent incorporating rounding issues, we first update the instance parameters to obtain integer blocksize and then consider the modified sampling of the information set. A formula for this exponent is obtained by substituting n, ω, κ in Eq. (23) by the parameters of the updated instance $\tilde{n}, \tilde{\omega} = w/\tilde{n}, \tilde{\kappa} = \tilde{k}/\tilde{n}$ from Eq. (21).

In Fig. 7a we report the relative difference $\tilde{c}/c - 1$ between both exponents. We observe that for low rates resolving rounding issues leads to an increase in time complexity while for larger rates the complexity *decreases*. This is due to the initial adaptation of the instance parameters. For small rates κ with correspondingly small worst case weight ω^* the initial rounding of the block size leads only to a small deviation from the worst case relative weight. For larger rates and larger ω^* the deviation becomes more significant. In those cases the decreased difficulty of the initial instance compensates for the lower success probability caused by the rounding on $v = k'/w$. The highest increase by 2.43% is obtained for rate $\kappa = 0.46$.

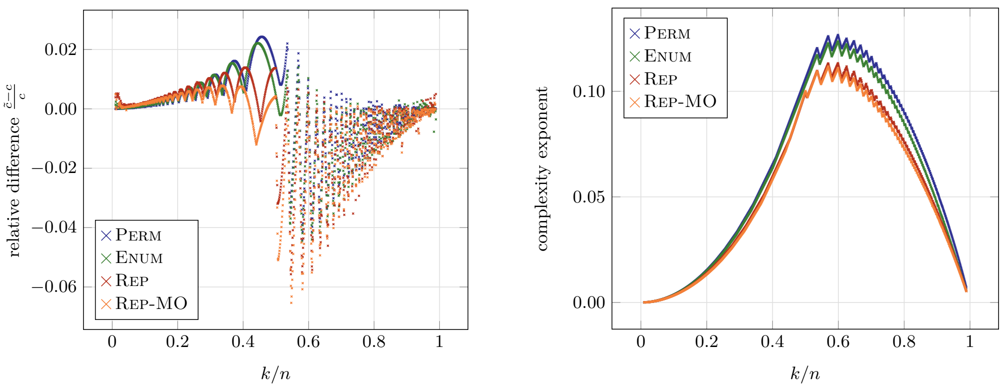
In Fig. 7b we visualize the absolute value of the obtained exponents. Since the rate which achieves the maximum relative difference does not align with the worst case rate, the influence on the worst case complexity of Perm is far smaller than 2.43%. In fact we obtain a new worst case complexity exponent increased by an additive 0.0012 (0.96%) of $\tilde{c}^* = 0.1268$ obtained for rate $\kappa^* = 0.599$ (as previously reported in Table 2). Generally, this shows that the complexity formula obtained in Section 4 forms a good approximate upper bound for the asymptotic complexity.



(a) Relative difference (b) Absolute value

Fig. 7: Comparison of complexity exponent c obtained via Theorem 4.1 vs. $\tilde{c}$ incorporating rounding issues.

Similar, but more technical computations can be performed for the other versions Enum, Rep and Rep-MO. For those variants we find that the impact of rounding issues is decreasing, which stems from more flexibility in how to account for these issues. In Fig. 8a we illustrate the relative difference between the complexity exponents after resolving rounding issues and the approximate exponents obtained in Sections 4.1 to 4.4, while Fig. 8b illustrates the absolute value of the obtained exponents after resolving rounding issues. We find that while for Perm the approximate exponent yields a valid upper bound for about 38.6% of all parameters, for Rep-MO the upper bound holds in more than 55% of all cases. Moreover the maximum excess of the upper-bound for Rep-MO is only 0.79%. We summarize these deviation statistics in Table 3. For completeness we provide the full details on how to resolve rounding issues for these variants in the remainder of this section.



(a) Relative difference

(b) Absolute value

Fig. 8: Complexity exponents $\tilde{c}$ for different regular-ISD variants after resolving rounding issues and relative difference to approximate exponents obtained in Sections 4.1 to 4.4.

Variant	Upperbound Valid	Maximum Excess	Below 1% Excess
Perm	38.6%	2.43%	80.6%
Enum	40.1%	2.21%	83.9%
Rep	47.7%	1.39%	89.5%
Rep-MO	55.2%	0.79%	100 %

Table 3: Deviation statistics of different variants after resolving rounding issues, including percentage of parameters for which upperbounds derived in Section 4 are valid, the maximum excess in case of a bound violation and the fraction of parameters lying below a 1% violation of the upperbound.

Resolving Rounding Issues for Enum. Recall that for the Enum variant the set J is chosen of larger size $|J| = k' + \ell$. The vector $\mathbf{e}_J \in \mathbb{F}_2^{k'+\ell}$ is then further split as $\mathbf{e}_J = (\mathbf{z}_1, \mathbf{z}_2)$, with $\mathbf{z}_1, \mathbf{z}_2 \in \mathbb{F}_2^{\frac{k'+\ell}{2}}$ each constructed by choosing coordinates from $w/2$ blocks. In order to sample exactly $\frac{k'+\ell}{2}$ coordinates for $\mathbf{z}_1$ and $\mathbf{z}_2$ respectively, similar to the Perm case, we choose from some blocks $v = \lfloor \frac{(k'+\ell)/2}{w/2} \rfloor = \lfloor \frac{k'+\ell}{w} \rfloor$ and from others $v+1$ many coordinates. More precisely we choose from w_+ of the blocks $v+1$ coordinates and from $w_- = w/2 - w_+$ we choose only v many. Since it holds $\frac{k'+\ell}{2} = w_- \cdot v + w_+ \cdot (v+1)$ we obtain

$$w_+ = \frac{k' + \ell - w \cdot v}{2} \quad \text{and} \quad w_- = \frac{w(v+1) - (k'+\ell)}{2}.$$

The probability that $\mathbf{z}_i$, $i = 1, 2$ chosen like this has the desired weight $p/2$ becomes

$$\sum_{i=0}^{p/2} \binom{w_-}{i} \left(1 - \frac{v}{b}\right)^{w_- - i} \left(\frac{v}{b}\right)^i \binom{w_+}{\frac{p}{2} - i} \left(1 - \frac{v+1}{b}\right)^{w_+ - \left(\frac{p}{2} - i\right)} \left(\frac{v+1}{b}\right)^{\frac{p}{2} - i}.$$

Here i denotes the number of blocks of length v that contribute to the weight of $\mathbf{z}_i$, i.e., the number of non-zero blocks among the w_- length-v blocks. In our adaptation we restrict to a specific choice of i (later determined by numerical optimization) actually resulting in the probability

$$\Pr\left[|\mathbf{z}_j| = p/2\right] = \binom{w_-}{i} \left(1 - \frac{v}{b}\right)^{w_- - i} \left(\frac{v}{b}\right)^i \binom{w_+}{\frac{p}{2} - i} \left(1 - \frac{v+1}{b}\right)^{w_+ - \left(\frac{p}{2} - i\right)} \left(\frac{v+1}{b}\right)^{\frac{p}{2} - i}. \quad (24)$$

The lists enumerating all possible candidates for $\mathbf{z}_i$ are then of size

$$|L_j| = \binom{w_-}{i} \binom{w_+}{p/2 - i} v^i (v+1)^{p/2 - i},$$

while the final list is of expected size $|L| = \frac{|L_1| \cdot |L_2|}{2^\ell}$. This results in a total expected running time after resolving rounding issues of

$$T' = \tilde{\mathcal{O}}\left(\Pr\left[|\mathbf{z}_j| = p/2\right]^{-2} \cdot \max(|L_j|, |L|)\right).$$

Resolving Rounding Issues for Rep In the case of Rep we use the same resolving strategy for rounding issues with respect to the permutation as for Enum. Therefore the probability of distributing the weight correctly remains $\Pr\left[|\mathbf{z}_j| = p/2\right]^2$ for the probability specified in Eq. (24). Additionally, we have to adapt the analysis of the enumeration procedure to the new block structure. Note that $\mathbf{e}_J = (\mathbf{z}_1, \mathbf{z}_2)$ contains $2w_-$ length-v blocks having a total weight of $2i$, and $2w_+$ length-$(v+1)$ blocks together having weight $p - 2i$. By treating those different length parts independently we can easily adapt the analysis from Section 4.3 to the new structure.

Therefore, we again split $\mathbf{e}_J = \mathbf{x}_1 + \mathbf{x}_2$, but ensure that $\mathbf{x}_1$ and $\mathbf{x}_2$ follow the same block structure, i.e., length-x blocks in $\mathbf{e}_j$ are formed as sums of two length-x blocks from $\mathbf{x}_1, \mathbf{x}_2$.

Furthermore we enforce that the length v-blocks in $\mathbf{x}_i$ have a total weight of $p_{\mathbf{x}}^- = i + \varepsilon_{\mathbf{x}}^-$, while the length-$(v+1)$ blocks total at a weight of $p_{\mathbf{x}}^+ = p/2 - i + \varepsilon_{\mathbf{x}}^+$. Note that this implies that there are

$$R_{\mathbf{x}} = \binom{2i}{i}\binom{2w_- - 2i}{\varepsilon_{\mathbf{x}}^-} \cdot v^{\varepsilon_{\mathbf{x}}^-}\binom{p-2i}{p/2-i}\binom{2w_+ - (p-2i)}{\varepsilon_{\mathbf{x}}^+} \cdot (v+1)^{\varepsilon_{\mathbf{x}}^+}$$

many different representations of $\mathbf{e}_J$ of that form (compare to Eq. (16)).

We continue the analogous splitting of $\mathbf{x}_1 = \mathbf{y}_1 + \mathbf{y}_2$ and $\mathbf{x}_2 = \mathbf{y}_3 + \mathbf{y}_4$ from Section 4.3, where again the $\mathbf{y}_j$ follow the same block structure. Again, we enforce different weight on the different sized blocks, namely $p_{\mathbf{y}}^- = i + \varepsilon_{\mathbf{y}}^-$ on length-v blocks and total weight $p_{\mathbf{y}}^+ = i + \varepsilon_{\mathbf{y}}^+$ on length-$v+1$ blocks, resulting in

$$R_{\mathbf{y}} = \binom{p_{\mathbf{x}}^-}{p_{\mathbf{x}}^-/2}\binom{2w_- - 2i}{\varepsilon_{\mathbf{y}}^-} \cdot v^{\varepsilon_{\mathbf{y}}^-}\binom{p_{\mathbf{x}}^+}{p_{\mathbf{x}}^+/2}\binom{2w_+ - (p-2i)}{\varepsilon_{\mathbf{y}}^+} \cdot (v+1)^{\varepsilon_{\mathbf{y}}^+},$$

different representations of any $\mathbf{x}_j$.

The time complexity of the enumeration routine is still following the same dependence on the list sizes as in Section 4.3, i.e.,

$$T_{\mathrm{E}} = \tilde{\mathcal{O}}\left(\max(|L_j|, |L_{\mathbf{y}_j}|, |L_{\mathbf{y}_j}|^2/2^{\ell_{\mathbf{x}}-\ell_{\mathbf{y}}}, |L_{\mathbf{x}_j}|, |L_{\mathbf{x}_j}|^2/2^{\ell-\ell_{\mathbf{x}}})\right).$$

However, the lists sizes themselves change. Now, the initial lists L_j contain all length $(k'+\ell)/2$ vectors composed of w_- length-v blocks of weight $p_{\mathbf{y}}^-/2$ and w_+ length-$(v+1)$ blocks of total weight $p_{\mathbf{y}}^+/2$. Those lists are therefore of size $|L_j| = \binom{w_-}{p_{\mathbf{y}}^-/2} v^{p_{\mathbf{y}}^-/2}\binom{w_+}{p_{\mathbf{y}}^+/2}(v+1)^{p_{\mathbf{y}}^+/2}$. The lists $L_{\mathbf{y}_o},\ o = 1,2,3,4$ are similarly constructed to before, hence their expected sizes are $|L_{\mathbf{y}_o}| = |L_j|^2/2^{\ell_{\mathbf{y}}}$. Finally, after filtering for the desired weight distribution, i.e., total weight $p_{\mathbf{x}}^-$ on length-v and total weight $p_{\mathbf{x}}^+$ on length-$(v+1)$ blocks, the lists $L_{\mathbf{x}_j}$ are of size $|L_{\mathbf{x}_j}| = \binom{2w_-}{p_{\mathbf{x}}^-} v^{p_{\mathbf{x}}^-}\binom{2w_+}{p_{\mathbf{x}}^+}(v+1)^{p_{\mathbf{x}}^+}/2^{\ell_{\mathbf{x}}}$. In total this results in a time complexity after resolving rounding issues of

$$T' = \tilde{\mathcal{O}}\left(\Pr\left[|\mathbf{z}_j| = p/2\right]^{-2} \cdot T_{\mathrm{E}}\right).$$

Resolving Rounding Issues for Rep-MO Note that to resolve the rounding issues for Rep-MO we proceed exactly as for Rep. The incorporation of the nearest-neighbor routine described in Section 4.4 does not add additional rounding issues. Therefore the time complexity stays as stated in Section 4.4 taking into account the updated success probability and list sizes after resolving the rounding issues as described in the last paragraph.

5 Concrete Complexity of Regular-ISD

We now consider the algorithms introduced in the previous section and derive their concrete time complexity. For all algorithms, the time complexity is of the form T_{it}/q, where T_{it} is the cost per iteration and q is the success probability. Notice that closed form expressions for q have already been provided in Section 4. In order to derive estimates for the concrete cost per iteration, we consider that each Gaussian elimination (as well as Partial Gaussian elimination) takes time $(n-k')^2 n$ and that merging two lists L_1 and L_2 on an ℓ bit constraint takes time $|L_1| + |L_2| + \frac{|L_1 \times L_2|}{2^\ell}$. Further, we assume that all the auxiliary operations, as computing the lists elements or checking the weight after a merge, come at a computational overhead of n.[6] Taking these considerations into account, deriving concrete expressions for T_{it} becomes a simple exercise. For the sake of completeness, we report the full expressions in Appendix A.

We do not consider Rep-MO and CCJ-MO in the concrete comparison, since the May-Ozerov nearest-neighbor routine is known to inherit large polynomial overheads [MO15, EKZ21].

Bit-Security Estimates for RSD Parameters In the following we provide a discussion on the performance of the different algorithms on the parameters suggested in the literature. We compare CCJ [CCJ23a], the recently proposed algebraic attacks by Briaud and Øygarden [BØ23], as well as classical ISD attacks not tailored to the regular case against the regular-ISD approach from Section 4. Note that Briaud and Øygarden do not provide an estimation script to estimate arbitrary instances. We therefore include their estimates only for those parameters provided in their original work. For the estimation of classical ISD attacks, if not provided in the work suggesting the parameters, we rely on the *CryptographicEstimators* library [EVZB23], which incorporates an extension of the *syndrome decoding estimator* by Esser and Bellini [EB22].

The literature suggests a big variety of different parameters. While we provide a discussion about all of them we display in the tables sometimes only a selection of them for clarity. However, we provide the estimates of all suggested parameters together with our source code at `github.com/Memphisd/Regular-ISD`.

For the concrete parameters, whenever the code length n is not a multiple of w, the solution is known to be of the form $\mathbf{e} = (\mathbf{e}', 0^{n-w\cdot b})$ for $b = \lfloor n/w \rfloor$, where $\mathbf{e}'$ is a regular vector of length $w \cdot b$ and weight w. The known zeros, as well as the corresponding columns of the parity check-matrix can be safely discarded. This affects both the code length (which becomes wb) and dimension (which becomes $wb - (n-k)$), hence, the co-dimension remains unchanged with $n-k$.

Rounding Issues. We provide in the following the numbers obtained from the concrete formulas derived by ignoring rounding issues. We note that since the instances we use as input have already an integer blocksize (see the paragraph above), the numbers we give form *valid lower bounds* for the concrete complexity. Therefore recall, that the down-wards deviation observed in Section 4.6 only stems from adapting the instance parameters, while resolving rounding issues for the algorithms themselves only leads to top-wards deviations. However, to verify that resolving those rounding issues leads only to a slight increase in the concrete time complexity we provide in Appendix A.2 those numbers for Perm and Enum.

Parameters Suggested in [LWYY22] In Table 4 we provide the estimated bit complexity of the different approaches on parameters suggested in the context of PCG constructions.

[6] More precise estimates would make assumptions on the specific implementation of the algorithm and make the analysis more involved for an insignificant difference.

(n, k, w)	previous approaches				regular-ISD		
	ISD [LWYY22]	Algebraic [BØ23]	CCJ [CCJ23b]	LINEAR. [CCJ23b]	PERM Section 4	ENUM Section 4	REP Section 4
$(2^{10}, 652, 106)$	164	145	129	139	133	115	**113**
$(2^{12}, 1589, 172)$	135	135	160	132	131	110	**109**
$(2^{14}, 3482, 338)$	135	138	204	141	140	**118**	**118**
$(2^{16}, 7391, 667)$	139	139	249	149	149	**126**	127
$(2^{18}, 15336, 1312)$	144	**122**	274	150	150	126	128
$(2^{20}, 32771, 2467)$	148	**125**	335	164	164	138	141
$(2^{22}, 64770, 4788)$	149	**103**	360	165	165	140	143
$(2^{10}, 652, 57)$	94	101	90	96	94	77	**76**
$(2^{12}, 1589, 98)$	86	103	115	96	96	**78**	**78**
$(2^{14}, 3482, 198)$	91	106	143	103	103	**84**	85
$(2^{16}, 7391, 389)$	96	108	171	110	110	**90**	92
$(2^{18}, 15336, 760)$	101	104	192	114	114	**93**	97
$(2^{20}, 32771, 1419)$	106	**98**	216	119	119	**98**	103
$(2^{22}, 64770, 2735)$	108	**103**	233	123	123	**103**	108

Table 4: Comparison of RSD solvers on large weight (top) and small weight (bottom) instances from [LWYY22].

Notice that the code parameters on the top and bottom of the table are the same, but the values of w are different; therefore, we distinguish between *large weight* (top) and *small weight* (bottom) instances.

We observe that, for almost all instances, regular-ISD either outperforms previous approaches or obtains a similar running time. The only instances for which this is not the case are the very low rate instances with higher weight (bottom of top half), for which the algebraic attack obtains the best performance. In general, regular-ISD obtains the highest gains if rate and relative weight are moderately large (see first rows of top and bottom halves). In case rate and relative weight decrease, we find that the performance of regular-ISD and standard ISD get closer (see last rows of top and bottom halves). This effect can be especially observed in the bottom half of Table 4 which includes the *small weight* instances. Recall, that we have shown in Section 3.4 that the complexities of PERM and general ISD algorithms converge for small relative weight. The bit-security results seem to further indicate that, as in the SD case [TS16], enumeration improvements become less effective for small weight, since also the complexities of ENUM and REP converge to those of ISD.

Parameters Suggested in [HOSS18] The authors of that work propose 32 different parameter sets for two different MPC protocols, all targeting 128-bit security.

The first twelve sets relate to the first MPC protocol (labeled GMW-style in [HOSS18]). Among those are seven parameters with high rate $\kappa \geq 0.96$ and small relative weight $\omega \leq 0.004$. For those regular-ISD is roughly on par with either CCJ or standard ISD. However, we find that some of those parameter sets even when considering only ISD do only roughly match the claimed security, which we show in Table 5. For the remaining four parameter sets with high rate we find huge margins of more than 110 bits. The parameters with rate

(n, k, w)	previous approaches			regular-ISD		
	ISD [EB22]	CCJ [CCJ23b]	LINEAR. [CCJ23b]	PERM Section 4	ENUM Section 4	REP Section 4
[HOSS18, Tab. 1, (GMW-style)]						
$(245760, 245460, 15)$	127	**124**	180	179	127	140
$(40960, 40660, 20)$	**126**	**126**	174	172	**126**	129
$(7680, 7380, 30)$	**127**	131	169	166	132	132
$(1280, 860, 80)$	134	132	141	137	117	**114**
[YWL+20, Tab. 5]						
$(609728, 36288, 1269)$	147	343	164	164	**140**	143
$(10805248, 589760, 1319)$	**155**	492	176	176	157	164
[CCJ23a, pp. 560]						
$(1842, 825, 307)$	289	193	183	178	156	**153**

Table 5: Comparison of different approaches on selected instances from the literature.

$\kappa < 0.96$ use larger (but still small) weight $\omega < 0.016$. For those sets regular-ISD obtains slight improvements, but still those sets enjoy a comfortable margin of at least 10 bits. A notable exception is the parameter set $(1280, 860, 80)$, which is the only set using moderate relative weight of $\omega = 80/1280 = 0.0625$. For this set regular-ISD lowers the bit-security from 132 to 115 and therefore below the security target of 128-bit.

The next twenty suggested parameters relate to the second MPC protocol, labeled BMR-style in [HOSS18]. Most of the parameters again use high rate, low weight with some exceptions. Especially, for those exceptions regular-ISD improves on previous techniques, lowering estimates by up to 11 bits. However, we find that still all sets but two incorporate huge margins of 60 to 270 bits. The two outliers obtain margins of 18 and 39 bit.

Parameters Suggested in [BCG+19a] The authors suggest in that work sixteen different parameter sets for a correlated OT application, eight aiming at 80-bit security and eight aiming for 128-bit. All parameter sets follow rate $\kappa \in \{\frac{1}{2}, \frac{3}{4}\}$, in combination with very low relative weight $\omega \leq 0.0063$. Due to the low relative weights, we find that regular-ISD and ISD algorithms perform roughly on par. All parameters satisfy the security goals, while for 80-bit instances we find margins between 8 to 29 bits and for 128-bit instances we find margins ranging from 10 to 42 bits.

Parameters Suggested in [YWL+20] This work uses two parameter sets within a correlated OT construction. For the set used within the setup, regular-ISD lowers the bit-security estimate from 147 to 140 bit, while for the second parameter set classic ISD leads to a security estimate of 155 bit as shown in Table 5.

Parameters Suggested in [CCJ23a] The authors use a single parameter set for their MPC-in-the-Head based signature construction. The regular-ISD approach lowers the bit-security estimate notably by 30 bits from 183 to 153 bits (see Table 5). Originally, this parameter set aims at 128-bit of security. However, the authors followed a conservative parameter selection in which they considered the running time of CCJ-MO assuming the nearest neighbor search routine comes at no additional cost (corresponding to $N = L$ in Eq. (20)). Note that while here this led to conservative parameters, in general this is not guaranteed. We find several parameters for which regular-ISD significantly outperforms such an "optimistic" version of CCJ-MO.

Acknowledgements We thank the anonymous reviewers of Eurocrypt'24 for their comments that greatly helped us in improving our work. We also want to express our sincere gratitude to Pierre Briaud and Martin Øygarden for fruitful discussions that greatly contributed to the refinement of this work.

References

AFS05. Daniel Augot, Matthieu Finiasz, and Nicolas Sendrier. A family of fast syndrome based cryptographic hash functions. In *Progress in Cryptology–Mycrypt 2005: First International Conference on Cryptology in Malaysia, Kuala Lumpur, Malaysia, September 28-30, 2005. Proceedings 1*, pages 64–83. Springer, 2005.

AG11. Sanjeev Arora and Rong Ge. New algorithms for learning in presence of errors. In Luca Aceto, Monika Henzinger, and Jiri Sgall, editors, *ICALP 2011, Part I*, volume 6755 of *LNCS*, pages 403–415. Springer, Heidelberg, July 2011. doi:10.1007/978-3-642-22006-7_34.

BCG+19a. Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Peter Rindal, and Peter Scholl. Efficient two-round OT extension and silent non-interactive secure computation. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 291–308. ACM Press, November 2019. doi:10.1145/3319535.3354255.

BCG+19b. Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators: Silent OT extension and more. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 489–518. Springer, Heidelberg, August 2019. doi:10.1007/978-3-030-26954-8_16.

BCG+20. Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators from ring-LPN. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 387–416. Springer, Heidelberg, August 2020. doi:10.1007/978-3-030-56880-1_14.

BCGI18. Elette Boyle, Geoffroy Couteau, Niv Gilboa, and Yuval Ishai. Compressing vector OLE. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 896–912. ACM Press, October 2018. doi:10.1145/3243734.3243868.

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 David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 520–536. Springer, Heidelberg, April 2012. doi:10.1007/978-3-642-29011-4_31.

BM18. Leif Both and Alexander May. Decoding linear codes with high error rate and its impact for LPN security. In Tanja Lange and Rainer Steinwandt, editors, *Post-Quantum Cryptography - 9th International Conference, PQCrypto 2018*, pages 25–46. Springer, Heidelberg, 2018. doi:10.1007/978-3-319-79063-3_2.

BØ23. Pierre Briaud and Morten Øygarden. A new algebraic approach to the regular syndrome decoding problem and implications for PCG constructions. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 391–422. Springer, Heidelberg, April 2023. doi:10.1007/978-3-031-30589-4_14.

CCJ23a. Eliana Carozza, Geoffroy Couteau, and Antoine Joux. Short signatures from regular syndrome decoding in the head. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 532–563. Springer, Heidelberg, April 2023. doi:10.1007/978-3-031-30589-4_19.

CCJ23b. Eliana Carozza, Geoffroy Couteau, and Antoine Joux. Short signatures from regular syndrome decoding in the head. Cryptology ePrint Archive, Paper 2023/1035, 2023. https://eprint.iacr.org/2023/1035. URL: https://eprint.iacr.org/2023/1035.

CDMHT22. Kevin Carrier, Thomas Debris-Alazard, Charles Meyer-Hilfiger, and Jean-Pierre Tillich. Statistical decoding 2.0: Reducing decoding to LPN. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part IV*, volume 13794 of *LNCS*, pages 477–507. Springer, Heidelberg, December 2022. doi:10.1007/978-3-031-22972-5_17.

CLY+24. Hongrui Cui, Hanlin Liu, Di Yan, Kang Yang, Yu Yu, and Kaiyi Zhang. Resolved: Shorter signatures from regular syndrome decoding and vole-in-the-head. Cryptology ePrint Archive, Paper 2024/040, 2024. https://eprint.iacr.org/2024/040. URL: https://eprint.iacr.org/2024/040.

EB22. Andre Esser and Emanuele Bellini. Syndrome decoding estimator. In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *PKC 2022, Part I*, volume 13177 of *LNCS*, pages 112–141. Springer, Heidelberg, March 2022. doi:10.1007/978-3-030-97121-2_5.

EKZ21. Andre Esser, Robert Kübler, and Floyd Zweydinger. A faster algorithm for finding closest pairs in hamming metric. In Mikolaj Bojanczyk and Chandra Chekuri, editors, *41st IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2021, December 15-17, 2021, Virtual Conference*, volume 213 of *LIPIcs*, pages 20:1–20:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.FSTTCS.2021.20.

Ess23. Andre Esser. Revisiting nearest-neighbor-based information set decoding. In *IMA International Conference on Cryptography and Coding*, pages 34–54. Springer, 2023.

EVZB23. Andre Esser, Javier Verbel, Floyd Zweydinger, and Emanuele Bellini. CryptographicEstimators: a software library for cryptographic hardness estimation. *Cryptology ePrint Archive*, 2023.

FG15. Jason Fulman and Larry Goldstein. Stein's method and the rank distribution of random matrices over finite fields. 2015.

FS09. Matthieu Finiasz and Nicolas Sendrier. Security bounds for the design of code-based cryptosystems. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 88–105. Springer, Heidelberg, December 2009. doi:10.1007/978-3-642-10366-7_6.

HJ10. Nick Howgrave-Graham and Antoine Joux. New generic algorithms for hard knapsacks. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 235–256. Springer, Heidelberg, May / June 2010. doi:10.1007/978-3-642-13190-5_12.

HKL+12. Stefan Heyse, Eike Kiltz, Vadim Lyubashevsky, Christof Paar, and Krzysztof Pietrzak. Lapin: An efficient authentication protocol based on ring-LPN. In Anne Canteaut, editor, *FSE 2012*, volume 7549 of *LNCS*, pages 346–365. Springer, Heidelberg, March 2012. doi:10.1007/978-3-642-34047-5_20.

HOSS18. Carmit Hazay, Emmanuela Orsini, Peter Scholl, and Eduardo Soria-Vazquez. TinyKeys: A new approach to efficient multi-party computation. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 3–33. Springer, Heidelberg, August 2018. doi:10.1007/978-3-319-96878-0_1.

LPR10. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 1–23. Springer, Heidelberg, May / June 2010. doi:10.1007/978-3-642-13190-5_1.

LWYY22. Hanlin Liu, Xiao Wang, Kang Yang, and Yu Yu. The hardness of LPN over any integer ring and field for PCG applications. Cryptology ePrint Archive, Report 2022/712, 2022. https://eprint.iacr.org/2022/712.

MMT11. Alexander May, Alexander Meurer, and Enrico Thomae. Decoding random linear codes in $\tilde{\mathcal{O}}(2^{0.054n})$. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 107–124. Springer, Heidelberg, December 2011. doi:10.1007/978-3-642-25385-0_6.

MO15. Alexander May and Ilya Ozerov. On computing nearest neighbors with applications to decoding of binary linear codes. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 203–228. Springer, Heidelberg, April 2015. doi:10.1007/978-3-662-46800-5_9.

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

Saa07. Markku-Juhani Olavi Saarinen. Linearization attacks against syndrome based hashes. In K. Srinathan, C. Pandu Rangan, and Moti Yung, editors, *INDOCRYPT 2007*, volume 4859 of *LNCS*, pages 1–9. Springer, Heidelberg, December 2007.

TS16. Rodolfo Canto Torres and Nicolas Sendrier. Analysis of information set decoding for a sub-linear error weight. In *Post-Quantum Cryptography*, pages 144–161. Springer, 2016.

YWL+20. Kang Yang, Chenkai Weng, Xiao Lan, Jiang Zhang, and Xiao Wang. Ferret: Fast extension for correlated OT with small communication. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 1607–1626. ACM Press, November 2020. doi:10.1145/3372297.3417276.

A Concrete Time Complexity Formulas

In this section we provide for completeness the concrete runtime formulas for the regular-ISD algorithms Perm, Enum and Rep, as well as the formula for CCJ.

A.1 Simplified Formulas Ignoring Rounding Issues

Permutation-based Regular-ISD The most time consuming operation of one iteration of Algorithm 1 is the Gaussian elimination. All the other operations (e.g., applying the permutation and checking the weight of $\mathbf{e}_1$) are less costly and can, hence, be neglected, so that $T_{\text{it}} = n(n-k')^2$.

Enumeration-based Regular ISD In each iteration, the algorithm performs one partial Gaussian elimination and creates two lists with size $|L_1| = |L_2| = \binom{w/2}{p/2} v^{p/2}$, with $v := \frac{k'+\ell}{2}$ and $k' = k - w$. The lists are then merged into L, with average size $|L| = |L_1|^2 2^{-\ell}$. Putting everything together, we have that the cost of each iteration is

$$\begin{aligned} T_{\text{it}} &= n\left((n-k')^2 + \left(|L_1| + |L_2| + |L|\right)\right) \\ &= n\left((n-k')^2 + \cdot \binom{w/2}{p/2} v^{p/2} \cdot \left(2 + \binom{w/2}{p/2} v^{p/2} 2^{-\ell}\right)\right). \end{aligned}$$

The optimal time complexity is achieved when $|L| \approx |L_1|$, which implies

$$\ell \approx \log |L_1| = \log_2 \left(\binom{w/2}{p/2} \right) + \frac{p}{2} \log_2 \left(\frac{k'+\ell}{2} \right).$$

Representation-based ISD Again we have $v = \frac{k'+\ell}{w}$. For each level, we recall the number of lists and their average sizes.

- *Creating the initial lists*: the algorithm starts with 8 lists L_i, each with the same size

$$|L_1| = \binom{w/2}{p_{\mathbf{y}}/2} v^{p_{\mathbf{y}}/2}.$$

- *Obtaining* $L_{\mathbf{y}_i}$, $i = 1, \cdots, 4$: each list $L_{\mathbf{y}_i}$ is obtained as the merge of two of the initial lists, searching for collisions in $\ell_{\mathbf{y}}$ coordinates. The average number of collisions, which also corresponds to the average size of each $L_{\mathbf{y}_i}$, is

$$|L_{\mathbf{y}_i}| = |L_1|^2 2^{-\ell_{\mathbf{y}}} = \binom{w/2}{p_{\mathbf{y}}/2}^2 v^{p_{\mathbf{y}}} 2^{-\ell_{\mathbf{y}}}.$$

- *Obtaining* $L_{\mathbf{x}_i}$, $i = 1, 2$: each of these lists is the merge of a pair of lists $L_{\mathbf{y}_i}$. The average number of collisions is

$$N_{\mathbf{y}} = |L_{\mathbf{y}_1}|^2 2^{-(\ell_{\mathbf{x}} - \ell_{\mathbf{y}})} = |L_1|^4 2^{-2\ell_{\mathbf{y}}} 2^{-(\ell_{\mathbf{x}} - \ell_{\mathbf{y}})} = \binom{w/2}{p_{\mathbf{y}}/2}^4 v^{2p_{\mathbf{y}}} 2^{-(\ell_{\mathbf{x}} + \ell_{\mathbf{y}})}.$$

  After collisions are filtered for the desired weight $p_{\mathbf{x}}$, the expected number of elements in each list $L_{\mathbf{x}_i}$ is

$$|L_{\mathbf{x}_i}| = \left(\binom{w/2}{p_{\mathbf{x}}/2} v^{p_{\mathbf{x}}/2} \right)^2 \cdot 2^{-\ell_{\mathbf{x}}} = \binom{w/2}{p_{\mathbf{x}}/2}^2 v^{p_{\mathbf{x}}} \cdot 2^{-\ell_{\mathbf{x}}}.$$

  Here we take into account that only elements balanced elements, i.e., elements with $p_{\mathbf{x}}/2$ non-zero blocks on the first and second half of their coordinates can be constructed. This fact was previously disregarded as it is subsumed in the Landau notation.
- *Obtaining* $L_{\mathbf{e}_J}$: the expected number of produced collisions is

$$N_{\mathbf{x}} = |L_{\mathbf{x}_i}|^2 2^{-(\ell - \ell_{\mathbf{x}})} = \binom{w/2}{p_{\mathbf{x}}/2}^4 v^{2p_{\mathbf{x}}} 2^{-(\ell + \ell_{\mathbf{x}})}.$$

Putting everything together, we get

$$T_{\text{it}} = n(n-k')^2 + n \cdot \big(8|L_1| + 4|L_{\mathbf{y}_1}| + 2N_{\mathbf{y}} + 2|L_{\mathbf{x}_1}| + N_{\mathbf{x}} \big).$$

Substituting each of the above terms, we get

$$T_{\text{it}} = n(n-k')^2 + n \cdot \left(8 \binom{w/2}{p_{\mathbf{y}}/2} v^{p_{\mathbf{y}}/2} + 4 \binom{w/2}{p_{\mathbf{y}}/2}^2 v^{p_{\mathbf{y}}} 2^{-\ell_{\mathbf{y}}} + 2 \binom{w/2}{p_{\mathbf{y}}/2}^4 v^{2p_{\mathbf{y}}} 2^{-(\ell_{\mathbf{x}} + \ell_{\mathbf{y}})} \right.$$
$$\left. + 2 \binom{w/2}{p_{\mathbf{x}}/2}^2 2^{-\ell_{\mathbf{x}}} + \binom{w/2}{p_{\mathbf{x}}/2}^4 v^{2p_{\mathbf{x}}} 2^{-(\ell + \ell_{\mathbf{x}})} \right)$$

Recall that $p_{\mathbf{x}} = p/2 + \varepsilon_{\mathbf{x}}$ and $p_{\mathbf{y}} = p_{\mathbf{x}}/2 + \varepsilon_{\mathbf{y}}$. Furthermore, $\ell_{\mathbf{x}} \approx \log R_{\mathbf{x}}$ and $\ell_{\mathbf{y}} \approx \log R_{\mathbf{y}}$, where[7]

$$R_{\mathbf{x}} = \binom{p/2}{p/4}^2 \binom{(w-p)/2}{\varepsilon_{\mathbf{x}}/2}^2 v^{\varepsilon_{\mathbf{x}}} \quad \text{and} \quad R_{\mathbf{y}} = \binom{p_{\mathbf{x}}/2}{p_{\mathbf{x}}/4}^2 \binom{(w-p_{\mathbf{x}})/2}{\varepsilon_{\mathbf{y}}/2}^2 v^{\varepsilon_{\mathbf{y}}}.$$

Linearization algorithm An iteration of the algorithm performs essentially one Gaussian elimination, plus other operations whose cost can be neglected. So, $T_{\text{it}} = n(n-k)^2$.

CCJ algorithm In this algorithm, lists have size $|L_1| = |L_2| = \left(\frac{n}{w}\right)^{\frac{w(\tilde{k}+\ell)}{2n}}$, where $\tilde{k} = k - \left(1 - \frac{w}{n}\right)^{-1}\left(1 - \frac{k+\ell}{n}\right) w$. The average number of collisions is $|L| = |L_1| \cdot |L_2| \cdot 2^{-\ell} = \left(\frac{n}{w}\right)^{\frac{w(\tilde{k}+\ell)}{n}} 2^{-\ell}$, so that the cost of one iteration is

$$T_{\text{it}} = n(n-\tilde{k})^2 + n \cdot \left(\frac{n}{w}\right)^{\frac{w(\tilde{k}+\ell)}{2n}} \cdot \left(2 + \left(\frac{n}{w}\right)^{\frac{w(\tilde{k}+\ell)}{2n}} 2^{-\ell}\right).$$

A.2 Formulas After Resolving Rounding Issues

As we already discussed in Section 4.6, the effect of rounding issues on the time complexity of the proposed algorithms is expected to be rather mild. Further in Section 5 only rounding issues regarding the algorithms themselves have been neglected, while an instance adaptation that leads to integer blocksize has been performed in all cases. Therefore incorporating rounding issues only leads to an increase in the time complexity. This is confirmed by numbers we present in Table 6 for the case of Perm and Enum when resolving rounding issues for the parameters considered in Tables 4 and 5.

The used formulas are obtained similar to the ones in Appendix A from the formulas stated in Section 4.6, by additionally resolving asymptotically irrelevant issues related to $w/2$ and $\frac{k'+\ell}{2}$ being non-integer.

Let $v = \left\lfloor \frac{k'+\ell}{w} \right\rfloor$ and denote by w_- and w_+ the number of blocks from which we select, respectively, v and $v+1$ coordinates. Since it must be $w_- v + v_+(v+1) = w$, it must be

$$w_- = w(v+1) - k' - \ell, \qquad w_+ = k' + \ell - wv$$

Further, let $w_-^{(1)} = \lfloor w_-/2 \rfloor$ and $w_+^{(1)} = \lceil w_+/2 \rceil$: to build $\mathbf{z}_1$, we sample v coordinates from $w_-^{(1)}$ blocks and $(v+1)$ coordinates from $w_+^{(1)}$. With analogous meaning, we set $w_-^{(2)} = \lceil w_-/2 \rceil$ and $w_+^{(2)} = \lfloor w_+/2 \rfloor$.

[7] Here we again account for the fact that we can construct only balanced elements, due to the meet-in-the-middle enumeration.

Then, all the relevant quantities for the enumeration-based algorithm are given by:

$$\Pr\left[|\mathbf{z}_j| = p/2\right] = \sum_{i=\max\left\{0\ ;\ \frac{p}{2}-w_+^{(j)}\right\}}^{\min\left\{p/2\ ;\ w_-^{(j)}\right\}} \binom{w_-^{(j)}}{i}\left(1-\frac{v^{(j)}}{b}\right)^{w_-^{(j)}-i}\left(\frac{v^{(j)}}{b}\right)^{i}$$

$$\binom{w_+^{(j)}}{\frac{p}{2}-i}\left(1-\frac{v^{(j)}+1}{b}\right)^{w_+^{(j)}-\left(\frac{p}{2}-i\right)}\left(\frac{v^{(j)}+1}{b}\right)^{\frac{p}{2}-i}.$$

$$|L_j| = \sum_{i=\max\left\{0\ ;\ \frac{p}{2}-w_+^{(j)}\right\}}^{\min\left\{p/2\ ;\ w_-^{(j)}\right\}} \binom{w'_j}{i}\binom{w''_j}{p/2-i} v_j^i (v_j+1)^{p/2-i}.$$

The time complexity of the algorithm is

$$\frac{T_{\text{it}}}{\Pr\left[|\mathbf{z}_1| = p/2\right]\cdot\Pr\left[|\mathbf{z}_2| = p/2\right]} = \frac{n\left((n-k')^2 + \left(|L_1| + |L_2| + |L_1|\cdot|L_2|/2^{\ell}\right)\right)}{\Pr\left[|\mathbf{z}_1| = p/2\right]\cdot\Pr\left[|\mathbf{z}_2| = p/2\right]}.$$

B Experiments

In order to confirm our analysis, especially with respect to Remark 4.1 we provide proof of concept implementations of the the permutation-based and enumeration-based regular-ISD algorithms, available at `github.com/Memphisd/Regular-ISD`. In the experiments, we considered different RSD parameter regimes, namely, varying code rate and w chosen according to the uniqueness bound (regime I) as well as fixed code rate and varying w (regime II). For each considered parameter set, we generated 100 random instances and measured the number of resulting collisions in an iteration of the Enum algorithm and compared those against the theoretical prediction. Thereby, the optimization parameters (namely, parameters p and ℓ) have been chosen in order to minimize the time complexity. For the list sizes, we used the formulas from Appendix A.2.

In all considered experiments, we verified that list sizes match the expectation from the uniform random case on reduced size instances, as indicated by Remark 4.1. A subset of the used parameters and resulting list sizes are reported in Table 7. The code to rerun the experiments, as well as the results for additional parameter sets are made publicly available via the GitHub repository.

(n, k, w)	Rounding Issues Remain		Rounding Issues Resolved	
	Perm	Enum	Perm	Enum
$(2^{10}, 652, 106)$	133	115	134	116
$(2^{12}, 1589, 172)$	131	110	132	111
$(2^{14}, 3482, 338)$	140	118	141	118
$(2^{16}, 7391, 667)$	149	126	149	126
$(2^{18}, 15336, 1312)$	150	126	150	126
$(2^{20}, 32771, 2467)$	164	138	164	139
$(2^{22}, 64770, 4788)$	165	140	165	141
$(2^{10}, 652, 57)$	94	77	95	78
$(2^{12}, 1589, 98)$	96	78	96	78
$(2^{14}, 3482, 198)$	103	84	103	84
$(2^{16}, 7391, 389)$	110	90	110	90
$(2^{18}, 15336, 760)$	114	93	114	94
$(2^{20}, 32771, 1419)$	119	98	119	99
$(2^{22}, 64770, 2735)$	123	103	123	103
$(245760, 245460, 15)$	179	127	179	144
$(40960, 40660, 20)$	172	126	172	137
$(7680, 7380, 30)$	166	132	166	136
$(1280, 860, 80)$	137	117	137	118
$(609728, 36288, 1269)$	164	140	164	140
$(10805248, 589760, 1319)$	176	157	176	157
$(1842, 825, 307)$	178	156	180	159

Table 6: Time complexities of Perm and Enum for the RSD instances from Tables 4 and 5, with and without rounding issues.

Regime	(n, k, b)	(p, ℓ)	Theoretical	Experimental		
				Min	Avg	Max
I	$(100, 50, 20)$	$(6, 9)$	10.58	10.50	10.58	10.64
	$(96, 52, 16)$	$(6, 10)$	10.74	10.62	10.74	10.83
	$(96, 57, 12)$	$(4, 7)$	9.27	9.14	9.28	9.41
	$(100, 65, 10)$	$(4, 8)$	9.26	9.14	9.26	9.37
	$(96, 67, 8)$	$(4, 9)$	8.52	8.25	8.52	8.67
	$(85, 63, 5)$	$(2, 2)$	7.75	7.51	7.77	8.01
	$(92, 73, 4)$	$(2, 4)$	6.38	5.73	6.37	6.67
II	$(100, 40, 5)$	$(2, 4)$	4.52	3.58	4.52	5.13
	$(98, 39, 7)$	$(2, 2)$	6.15	5.70	6.14	6.38
	$(99, 39, 9)$	$(2, 1)$	6.89	6.66	6.90	7.07
	$(99, 39, 11)$	$(2, 1)$	6.70	6.39	6.70	6.97
	$(91, 36, 13)$	$(4, 7)$	6.11	5.67	6.12	6.69
	$(90, 36, 15)$	$(4, 4)$	8.12	7.63	8.12	8.48
	$(85, 34, 17)$	$(4, 4)$	7.15	6.84	7.14	7.44
	$(95, 38, 19)$	$(4, 4)$	7.73	7.34	7.73	7.99

Table 7: Number of collisions for several RSD instances, for enumeration-based regular-ISD. The number of collisions are expressed in $\log_2$ units. For each parameter set, we have considered 100 random RSD instances.